KONSTRUKČNÍ ELEKTRONIKA A Radio

ROČNÍK X/2005. ČÍSLO 3

Amatérské RADIO

ŘADA B - PRO KONSTRUKTÉRY

ROČNÍK LIV/2005. ČÍSLO 3

V TOMTO SEŠITĚ

KONSTRUKČNÍ ELEKTRONIKA A RADIO

Vydavatel: AMARO spol. s r. o.
Redakce: Zborovská 27, 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 (Zborovská 27, 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.: 800 800 890.
Objednávky a predplatné v Slovenskej republike vybavuje MAGNET-PRESS Slovakia s. r. o., Šustekova 8, 851 04 Bratislava, tel.: 00421 2 / 6720 1931 - 33
email: predplatne@press.sk ; www.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, Zborovská 27, 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., Šustekova 8, 851 04 Bratislava, tel.: 00421 2 / 6720 1931 - 33 ; www.press.sk
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, MK ČR E 7443

© AMARO spol. s r. o.

Z dějin vědy a techniky

Historie elektřiny a magnetizmu

Albert Einstein - osobnost plná paradoxů

Letošní rok byl vyhlášen rokem fyziky. Dnes jistě nikdo nebude namítat nic proti myšlence, že k největším fyzikům, které lidstvo zná, patří Albert Einstein. Ale nebylo tomu tak vždy. Řada vědců nepochopila vůbec jeho geniální myšlenky, někteří byli zmanipulováni tehdejší nacistickou propagandou a pro jejich averzi stačilo, že se narodil v židovské rodině. V Německu byla dokonce založeno přísně árijská společnost Deutsche Physik, jejímiž členy byli také nositelé Nobelových cen, kteří před nástupem nacismu s Einsteinem spolupracovali, ale později se stali jeho zavilými odpůrci.

Einstein se narodil ve městě Ulmu v podhůří Alp 14. března 1879. Otec, Hermann Einstein, byl drobný obchodník, matka Paulina vypomáhala svému muži a starala se o domácnost. Zakrátko se celá rodina přestěhovala do Mnichova. Za rok poté se mu narodila setra. Rodinné vztahy ale nebyly právě vřelé, což Alberta dosti poznamenalo a mělo silný vliv i na jeho život v dospělosti.

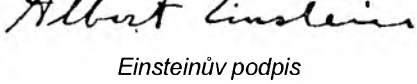
Na základní škole působil na učitele spíše dojmem zaostávajícího žáka. Jeho pozdější vztah k fyzikálním jevům měl pravděpodobně základ v tom, co pochytil u svého otce a hlavně strýce Jakuba - oba totiž nakonec pracovali v oblasti, kterou bychom dnes nazvali elektrotechnika a telekomunikace, i když pochopitelně na úrovni praktického poznání tehdejších malých obchodníků. Strýc Jakub mu pomohl také k získání základních poznatků o geometrii a algebře, spíše než to dokázala škola, když pro něj vymyslel hru založenou na jejich principu. Po matce zdědil lásku k hudbě a velmi dobře hrál na housle - dokonce v Americe několikrát i veřejně vystupoval na koncertech.

V Mnichově začal studovat gymnázium, ale i tam měl problémy. I přesto, že jeho studijní výsledky nebyly dobré, styděl se před ostatními za nízkou intelektuální úroveň svých rodičů. Rodiče již v té době žili v Itálii (Milano) a Albert nakrátko za nimi odejel a pokračoval ve studiu. Dlouho tam však nevydržel, měl zájem studovat v Curychu na vysoké škole technické, ovšem nepodařilo se mu složit přijímací zkoušku. Nakonec v roce 1895 odešel do švýcarského Aargau. Tam na něj zapůsobil silně profesor, který bral na byt chudé studenty a u kterého bydlel. Ten s nimi po večerech diskutoval o nejrůznějších filozofických otázkách. V Aargau studia dokončil a znovu se přihlásil v Curychu ke studiu na matematicko-fyzikálním ústavu vysoké polytechnické školy, kterou dokončil v roce 1900 a získal i učitelský diplom.

Einstein při hře na housle (kresba L. Pastemaka)

I v Curychu jej po absolvování provází neúspěch - nezískal místo asistenta na vysoké škole, o které se ucházel. Do Německa nemohl - musel by nastoupit vojenskou službu, a proto se vzdal německého občanství. Protloukal se proto, jak se dalo, krátce pracuje jako pomocný učitel na střední technické škole, pak v chlapeckém penzionátě. Teprve po dvou letech získal švýcarské státní občanství, což mu umožnilo nastoupit jako úředník na patentový úřad v Bernu. V roce 1903 se oženil s Maďarkou Milevou Maric, kterou poznal v Curychu. Ta mu do manželství přinesla i dceru, narozenou rok předtím, a kterou následně adoptoval. V roce 1904 se mu narodil syn, který se později stal profesorem na univerzitě v USA.

Současně ovšem studoval filozofii na curyšské univerzitě a tam v roce



Einsteinův podpis

1905 obhájil doktorát. Od dob, kdy studoval na technice, přemýšlel o teorii elektromagnetického pole, neboť v tehdy přednášené teorii bylo mnoho nejasností. Řešení nalezl ve speciální teorii relativity, jejíž principy zveřejnil v roce 1905. Je to teorie, která nahradila v té době již nevyhovující Newtonovu gravitační teorii, kterou nebylo možné aplikovat při rychlostech blížících se rychlosti světla.

Je s podivem, že vlastně všechny své významné práce dokončil a publikoval právě v tomto roce, v roce, ve kterém mu bylo teprve 26 let! Dvě nejvýznamnější se zabývaly popisem a vysvětlením fotoelektrického jevu a již zmíněnou speciální teorií relativity.

Tou druhou se stal známým na celém světě, rovnicí $E = m \cdot c^2$ zná téměř každý (i když vysvětlit její význam již může málokdo) a o ostatních jeho pracech se ví spíše jen v odborných kruzích. Proto se musí nezasvěcenému zdát paradoxní i to, že největší ocenění, Nobelovu cenu, získal za něco úplně jiného - za vysvětlení fotoelektrického jevu.

Skutečně, jeho speciální teorie relativity mu zakrátko přinesla velký věhlas. Přinesla také formulaci čtyřrozměrného časoprostoru. Sám o tom později prohlásil: „Můj intelektuální vývoj byl zpožděný, což způsobilo, že jsem začal přemýšlet o prostoru a času až když jsem dospíval, zatím co jiní si utvoří názor na tyto veličiny již v mládí".

Z teorie relativity Einstein odvodil Lorentzovu transformaci a dokázal její platnost pro všechny inerciální soustavy (soustavy, v nichž platí princip setrvačnosti). Na druhé straně mnoho tehdejších vědeckých kapacit tuto teorii naopak zatracovalo.

Einstein se v roce 1909 zúčastnil přednášek a shromáždění přírodovědců v Salzburgu, na kterém se seznámil s řadou v té době již velmi známých vědců, jako byl např. Planck či Born, se kterými pak udržoval čilé styky. Vzdal se nakonec svého místa v Bernu, neboť po udělení čestného doktorátu na univerzitě v Janově byl jmenován profesorem na univerzitě v Curychu.

Za dva roky poté odešel do Prahy, kde přednášel na německé univerzitě teoretickou fyziku a seznámil se s dalším významným fyzikem, Maxem Brodem. Byl známý tím, že od studentů nevyžadoval poplatky za studium a zkoušky, ale na druhé straně nevyhledával žádné osobní kontakty mezi Pražany, vyjma několika svých kolegů.

Účast na kongresu v Bruselu znamená další osobní kontakty - velkým dojmem na něj zapůsobila Marie Curie, Rutheford a Lorentz. Nakrátko se vrací na vysokou školu technickou do Curychu a v roce 1913 odchází na Fyzikální ústav císaře Viléma v Berlíně a stává se jeho ředitelem. Oslavován je na všech stranách. Přednáší na berlínské univerzitě, stává se členem Pruské akademie věd.

V roce 1915 uveřejnil obecnou teorii relativity, která dále zobecnila poznatky zveřejněné ve speciální teorii relativity. Jedním z jejích hlavních poznatků je, že hmota působí zakřivení prostoru, což má spojitost s účinky gravitace. Pak se na čas věnoval sepisování publikace, která by jeho teorii relativity přiblížila širší veřejnosti, neboť ani mnoho odborníků ji stále nemohlo pochopit.

V roce 1919 se účastní expedice Londýnské královské společnosti do Jižní Ameriky k pozorování zatmění Slunce, a tam se potvrdila správnost jeho teorie na zakřivení světelných paprsků v silných gravitačních polích a později i posuv spektrálních čar hvězd s velkou hmotností, který způsobují kmitočtové změny jejich světelného záření.

Své ženě se odcizil - ta odmítla odejet z Curychu, takže se po rozvodu oženil se svou sestřenicí.

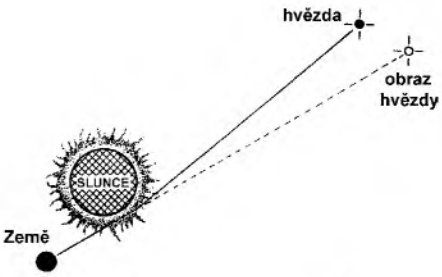
Roku 1921 získává Nobelovu cenu za vysvětlení fotoelektrického jevu. Jeho teorie světelných kvant, neboli fotonů, byla přijata i pozdějšími odpůrci.

V té době se také formuloval jeho světonázor. Již dříve se díky zprávám z obou stran válčící Evropy stal téměř fanatickým pacifistou, ale bojuje proti autoritářství všeobecně a hlasitě vystupuje proti některým tehdy hlásaným ideologickým směrům propagujícím nadřazenost - sestavuje např. manifest proti všeobecné branné povinnosti a vydal nakonec soubor svých statí pod názvem „Jak vidím svět". Nastupující hitlerovské zvůli však nebylo možné vzdorovat.

Na protest proti nastupujícímu nacismu se vzdává své funkce v Pruské akademii věd, je zbaven německého občanství, je mu zabaven (jako ostatním židům) majetek a dokonce snad byla vypsána odměna na jeho dopadení. V té době ovšem již přednášel v Belgii a v roce 1933 odjíždí natrvalo do Ameriky (kde pobýval již předtím - v roce 1930 tam přednášel) a působil pak jako profesor na Institutu pokročilých studií v Princetonu (překládáno i jako Ústav vědeckého výzkumu).

V roce 1936 umírá jeho druhá žena a Einstein vystupuje proti snaze vojensky využít atomovou energii - když však viděl nebezpečí plynoucí z expanze Německa v roce 1939 (v té době se zdálo že Hitler je neporazitelný a skutečně si podmaní celý svět), napsal spolu s dalšími vědci prezidentu Rooseveltovi dopis, který byl zárodkem projektu Manhattan s cílem vyvinutí atomové bomby. Sám se však takového projektu odmítl zúčastnit a nakonec litoval toho, že došlo k vynálezu a použití atomové pumy.

Roku 1940 získal americké občanství. V poválečné době se stal prezidentem Sdružení k odvrácení atomové války a všemožně se snažil přemluvit státníky, aby se dále nepracovalo na vývoji atomových zbraní - pochopitelně marně. V roce 1953 mu byl dokonce nabídnut úřad prezidenta nového státu Izrael, který odmítl.



Ohyb světelných paprsků hvězd v gravitačním poli Slunce - jeden z experimentů, které potvrzují Einsteinovy teorie

Zúčastnil se ale ještě ve stejném roce expedice do Súdánu, při které se potvrdila jeho teorie o zakřivení světelných paprsků.

Řada jeho vizí o tom, jak by měla být uspořádána společnost, je spíše utopistického charakteru. Ale nesmíme zapomenout, že v Americe žil v relativním přepychu, a při návštěvách na jiných kontinentech se setkával s bídou na každém kroku. Tvrdil např., že je špatné, že lidé kvůli tomu, aby si zajistili své nezbytné potřeby, musí pracovat tak, že jim již nezbývá čas ani energie na vlastní aktivity.

Také jeho pohled na židovstvo vychází ze židovské úcty a respektu ke vzdělání, kterého dosahovali i při nedostatku nadání svou disciplínou, pílí a odhodláním. Židy považoval za hybnou sílu světového rozvoje.

Do koce života se pak snažil vytvořit jednotnou teorii pole, která by v sobě zahrnovala gravitaci i elektromagnetickou interakci. Tento problém však přetrvává nedořešen dodnes.

V roce 1954 vážně onemocněl a 18. dubna 1955 v Princetonu zemřel.

Z mnoha životopisů publikovaných v časopisech i na internetu zpracoval

QX

Einstein na plachetnici v USA (1935)

Literatura

[1] *Kuzněcov, B. G.:* Ejnštejn. Moskva 1963.

[2] *Herneck, F.:* Albert Einstein. Berlin 1963.

PRAKTICKÉ KONSTRUKCE PRO USB S MIKROŘADIČEM AN2131

Ing. David Matoušek

Tento článek ukazuje praktické použití mikrořadiče AN2131 (označovaného též jako EZ-USB), který v sobě seskupuje jádro USB a procesor typu 8051.

Kromě nutného popisu základních vlastností, registrů a vestavěných periférií si můžete vyrobit vývojový kit USB2131KIT, který vám usnadní první kroky při vývoji zařízení USB. Méně zkušení amatéři si mohou kit objednat přímo od autora.

Na jednodušších příkladech i složitějších aplikacích se naučíte vytvářet vlastní přístroje řízené z USB.

1. Úvod do USB

Na tomto místě si neklademe za cíl podat podrobné charakteristiky sběrnice USB. Jen stručně vysvětlíme základní pojmy. Jedná se o nutný popis důležitý pro pochopení dalších pasáží.

Základní výhody USB

- Sériová sběrnice,
- jednotný kabel obsahující kromě datových linek i napájecí vodiče (odběr až 500 mA),
- relativně velká přenosová rychlost (12 Mb/s ve standardu USB 1.1 nebo 480 Mb/s ve standardu USB 2.0),
- velký počet připojitelných zařízení (při použití rozbočovačů - hubů až 127),
- skutečný plug & play (po připojení vyhledá operační systém žádaný ovladač; možnost odpojovat a připojovat zařízení za chodu počítače).

Konektor USB

Konektory USB se vyrábějí ve dvou variantách - obr. 1.1. **Konektor typu A** najdeme v počítači, **konektor typu B** je pak na straně zařízení. Oba konektory mají 4 vývody.

Poměrně podstatnou zvláštností je skutečnost, že se jedná o sériovou sběrnici. Data se přenášejí po linkách **D+ a D-** ve vzájemně invertované podobě. Jedná se tedy o diferenční signály. Tímto způsobem se výrazně zmenší rušení přenášených dat, takže přípojná vzdálenost může být až 5 m. Napěťové úrovně na datových linkách jsou v rozmezí 0 až 3,3 V.

Hodinový signál není přenášen, je rekonstruován ze signálů datových linek. Každé zařízení má definovanou přenosovou rychlost. Pro případ mikrořadiče **AN2131** se používá přenosová rychlost 12 Mb/s (full speed).

Kromě datových signálů jsou k dispozici napájecí vodiče **GND** a **UCC**. Napětí se podle specifikace může pohybovat v rozmezí 4,4 až 5,25 V. Obvyklé je, že mikrořadič používá zmenšené napájecí napětí 3,3 V. Proto musí být použit stabilizátor s malým průchozím úbytkem (low-drop). Na našem trhu připadá v úvahu typ **LM1084IT-3.3**.

Hostitel je pán (Host is Master)

Heslo „Hostitel je pán" je základní koncept funkce USB.

Sběrnice USB má jediný řídicí obvod (pán, master), od něhož vycházejí všechny aktivity. V případě zařízení připojovaných k počítači je **„pánem sběrnice"** řadič USB v počítači. Takže žádné zařízení připojené na sběrnici USB nemůže vysílat samo, ale až na pokyn, který vyjde z počítače.

Tento způsob komunikace dosti mění klasický pohled na ovládání zařízení. Na rozdíl od zařízení připojovaných na sériové nebo paralelní porty nelze používat přerušení. Zařízení USB si prostě nemůže vyžádat přednostní pozornost ze strany počítače.

Číslo vývodu	Význam
1	+5 V (UCC)
2	Data+ (přímá data)
3	Data– (negovaná data)
4	GND (zem)

Obr. 1.1. Konektory USB

Enumerace

Enumerace (vyčítání charakteristik zařízení) je jedním z „magických prvků" funkce sběrnice USB.

Po připojení zařízení se nejdříve zjistí přenosová rychlost a tou se ze strany počítače vyšle požadavek na zjištění charakteristik zařízení (nejdůležitější je patrně VID a PID, viz níže).

Podle zjištěných informací pak systém nahraje do paměti počítače odpovídající ovladač. Pokud není ovladač instalován, vyzve operační systém k jeho instalaci.

Po odpojení zařízení je ovladač uvolněn z paměti.

Skutečnost, že systém zjišťuje typ zařízení, pak dovoluje připojovat/odpojovat zařízení USB na libovolný port počítače kdykoli potřebujeme.

VID a PID

VID (Vendor ID, identifikační číslo výrobce) a PID (Product ID, identifikační číslo výrobku) zajišťují jednoznačné rozpoznání připojeného zařízení. Na základě těchto čísel a údajů v informačních souborech instalace (přípona **INF**) pak systém zavádí příslušný ovladač.

Identifikátory **VID** jsou přidělovány centrálně organizací USB (**www.usb.org**) všem výrobcům zařízení. Např. firma **Cypress** (výrobce mikrořadiče AN2131) má pro mikrořadič **AN2131** stanoveno identifikační číslo 0x0547 (547 hexadecimálně).

Identifikátory **PID** jsou pak určovány samotným výrobcem. Výrobce tak rozlišuje jednotlivé typy zařízení. Např. výchozí hodnota PID pro mikrořadič **AN2131** je stanovena jako 0x2131 (2131 hexadecimálně).

Typy přenosů

USB definuje celkem čtyři typy přenosů, které se liší použitelnou šíří přenosového pásma a zabezpečením dat:

• **Hromadný přenos** (Bulk Transfer) - je určen pro přenosy větších objemů dat se zajištěnou kontrolou platnosti (automatický mechanismus pro opětovné vysílání chybně přijatých dat); tento typ není použitelný pro časově kritické operace (spouští se v okamžiku, kde je sběrnice volná). Prakticky je používán např. pro tiskárny nebo scannery.

• **Přenos s přerušením** (Interrupt Transfer). Přestože zařízení nemohou generovat přerušení, chová se tento typ přenosu podobně jako přerušení. Přenos s přerušením je použitelný pro přenosy kratších dat v periodicky se opakujících časových intervalech. Prakticky se tento typ přenosu používá u pomalých zařízení, která vyžadují stálé sledování stavu (myš nebo klávesnice).

• **Izochronní přenos** (Isochronous Transfer) - je určen pro přenos časově kritických dat bez zabezpečení jejich platnosti. Typicky se používá např. u externích zvukových karet, u kterých by prodleva v odesílání dat způsobila větší chybu než neplatnost dat.

• **Řídicí přenos** (Control Transfer) - slouží pro konfiguraci zařízení a vysílání řídicích příkazů. Přenosová rychlost je vysoká (tento přenos má nejvyšší prioritu) a je zabezpečena kontrola platnosti dat.

Endpoint (koncový bod)

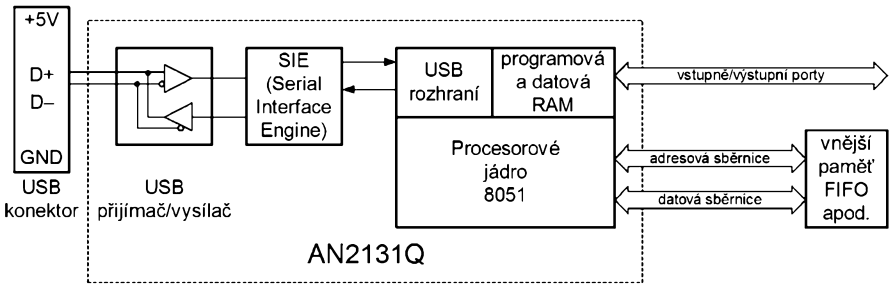
Specifikace sběrnice USB definuje **endpoint** (koncový bod) jako zdroj nebo noru pro data. Prakticky je endpoint představován vyrovnávací pamětí **FIFO** (chová se jako fronta: data uložená do fronty se odebírají ve stejném pořadí, v jakém přišla), která je plněna nebo vyprazdňována bajty z USB.

Hostitel vybírá endpoint zařízení tak, že pošle čtyřbitovou adresu doplněnou bitem určujícím směr toku dat. Takže zařízení USB může disponovat až 16 vstupními endpointy a 16 výstupními endpointy (protože 4 bity umožní adresovat 16 endpointů, první polovina je vstupní a druhá výstupní).

Z pohledu mikrořadiče **AN2131** je endpointem buffer (vyrovnávací paměť) pro příjem nebo vysílání dat přes USB. Data čteme přes výstupní buffer (výstupní směr se stanovuje vzhledem k hostiteli), zápis dat pro vysílání zajišťujeme přes vstupní buffer (je vstupní z hlediska hostitele).

2. Popis mikrořadiče AN2131Q

V této kapitole se budeme zabývat pouze jednou z variant mikrořadiče AN2131, která je označena jako **AN2131Q**. Tato varianta poskytuje nejvíce možností a je použita i v dále popsaném vývojovém kitu. Blokové schéma mikrořadiče je na obr. 2.1.



Obr. 2.1. Blokové schéma mikrořadiče AN2131Q

Mikrořadič AN2131Q se vyznačuje těmito základními vlastnostmi:

• Používá **procesorové jádro typu 8051**; z toho vyplývá, že pro vývoj programů lze používat překladače stejné jako pro běžnou řadu 8051; dále lze zužitkovat znalosti programování procesorů typu 8051, protože tento typ je patrně nejznámnější,

• použité jádro typu 8051 má upravený řadič, který zajišťuje provádění strojových cyklů pouze během čtyř hodinových cyklů (8051 potřebuje na provedení jednoho strojového cyklu 12 hodinových cyklů); tato skutečnost poskytuje při stejném hodinovém kmitočtu **až 3x větší výpočetní výkon**,

• mikrořadič má vestavěnou násobičku kmitočtu; kmitočet standardního krystalu 12 MHz je vnitřně vynásoben 2x a na tomto kmitočtu pak pracuje jádro,

• mikrořadič poskytuje mnoho vlastností známých z procesorů řady 8052 (což je rozšíření původní řady 8051): 3 čítače/časovače, 256 bajtů pro implementaci registrů, dva datové ukazatele (DPTR0, DPTR1),

• na čipu je také **8 KB RAM** určená pro uložení programu nebo dat (tato paměť je poměrně snadno dostupná přes USB),

• mikrořadič má zaveden **rychlý přenosový režim**, který zajišťuje přenosy mezi vnitřní pamětí FIFO a vnějšími zařízeními rychlostí až 3 MB/s,

• je rozšířen přerušovací systém (kromě přerušení zavedených u 8052 jsou připojena přerušení související s USB),

• mikrořadič poskytuje 4 zdroje resetu: při zapnutí napájení, při enumeraci zařízení přes USB, reset vyvolaný přes USB, reset 8051,

• jednotka řízení spotřeby poskytuje možnost zmenšit spotřebu mikrořadiče na 500 µA v režimu USB suspend,

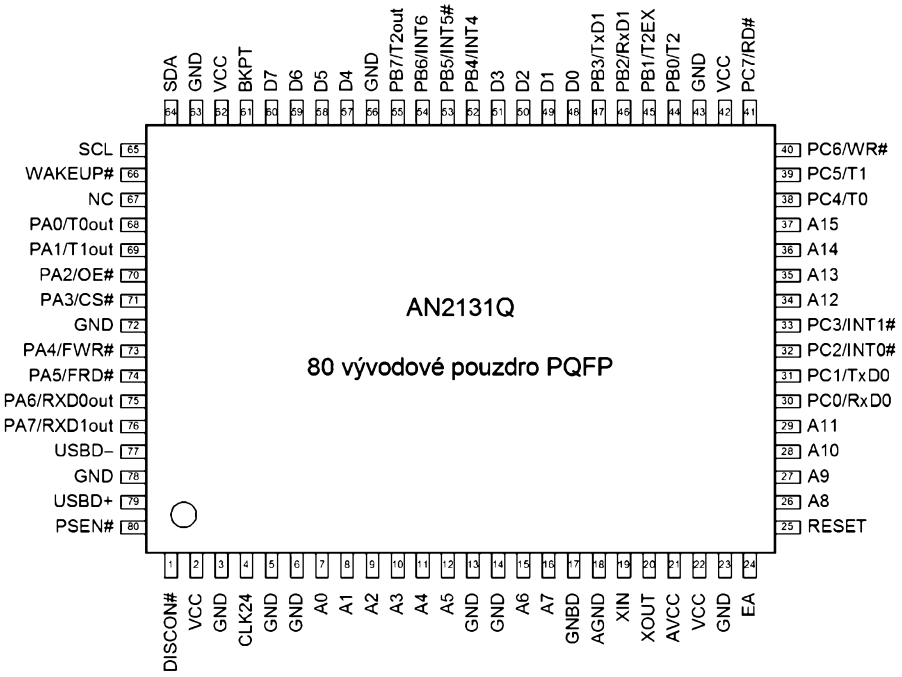
• disponuje **třemi obousměrnými porty označenými jako PA, PB a PC**; tyto porty však mají odlišné vlastnosti než klasické porty procesoru 8051,

• dva sériové kanály (UART) a jeden kanál I^2C,

• možnost připojit vnější paměť (nebo jiná vnější zařízení) přes adresovou a datovou sběrnici; vývody adresové (A0 až A15) a datové sběrnice (D0 až D7) nejsou na rozdíl od původního procesoru 8051 sdíleny s vývody portů,

• zmenšené napájecí napětí 3,3 V. Vývody portů jsou však tolerantní k napětí 5 V, takže na vstupy lze připojovat napěťové úrovně v rozmezí -0,5 až +5,8 V,

• 80vývodové pouzdro PQFP (pro povrchovou montáž; mikrořadič AN2131 se v jiném pouzdře nevyrábí).



Obr. 2.2. Vývody mikrořadiče AN2131Q (# značí invertovaný signál)

Stručný popis vývodů

Popis vývodů mikrořadiče AN2131Q je zřejmý z obr. 2.2.

Popis signálů je pro přehlednost podán formou několika tabulek (tab. 2.1 až tab. 2.4).

Zapojení krystalu

Mikrořadič používá hodinový kmitočet **12 MHz**.

Nejčastěji se připojuje krystal 12 MHz spolu s vazebními kondenzátory o kapacitě 27 pF mezi vývody **XIN** a **XOUT** (viz obr. 2.3).

Jinou možností je použít vnější zdroj hodin a připojit jej na vývod XIN.

Z obr. 2.3 je patrné, že na vývodu **CLK24** lze získat pomocný kmitočet 24 MHz (má smysl například v rychlém přenosovém režimu). Kmitočet 48 MHz (pouze uvnitř pouzdra) je používán pro synchronizaci se signály D+ a D- sběrnice USB.

Zapojení resetovacího obvodu

Zapojení resetovacího obvodu je velmi jednoduché (obr. 2.3). Výrobce doporučuje hodnoty součástek časovacího obvodu 100 nF a 10 kΩ.

Reenumerace a DISCON#

Na obr. 2.4 je uvedeno doporučené připojení vývodů souvisejících se sběrnicí USB.

Vývody **USBD-** a **USBD+** jsou napojeny na odpovídající vývody **D-** a **D+** přes omezovací rezistory s odporem 27 Ω.

Vývod D+ je navíc „vytažen" rezistorem o odporu 1,5 kΩ směrem k napětí 3,3 V. Toto napětí poskytuje vývod **DISCON#**. Tak je stanoveno, že se jedná o zařízení USB pracující v režimu full-speed (přenosová rychlost 12 Mb/s).

Jistě je zajímavé, proč není druhý pól rezistoru připojen přímo na napětí 3,3 V, ale na vývod DISCON#. Mikrořadič je totiž schopen vývod DISCON# řídit. Pokud se DISCON# uvede do stavu vysoké impedance (odpojí se), simuluje se stav odpovídající odpojení zařízení od USB sběrnice.

Tak je možno nahrát do mikrořadiče nový program (nebo změnit VID a PID) a přihlásit zařízení s novými vlastnostmi. Tato schopnost se označuje jako **reenumerace** (schopnost nové enu-

Tab. 2.1. Základní vývody

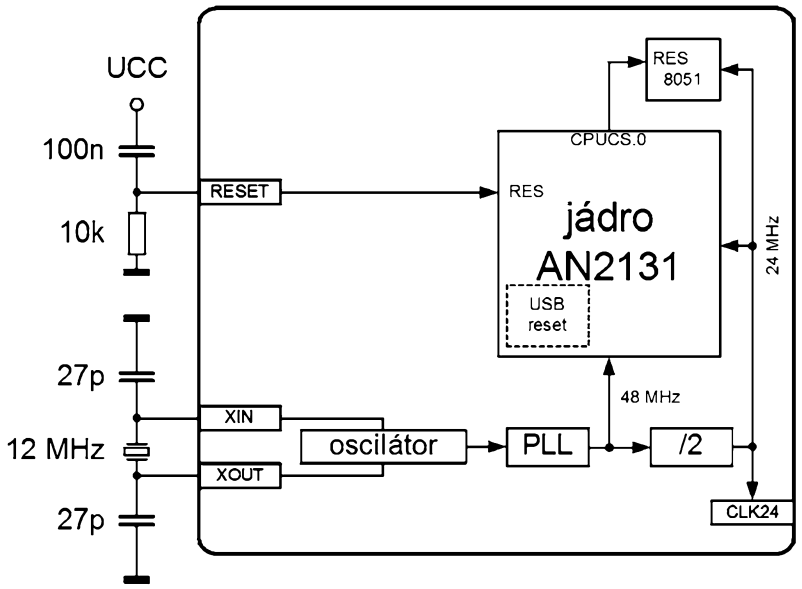
Vývod	Typ	Funkce
VCC	napájení	napájecí napětí (3,0 až 3,6 V)
GND	napájení	zem
AVCC	napájení	napětí pro analogovou část obvodu
AGND	napájení	analogová zem
XIN	vstup	vstup hodinového oscilátoru (viz obr. 2.3)
XOUT	výstup	výstup hodinového oscilátoru (viz obr. 2.3)
RESET	vstup	vnější resetovací vstup aktivní v log. 1 (viz obr. 2.3)

Tab. 2.2. Vývody pro USB

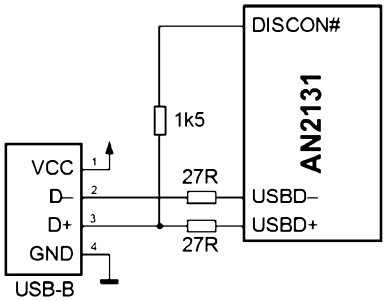
Vývod	Typ	Funkce
USBD+	vstup/výstup	připojí se na D+ přes odpor 27 Ω
USBD–	vstup/výstup	připojí se na D– přes odpor 27 Ω
DISCON#	výstup	umožní simulované odpojení mikrořadiče od USB (viz obr. 2.4)
WAKEUP#	vstup	je-li mikrořadič ve stavu suspend, umožní sestupná hrana přivedená na tento vývod spuštění krystalového oscilátoru a tedy ukončení režimu suspend. Je-li WAKEUP# = 0, je stav suspend blokován

Tab. 2.3. Porty a sběrnice srovnatelné s 8051

Vývod	Typ	Funkce
PA0 až PA7	vstup/výstup	vývody portu PA (multiplex, viz obr. 2.5)
PB0 až PB7	vstup/výstup	vývody portu PB (multiplex, viz obr. 2.5)
PC0 až PC7	vstup/výstup	vývodu portu PC (multiplex, viz obr. 2.5)
A0 až A15	výstup	adresová sběrnice pro vnější paměť
D0 až D7	vstup/výstup	datová sběrnice pro vnější paměť, lze použít i pro vysokorychlostní přenos
EA	vstup	při EA = 1 je program prováděn z vnější paměti, při EA = 0 je prováděn program z vnitřní RAM
RxD0	vstup	vstup sériového kanálu 0
TxD0	výstup	výstup sériového kanálu 0
INT0#	vstup	vstup vnějšího přerušení 0
INT1#	vstup	vstup vnějšího přerušení 1
T0	vstup	vstup čítače/časovače 0
T1	vstup	vstup čítače/časovače 1
WR#	výstup	strobovací signál zápisu do vnější paměti
RD#	výstup	strobovací signál čtení z vnější paměti
PSEN#	výstup	čtecí signál pro vnější paměť programu



Obr. 2.3. Zapojení krystalu a resetovacího obvodu



Obr. 2.4. Připojení vývodu DISCON#

Tab. 2.4. Rozšířené signály mikrořadiče AN2131Q

Vývod	Typ	Funkce
RxD1	výstup	vstup sériového kanálu 1
TxD1	výstup	výstup sériového kanálu 1
INT4	vstup	vstup vnějšího přerušení 4
INT5#	vstup	vstup vnějšího přerušení 5
INT6	vstup	vstup vnějšího přerušení 6
T0out	výstup	výstup čítače/časovače 0
T1out	výstup	výstup čítače/časovače 1
T2	vstup	vstup čítače/časovače 2
T2EX	vstup	hradlovací vstup čítače/časovače 2
T2out	výstup	výstup čítače/časovače 2
RXD0out	výstup	výstupní data sériového kanálu 0 (pouze v režimu 1)
RXD1out	výstup	výstupní data sériového kanálu 1 (pouze v režimu 1)
SDA	vstup/výstup	datový signál sběrnice I^2C
SCL	výstup	hodinový signál sběrnice I^2C
OE#	vstup	signál Output Enable vnější paměti
CS#	vstup	signál Chip Select vnější paměti
CLK24	výstup	výstup zabudované násobičky kmitočtu (kmitočet krystalu 12 MHz je násoben 2×), používá se například pro synchronizaci při rychlém přenosovém režimu
FWR#	výstup	strobovací signál zápisu při rychlém přenosovém režimu
FRD#	výstup	strobovací signál čtení při rychlém přenosovém režimu
BKPT	výstup	zajišťuje podporu pro hardwarové ladění

merace zajištěná výhradně ovládáním vývodu DISCON#; tedy programová simulace znovupřipojení zařízení s novými vlastnostmi).

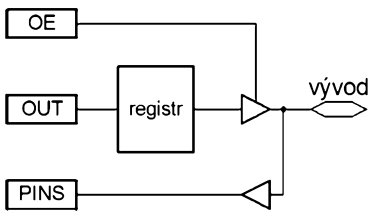
Porty

V procesorech typu 8051 jsou k dispozici 4 porty (P0 až P3) pracující v režimu označovaném jako kvazioboustranný. Znamená to, že s jistým omezením lze libovolný vývod používat buď jako vstup nebo jako výstup. Ovšem výstupní úroveň „log. 1" je realizována rezistorem a tak nelze odebírat v této log. úrovni proudy větší než asi 100 µA.

Naproti tomu mikrořadič AN2131 disponuje 3 porty (aby se to nepletlo, označují se jako PA až PC), které jsou skutečně obousměrné.

Vnitřní zapojení jednoho vývodu portu je uvedeno na obr. 2.5. Je zřejmé, že každý port je ovládán trojicí registrů:

• Registy **OEA**, **OEB** a **OEC** rozhodují o směru vývodu. Je-li např. nastaven nejnižší bit registru OEA, je vývod PA0 konfigurován jako výstup. Naopak, je-li nejnižší bit registru OEA vynulován, je vývod PA0 konfigurován jako vstup. Po resetu jsou tyto registry vynulovány, takže všechny porty jsou konfigurovány jako vstupní.



Obr. 2.5. Vnitřní zapojení portu

• Registry **OUTA**, **OUTB** a **OUTC** udržují data pro výstupní vývody. Takže pokud uvažujeme, že je vývod PA0 konfigurován jako výstup, nastavíme výstupní hodnotu tohoto vývodu nejnižším bitem registru OUTA.

• Registry **PINSA**, **PINSB** a **PINSC** umožňují číst stav vývodů bez ohledu na skutečnost, jestli je vývod konfigurován jako vstup nebo výstup. Chceme-li tedy zjistit stav vývodu PA0, musíme se zajímat o hodnotu nejnižšího bitu registru PINSA.

Určitou komplikací je skutečnost, že téměř všechny vývody portů mají **multiplexovanou funkci** s některou z vestavěných periferií. Např. vývod **PA0** může být také používán jako výstup čítače/časovače 0 (označen jako T0out).

O tom, jestli se vývod „přepne" do této funkce, pak rozhodují registry **PORTACFG**, **PORTBCFG** a **PORTCCFG**. Konkrétně pro vývod PA0 by nastavení nejnižšího bitu registru PORTACFG znamenalo, že by přešel do **alternativní funkce T0out**. Pokud je však nejnižší bit registru PORTACFG vynulován, pracuje jako nejnižší bit portu PA. Po resetu jsou tyto registry vynulovány, takže všechny vývody mají funkci portů.

Vestavěná sběrnice I^2C

Určitou méně tradiční vlastností mikrořadiče **AN2131** je vestavěná **sběrnice I^2C**. Signály **SDA** a **SCL** pak lze používat pro připojení různých obvodů I^2C, které rozšiřují schopnosti tohoto mikrořadiče.

Pro praktické použití je třeba připojit mezi vývody SDA a SCL a VCC rezistory s odporem v rozmezí 2,2 až 4,7 kΩ (připojíme-li rezistory vůči napětí +5 V, lze používat obvody I^2C vyžadující napájecí napětí +5 V).

Mnohem důležitější je však skutečnost, že na tyto vývody lze připojovat tzv. **konfigurační E^2PROM**. Do konfigurační E^2PROM lze zapsat hodnoty identifikátorů VID a PID nebo (při použití E^2PROM s větší kapacitou) může být z E^2PROM nahrán program mikrořadiče.

I^2C boot loader

Pokud **není** na vývodech **SDA** a **SCL** připojena E^2PROM (nebo pokud je obsah prvního bajtu odlišný od 0xB0 resp. 0xB2), vrací mikrořadič AN2131 tyto standardní hodnoty: **VID = 0x0547** (indikuje výrobce Cypress Semiconductor) a **PID = 0x2131** (indikuje mikrořadič AN2131).

Je-li E^2PROM připojena a první bajt má hodnotu 0xB0, jsou další 4 bajty použity pro stanovení nových hodnot VID a PID (**DID** značí identifikátor zařízení, lze jej používat jako další pomocný identifikátor; pro USB ale nemá prakticky žádný význam). Viz tab. 2.5.

Např. pro prvních 5 bajtů: 0xB0, 0x34, 0x12, 0xCD, 0xAB dostaneme VID = 0x1234 a PID = 0xABCD.

Je-li E^2PROM připojena a první bajt má hodnotu 0xB2, jsou další 4 bajty použity pro stanovení nových hodnot VID a PID. Další bajty pak tvoří bloky určené pro nahraní do vnitřní RAM. Viz tab. 2.6.

Každý blok začíná dvěma bajty určujícími délku bloku (maximální délka jednoho bloku je 1023 bajtů). Další dva bajty určují zaváděcí adresu (od této adresy se budou ukládat data z E^2PROM). Následují samotná data.

Poslední blok začíná hodnotou 0x80, další bajt má hodnotu 0x01 (pouze jeden bajt). Adresa je 0x7F92 (odpovídá registru **CPUCS**) a jediný datový bajt má obvykle hodnotu 0x00 (tato hodnota zapsaná do registru CPUCS způsobí reset mikrořadiče).

Významnou odlišností daného režimu je skutečnost, že zavedený program musí sám zajistit plnou obsluhu požadavků USB, nelze se tedy spolehnout

Tab. 2.5. Formát E^2PROM pro první bajt hodnoty 0xB0

E^2PROM adresa	Obsah
0	0xB0
1	dolní bajt VID
2	horní bajt VID
3	dolní bajt PID
4	horní bajt PID
5	dolní bajt DID
6	horní bajt DID

Tab. 2.6. Formát E²PROM pro první bajt hodnoty 0xB2

E²PROM adresa	Obsah
0	0xB2
1	dolní bajt VID
2	horní bajt VID
3	dolní bajt PID
4	horní bajt PID
5	dolní bajt DID
6	horní bajt DID
7	dolní bajt délky bloku
8	horní bajt délky bloku
9	dolní bajt adresy
10	horní bajt adresy
11	vlastní data
XX	dolní bajt délky bloku
XX+1	horní bajt délky bloku
XX+2	dolní bajt adresy
XX+3	horní bajt adresy
XX+4	vlastní data
YY	poslední blok (0x80)
YY+1	délka (0x01)
YY+2	horní bajt adresy (0x7F)
YY+3	dolní bajt adresy (0x92)
YY+4	hodnota CPUCS (0x00)

Tab. 2.7. Připojení adresovacích vývodů u jednotlivých typů E²PROM

Kapacita	Typ	A2	A1	A0
16 B	**24LC00**	–	–	–
128 B	**24LC01**	0	0	0
256 B	**24LC02**	0	0	0
4 KB	**24LC32**	0	0	1
8 KB	**24LC64**	0	0	1

na jádro USB obsažené v mikrořadiči. Vytvořit odpovídající obsah E²PROM pak patří mezi dosti složité úlohy.

Vlastní připojení E²PROM se liší podle použitého typu (každý typ totiž používá jiný způsob adresování). Situace je shrnuta formou tab. 2.7 (typ 24LC00 nemá adresovací vodiče).

Vstupy přerušení

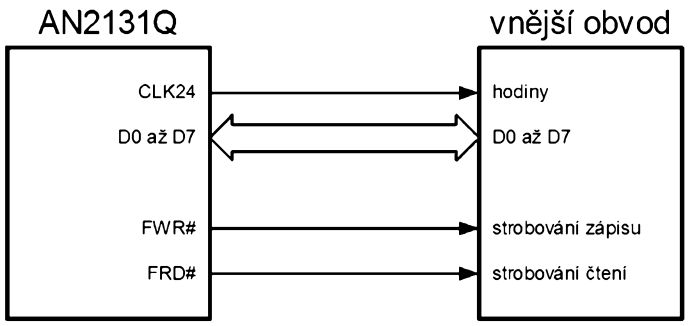
Procesory 8051 disponují pěti přerušeními: INT0 a INT1 (vnější přerušení z vývodů INT0# a INT1#), od čítačů/časovačů 0 a 1, od sériového kanálu (příjem znaku/odvysílání znaku). Mikrořadič AN2131 připojuje poměrně velké množství dalších přerušení:

- INT2 - vnitřní přerušení od USB,
- INT3 - vnitřní přerušení od řadiče I²C,
- INT4, INT5, INT6 - vnější vstupy INT4, INT5#, INT6
- PF1 - vývod WAKEUP#,
- sériový kanál 1 - vnitřní přerušení vyvolané příjmem/odvysíláním znaku na druhém sériovém kanálu,
- vnitřní přerušení od čítače/časovače 2.

Čítače/časovače 0 a 1

Čítače/časovače 0 a 1 pracují téměř stejně jako u klasického procesoru 8051 (viz [2]). Takže rozlišují tyto 4 režimy:

- 13bitový čítač/časovač,
- 16bitový čítač/časovač,
- 8bitový čítač/časovač s autoreloadem,
- dva 8bitové čítače/časovače (použitelné pouze pro čítač/časovač 0).



Obr. 2.6. Blokové schéma zapojení pro rychlý přenosový režim

Podstatnější odlišností je velikost pracovního kmitočtu, který se používá v režimu časovače. Připomeňme, že v režimu časovače se čítá kmitočet odvozený od hodinového taktu mikrořadiče. Pro procesor 8051 platilo, že se hodinový signál dělil 12 a takto získaný kmitočet pak byl použit jako zdroj synchronizace (např. pro krystal 12 MHz se používal kmitočet 1 MHz).

U mikrořadiče AN2131 se jako hodinový signál používá kmitočet 24 MHz (získaný vynásobením dvěma z krystalového oscilátoru 12 MHz). Pro čítání se pak použije kmitočet dělený 12 (je-li bit T0M = 0 pro čítač/časovač 0 nebo je-li bit T1M = 0 pro čítač/časovač 1) nebo dělený pouze 4 (je-li bit T0M = 1 pro čítač/časovač 0 nebo je-li bit T1M = 1 pro čítač/časovač 1). Takže jako synchronizační zdroj lze použít buď kmitočet 2 MHz nebo dokonce 6 MHz. Bity **T0M** a **T1M** jsou uloženy v registru **CKCON**.

Čítač/časovač 2

Čítač/časovač 2 pracuje téměř stejně, jako u rozšířené varianty procesoru 8051, která se obvykle označuje jako 8052. Popis nebude na tomto místě uveden, protože jej lze nalézt např. v [2]. Připomeňme pouze, že se rozlišují 4 režimy:

- 16bitový čítač/časovač,
- 16bitový čítač s jednotkou Input Capture,
- 16bitový čítač/časovač s autoreloadem,
- generátor přenosové rychlosti pro sériový kanál.

V režimu časovače se u mikrořadiče AN2131 jako hodinový signál používá kmitočet 24 MHz (získaný vynásobením dvěma z krystalového oscilátoru 12 MHz). Pro čítání se pak použije kmitočet dělený 12 (je-li bit T2M = 0) nebo dělený pouze 4 (je-li bit T2M = 1). Takže jako synchronizační zdroj lze použít buď kmitočet 2 MHz nebo dokonce 6 MHz. Bit **T2M** je uložen v registru **CKCON**.

Vnější paměť

Zvláštností mikrořadiče AN2131Q je skutečnost, že vnější adresová a datová sběrnice není multiplexována s vývody portů tak, jak tomu bylo u klasického procesoru 8051.

Vývody A0 až A15 představují adresovou sběrnici. Vývody D0 až D7 představují datovou sběrnici. Pouze vývody WD# a RD# jsou multiplexovány s vývody portů PC6 a PC7.

Rychlý přenosový režim

Skutečnost, že adresová i datová sběrnice není multiplexována s porty, umožnila realizovat rychlý přenosový režim (obr. 2.6). V tomto režimu mohou data „proudit" z nebo do mikrořadiče AN2131 rychlostí až 3 MB/s.

Rychlý přenosový režim je možný pouze při použití jednoho ze 16 izochronních registrů FIFO (povoluje se nastavením bitu **FISO** umístěného v registru **FASTXFR**). Čtení nebo zápis do izochronního registru FIFO způsobí, že jádro mikrořadiče přenáší data přes vnější datovou sběrnici **D0 až D7** a současně generuje strobovací signály pro čtení i zápis (**FRD#** a **FWR#**). Na jednu instrukci **MOVX** se tak přenese bajt dat mezi endpointem a vnějším zařízením. Vykonání instrukce MOVX trvá 2 strojové cykly, tedy 333 ns.

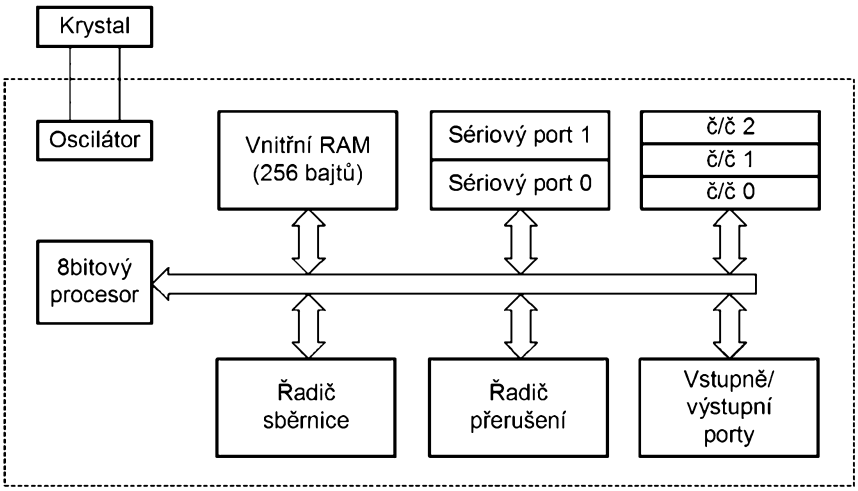
Navíc je možno používat režim označovaný jako **autoinkrementace datového ukazatele**. Připomeňme nejdříve, že instrukce MOVX pracuje s registrem **DPTR** (adresování je nepřímé, určeno obsahem tohoto ukazatele). V režimu autoinkrementace datového ukazatele se používá speciální registr **AUTOPTR**, který po každé operaci zvýší svůj obsah o 1. Nastavíme-li registr DPTR na adresu registru **AUTODATA**, bude velmi jednoduše zaručen přenos několika bajtů uložených v paměti za sebou.

Praktický příklad použití rychlého přenosového režimu najdete v kap. 12.

3. Programátorský model mikrořadiče AN2131Q

Pro vlastní práci s mikrořadičem AN2131Q je nutné podat popis jeho nejdůležitějších částí (např. paměti, registrů apod.).

Mikrořadič AN2131 má rozšířené jádro procesoru 8051 (obr. 3.1). Zajímavostí je zejména rozšířená vnitřní RAM (místo 128 bajtů má 256 bajtů), další čítač/časovač (označený jako č/č 2; stručně byl popsán ve 2. kapitole), dru-



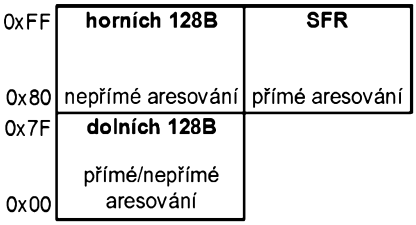
Obr. 3.1. Blokové schéma rozšířeného jádra 8051

hý sériový port a přepracované vstupně/výstupní porty (byly již popsány ve 2. kapitole).

Mapa paměti

Mapa vnitřní paměti RAM je uvedena na obr. 3.2. Platí všechna pravidla jako u procesoru 8051 (prvních 32 bajtů představuje 4 banky registrů R0 až R7; následuje 16 bajtů, které lze adresovat i bitově; dále je k dispozici 80 bajtů pro libovolné použití; posledním blokem jsou SFR - registry speciální funkce), ale kapacita paměti je zvětšena na 256 bajtů. To přináší určité změny:

- Dolních 128 bajtů se používá naprosto stejně jako u procesoru 8051 (je dovoleno přímé i nepřímé adresování),
- SFR jsou dostupné pouze přímým adresováním (prostor adres 0x80 až 0xFF je sdílen s rozšířenou RAM),

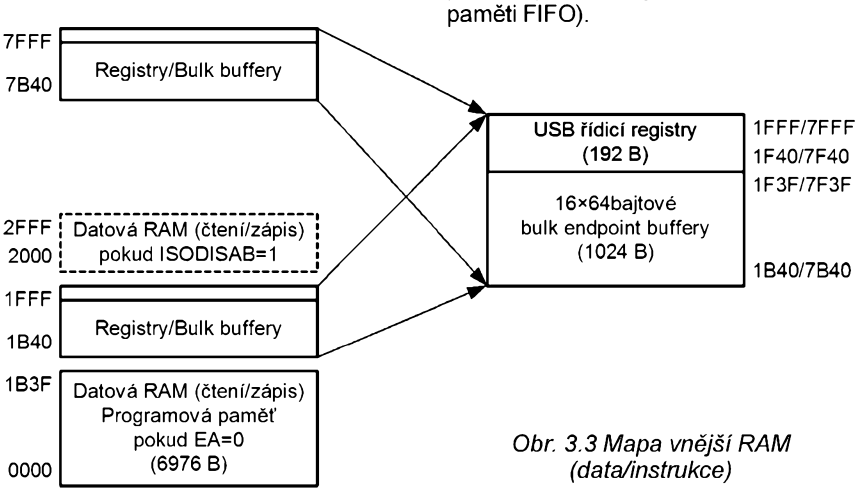


Obr. 3.2. Mapa vnitřní RAM (pouze data)

- rozšířená RAM je adresovatelná pouze nepřímo (tedy pomocí registrů R0, R1).

Mapa vnější RAM je uvedena na obr. 3.3. Přestože nebudeme uvažovat připojení vnějšího pamětového čipu k mikrořadiči AN2131Q (má k dispozici adresovou i datovou sběrnici), budeme tento blok paměti označovat jako vnější RAM. Důvod je jednoduchý, přístup k této paměti se z hlediska programu bude provádět instrukcemi **MOVX** (což odpovídá čtení dat z vnější datové paměti) a **MOVC** (což odpovídá čtení kódu instrukcí; zde se vnitřní a vnější paměť programu odlišuje aktivitou signálu EA). Vnější RAM má tyto vlastnosti:

- Adresy 0x0000 až 0x1B3F představují obecně použitelnou RAM velikosti 6976 bajtů pro uložení dat nebo instrukcí,
- Adresy 0x1B40 až 0x1FFF jsou zrcadleny na prostor 0x7B40 až 0x7FFF, je zde implementováno 16 bufferů pro hromadné přenosy (každý má velikost 64 bajtů) a dále se zde nacházejí řídicí registry pro USB. Z hlediska přístupu k bufferům je jedno, zda používáme počáteční adresu 0x1B40 nebo 0x7B40, ovšem ovládací program musí používat adresy počínaje 0x7B40.
- Adresy 0x2000 až 0x27FF lze používat pouze v tom případě, že není aktivován žádný ze 16 izochronních endpointů (v opačném případě tento prostor představuje 2 KB izochronní paměti FIFO).



Obr. 3.3 Mapa vnější RAM (data/instrukce)

Registry

Jak již bylo naznačeno formou obrázků obr. 3.2 a 3.3, má mikrořadič AN2131 k dispozici dva typy registrů:

- Jednak jsou to klasické **SFR** zavedené již u procesoru 8051 (jsou přidány některé nové). Pro lepší přehled je jejich seznam uveden v tab. 3.1.
- Další registry jsou implementovány ve vnější RAM a poskytují rozšířené funkce spojené s USB a řízením mikrořadiče AN2131. Pro jednoznačnost je bude označovat jako **registry EZ-USB** (viz tab. 3.2).

K tab. 3.1 je nutno dodat, že:

- mikrořadič má k dispozici dva datové ukazatele (datapointery DPTR0 a DPTR1), ovládají se po bajtech: **DPL0**, **DPH0**, **DPL1**, **DPH1**,
- protože instrukční soubor dovoluje manipulovat pouze s jedním datovým ukazatelem (označeným jako **DPTR**) zajišťuje registr **DPS** výběr mezi **DPTR0** a **DPTR1** (sudá hodnota pro DPTR0, lichá hodnota pro DPTR1), nejsnáze se vybraný datový ukazatel mění instrukcí **INC DPS**,
- registr **CKCON** jednak umožňuje volit hodiny pro čítače/časovače (pracující v režimu časovače) jako 2 MHz nebo 6 MHz; dále umožňuje volit dobu provádění instrukce **MOVX** (buď její provedení trvá 2 strojní cykly nebo až 9 strojních cyklů), to dává možnost upravovat rychlost tak, aby šlo připojit i pomalejší paměťový čip,
- protože u mikrořadiče není zaveden port P2, používá instrukce **MOVX @R0/R1** (adresování vnější datové paměti po stránkách) jako horní bajt adresy obsah registru **MPAGE**.

K tab. 3.2 je nutno dodat, že:

- registr **CPUCS** je řídicí a stavový registr procesoru, obsahuje 6 významných bitů (viz obr. 3.4):

bit **8051RES** - zápisem 1 do tohoto bitu vyvoláme reset, následným vynulováním je program spuštěn od adresy 0x0000; tento bit lze ovládat pouze přes USB (viz obr. 3.4),

bit **CLK24OE** - povoluje výstup hodinového signálu 24 MHz na vývod CLK24; tento bit nelze ovládat přes USB (lze ovládat jen vloženým programem),

bity **RV0 až RV3** - číslo revize procesoru (lze pouze číst),

- registry **PORTACFG**, **PORTBCFG** a **PORTCCFG** volí, jestli vývod pracuje jako jeden bit portu nebo má alternativní funkci (více bylo uvedeno v kapitole 2); vynulovaný bit značí, že se vývod chová jako bit portu; nastavený bit způsobí přechod do alternativní funkce,
- registry **OUTA**, **OUTB** a **OUTC** jsou registry pro zápis výstupních dat na vývody portů (je-li bit výstupní, použije se pro vybavení odpovídajícího stavu),
- registry **PINSA**, **PINSB** a **PINSC** umožňují číst stav vývodů portů bez ohledu na to, zda jsou konfigurovány jako vstupní či výstupní,

Tab. 3.1. SFR zavedené u mikrořadiče AN2131

Název	Adresa	Stručný popis
SP	0x81	ukazatel vrcholu zásobníku (Stack Pointer)
DPL0	0x82	datový ukazatel (Data Pointer) 0, dolní bajt
DPH0	0x83	datový ukazatel (Data Pointer) 0, horní bajt
DPL1	0x84	datový ukazatel (Data Pointer) 1, dolní bajt
DPH1	0x85	datový ukazatel (Data Pointer) 1, horní bajt
DPS	0x86	výběr datového ukazatele (Data Pointer Select)
PCON	0x87	registr řízení spotřeby, nastavení předděličky pro sériový kanál
TCON	0x88	konfigurace čítače/časovače 0 a 1 a vnějších vstupů přerušení
TMOD	0x89	výběr režimu pro čítače/časovače 0 a 1
TL0	0x8A	obsah čítače/časovače 0, dolní bajt
TH0	0x8B	obsah čítače/časovače 0, horní bajt
TL1	0x8C	obsah čítače/časovače 1, dolní bajt
TH1	0x8D	obsah čítače/časovače 1, horní bajt
CKCON	0x8E	volba hodin pro čítače/časovače, volba počtu cyklů provádění instrukce MOVX
SPC_FNC	0x8F	???
EXIF	0x91	příznaky přerušení od vnějších vstupů INT2 až INT5
MPAGE	0x92	nahrazuje standardní port P2 (z 8051) při nepřímém adresování vnější datové paměti ve stránkovacím režimu
SCON0	0x98	konfigurační registr pro sériový kanál 0
SBUF0	0x99	buffer pro sériový kanál 0
IE	0xA8	povolení přerušení
IP	0xB8	priorita přerušení
SCON1	0xC0	konfigurační registr pro sériový kanál 1
SBUF1	0xC1	buffer pro sériový kanál 1
T2CON	0xC8	konfigurační registr pro čítač/časovač 2
RCAP2L	0xCA	registr předvolby/záchytu čítače/časovače 2, dolní bajt
RCAP2H	0xCB	registr předvolby/záchytu čítače/časovače 2, horní bajt
TL2	0xCC	obsah čítače/časovače 2, dolní bajt
TH2	0xCD	obsah čítače/časovače 2, horní bajt
PSW	0xD0	stavový registr (příznaky)
EICON	0xD8	příznak přerušení vnějšího vstupu INT6 povolení přerušení a příznak přerušení pro vstup WAKEUP#
ACC	0xE0	akumulátor (střadač)
EIE	0xE8	rozšířený registr povolení přerušení
B	0xF0	pomocný registr (například při násobení a dělení)
EIP	0xF8	rozšířený registr priority přerušení

Tab. 3.2. Registry EZ-USB (výběr nejpoužívanějších)

Název	Adresa	Stručný popis
CPUCS	0x7F92	řídicí/stavový registr procesoru (viz obr. 3.4)
PORTACFG	0x7F93	konfigurace portu A (viz kapitolu 2)
PORTBCFG	0x7F94	konfigurace portu B
PORTCCFG	0x7F95	konfigurace portu C
OUTA	0x7F96	výstupní registr portu A (viz kapitolu 2)
OUTB	0x7F97	výstupní registr portu B
OUTC	0x7F98	výstupní registr portu C
PINSA	0x7F99	vývodu portu A (viz kapitolu 2)
PINSB	0x7F9A	vývodu portu B
PINSC	0x7F9B	vývodu portu C
OEA	0x7F9C	povolení výstupu na port A (viz kapitolu 2)
OEB	0x7F9D	povolení výstupu na port B
OEC	0x7F9E	povolení výstupu na port C
I2CS	0x7FA5	řídicí/stavový registr I^2C sběrnice (viz obr. 3.5)
I2DAT	0x7FA6	datový registr I^2C sběrnice (viz obr. 3.5)

- registry **OEA**, **OEB** a **OEC** volí mezi vstupním/výstupním režimem vývodů portů (0 pro vstup, 1 pro výstup),
- registry **I2CS** a **I2DAT** budou popsány v dalším textu, který se týká sběrnice I^2C.

Uvedené registry EZ-USB lze programově ovládat pouze pomocí datového ukazatele (pochopitelně, žádný z těchto registrů není adresovatelný bitově). Chceme-li např. přečíst stav portu PA, musíme nejdříve nastavit adresu 0x7F99 (což odpovídá registru **PINSA**) do registru **DPTR**. Následně, provedením instrukce **MOVX**, lze stav přečíst do akumulátoru (viz následující ukázka kódu):

```
PINSA   EQU 7F99H       ;definice PINSA
        .
        .
        MOV DPTR,#PINSA ;nastav DPTR
        MOVX A,@DPTR    ;vlastní čtení
                         stavu PA
```

Časování instrukcí

Mikrořadič AN2131 je, co se týká instrukčního kódu, plně kompatibilní s procesorem 8051. Takže pro vývoj programů lze používat stejné překladače jako pro oblíbený procesor 8051.

Určitou dosti podstatnou změnou je časování. Mikrořadič AN2131 se vyznačuje tím, že strojový cyklus trvá pouze 4 hodinové cykly oscilátoru (kdežto u procesoru 8051 trval jeden strojový cyklus 12 hodinových cyklů oscilátoru). Navíc jádro procesoru pracuje na kmitočtu 24 MHz, takže ve srovnání s procesorem 8051 taktovaným 12 MHz vychází strojový cyklus mikrořadiče AN2131 celkově 6x kratší.

Procesor 8051 vykoná instrukci za 1 až 4 strojové cykly. Mikrořadič AN2131 má tento násobitel ještě větší, pohybuje se od 1 do 9 strojových cyklů (např. u instrukce **MOVX** lze počet strojových cyklů volit). Nelze se tedy spoléhat na dobu provádění instrukcí rovnou jedné šestině doby procesoru 8051. Pokud je třeba znát přesný čas vykonání určitého sledu instrukcí, musíme nahlédnou do datasheetu mikrořadiče AN2131 (od strany B-14).

Vestavěná sběrnice I^2C

Mikrořadič AN2131 je také vybaven sběrnicí I^2C (vývody **SDA** a **SCL**). Tato sběrnice je podstatná jednak pro připojení vnější konfigurační E^2PROM (může obsahovat hodnoty VID, PID nebo celý program pro mikrořadič), ale také pro rozšíření systému například o A/D a D/A převodníky (které nejsou v tomto mikrořadiči k dispozici).

Na tomto místě nebudeme rozebírat funkci sběrnice I^2C, předpokládáme, že se jedná o známé informace.

Sběrnice I^2C je v mikrořadiči AN2131 ovládána dvěma registry: řídicím/stavo-

Bit	7	6	5	4	3	2	1	0
0x7F92	RV3	RV2	RV1	RV0	0	0	CLK24OE	8051RES
Čtení/zápis	R	R	R	R	R	R	R/W	R
Výchozí hodnota	RV3	RV2	RV1	RV0	0	0	1	1

Obr. 3.4. Registr CPUCS

Obr. 3.5. Registr I2CS

Bit	7	6	5	4	3	2	1	0
0x7FA5	START	STOP	LASTRD	ID1	ID0	BERR	ACK	DONE
Čtení/zápis	R/W	R/W	R/W	R	R	R	R	R
Výchozí hodnota	0	0	0	X	X	0	0	0

Obr. 3.6. Registr I2DAT

Bit	7	6	5	4	3	2	1	0
0x7FA6	D7	D6	D5	D4	D3	D2	D1	D0
Čtení/zápis	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Výchozí hodnota	X	X	X	X	X	X	X	X

vým registrem **I2CS** (obr. 3.5) a datovým registrem **I2DAT** (obr. 3.6).

Registr I2CS obsahuje tyto bity:

bit **START** - nastavení START = 1 způsobí, že po zápisu dat do registru **I2DAT** se generuje signál START na sběrnici, vynulování je provedeno automaticky,

bit **STOP** - nastavení STOP = 1 způsobí vygenerování signálu STOP okamžitě po odeslání zbývajících bitů z registru **I2DAT**, vynulování je provedeno automaticky,

bit **LASTRD** - nastavení LASTRD = = 1 se používá při posledním čtení proto, aby byl potlačen signál ACK (vynulování je provedeno automaticky). Nastavení tohoto bitu nezajišťuje automatické generování STOP signálu (bit STOP musí být rovněž nastaven),

bity **ID1, ID0** - tyto bity jsou pouze pro čtení, indikují typ připojené E^2PROM (00 odpovídá nepřipojené E^2PROM, 01 odpovídá paměti s 8bitovou adresou, 10 odpovídá paměti s 16bitovou adresou),

bit **BERR** - tento bit je pouze pro čtení, indikuje chybu sběrnice (nuluje se automaticky zápisem/čtením do registru **I2DAT**),

bit **ACK** - tento bit je pouze pro čtení, indikuje potvrzení vysílané podřízeným obvodem (dá se použít pro test, zda je obvod připojen),

bit **DONE** - tento bit je pouze pro čtení, indikuje konec přenosu.

Registr I2DAT obsahuje při vysílání data pro vysílání, při příjmu pak data přijatá sběrnicí.

Pro lepší vysvětlení uvedeme, jak odeslat data například na 8bitový D/A převodník **TC1320EOA** (adresa pro zápis 10010000B, adresa pro čtení 10010001B). V uvedeném příkladu nejsou pro jednoduchost testovány bity **BERR** a **ACK** (nezjišťuje se, zda je obvod skutečně připojen):

- nejdříve nastavíme bit **START**, proto zapíšeme do registru **I2CS** hodnotu **1000000B**,
- zapíšeme adresu obvodu **TC1320EOA** do registru **I2DAT**, proto zapíšeme do registru **I2DAT** hodnotu **10010000B**,
- čekáme na nastavení bitu **DONE** z registru **I2CS** (čekání na odeslání adresy),
- zvolíme režim funkce (pro zápis dat je třeba poslat nulový bajt) zápisem hodnoty **00000000B** do registru **I2DAT** a vyčkáme na opětovnou aktivaci bitu **DONE** z registru **I2CS**,
- zapíšeme žádaná data do registru **I2DAT** a vyčkáme na opětovnou aktivaci **DONE** bitu z registru **I2CS**,
- nyní můžeme zápisem do **I2DAT** vyvolat vyslání dalších datových bajtů,
- přenos ukončíme nastavením bitu **STOP** z registru **I2CS** (zápisem hodnoty **01000000B**).

Čtení dat bude poněkud komplikovanější, uvedeme příklad čtení dat z 8bitového A/D převodníku **PCF8591** (uvažujme, že vývody A2 až A0 jsou připojeny na VCC; adresa pro zápis pak bude 10011110B, adresa pro čtení 10011111B):

- nejdříve nastavíme bit **START**, proto zapíšeme do registru **I2CS** hodnotu **1000000B**,
- vyšleme adresu zápisem hodnoty **10011111B** do registru **I2DAT** a počkáme na aktivaci bitu **DONE**,
- přečteme jeden bajt z registru **I2DAT** (tím vyvoláme zahájení přenosu ze strany obvodu **PCF8591**), počkáme na **DONE**,
- nyní je možno čtením registru **I2DAT** zjistit výsledek převodu A/D (pokud chceme číst pouze jediný bajt, musíme před touto operací nastavit bit **LASTRD** zápisem hodnoty **00100000B** do registru **I2CS**), počkáme na **DONE**,
- předchozí krok lze opakovat pro čtení dalších bajtů,
- přenos ukončíme nastavením bitu **STOP** z registru **I2CS** (zápisem hodnoty **01000000B**),
- v registru **I2DAT** zůstane jeden bajt, ten je třeba přečíst (např. proto, aby se vynuloval bit **BERR**).

4. Endpoint 0

Endpoint 0 má pro sběrnici USB velký význam. Jedná se o řídicí endpoint, který je nutný pro libovolné zařízení USB. Hostitelský počítač totiž přes endpoint 0 vysílá řadu standardních dotazů.

Protože je mikrořadič AN2131 schopen enumerace bez vloženého firmware (řídicího programu), obsahuje jeho jádro logiku zajišťující provedení enumerace. Je-li však použita paměť E^2PROM obsahující kompletní firmware, musí být kromě jiného zajištěna i obsluha endpointu 0.

Výhodou endpointu 0 je především jeho obousměrnost. Mikrořadič AN2131 pro tento účel poskytuje dva 64bajtové buffery označené jako **IN0BUF** a **OUT0BUF**.

Dotazy USB

Dotazy USB posílané přes endpoint 0 jsou definovány přímo standardem USB. Ze strany hostitele se jedná 8bajtové pakety. Struktura je zřejmá z tab 4.1.

Standard USB definuje 13 standardních dotazů označených hodnotami bRequest = 0x00 až 0x0C. Kromě toho je definován dotaz **bRequest = 0xA0** (je velmi důležitý pro download/upload firmware do mikrořadiče AN2131, viz dále). Dotazy bRequest = 0xA1 až 0xAF jsou

Tab. 4.1. Formát osmibajtového paketu USB SETUP

Bajt	Označení	Význam
0	bmRequestType	typ dotazu, směr, příjemce
1	bRequest	konkrétní označení dotazu
2	wValueL	hodnota typu slovo (použití závisí na bRequest), dolní bajt
3	wValueH	hodnota typu slovo (použití závisí na bRequest), horní bajt
4	wIndexL	hodnota typu slovo (použití závisí na bRequest), dolní bajt
5	wIndexH	hodnota typu slovo (použití závisí na bRequest), horní bajt
6	wLengthL	počet bajtů pro přenos, dolní bajt
7	wLengthH	počet bajtů pro přenos, horní bajt

Tab. 4.2. Formát dotazu „Firmware download" (uložení programu)

Bajt	Označení	Hodnota	Význam
0	bmRequestType	0x40	výstup (ze směru hostitele)
1	bRequest	0xA0	download/upload
2	wValueL	AddrL	startovací adresa, dolní bajt
3	wValueH	AddrH	startovací adresa, horní bajt
4	wIndexL	0x00	nepoužito
5	wIndexH	0x00	nepoužito
6	wLengthL	LenL	počet zapisovaných bajtů, dolní bajt
7	wLengthH	LenH	počet zapisovaných bajtů, horní bajt

vyhrazeny výrobcem. Ostatní lze libovolně definovat pro vlastní potřebu.

Abychom zkrátili teoretickou část tohoto článku, nebudeme popisovat standardní dotazy USB. Zaměříme se na nejdůležitější (pro první pokusy s mikrořadičem AN2131) dotaz bRequest = 0xA0.

Firmware download

Tento dotaz USB slouží pro uložení firmware do RAM mikrořadiče AN2131.

Pokud tedy zapíšeme kód programu a provedeme překlad, můžeme výsledek překladu nahrát do mikrořadiče. Okamžitě po nahrání lze program spustit.

Tento dotaz také dovoluje ovládat většinu řídicích registrů (neboť jsou umístěny v RAM), např. lze provádět reset mikrořadiče.

Formát dotazu je uveden v tab. 4.2:

- dotaz začíná dvěma bajty 0x40, 0xA0 (značí download programu),
- následuje slovo určující startovací adresu (např. 0x0000 pro počátek RAM),
- další dva bajty nejsou použity a musí mít hodnotu 0x00,
- poslední je slovo určující počet zapisovaných bajtů,
- za paketem SETUP následují samotné bajty pro zápis (jejich počet musí odpovídat dříve stanovené délce).

Firmware upload

Tento dotaz USB slouží pro načtení firmware z RAM.

Pomocí tohoto dotazu tedy lze verifikovat download programu nebo číst libovolnou buňku RAM. To má velký význam pro čtení řídicích registrů (takto lze například sledovat registry typu **PINS** a číst tak stav portů).

Formát dotazu je uveden v tab. 4.3:

- dotaz začíná dvěma bajty 0x40, 0xC0 (značí upload programu),
- následuje slovo určující startovací adresu (např. 0x0000 pro počátek RAM),
- další dva bajty nejsou použity a musí mít hodnotu 0x00,
- poslední je slovo určující počet čtených bajtů,
- mikrořadič odesílá zpět žádaný počet bajtů.

Dotazy USB firmware download a firmware upload lze úspěšně používat pouze pro datovou/programovou RAM v rozsahu adres 0000 až 1B3F (viz obr. 3.3). Tímto způsobem nelze nastavovat registry SFR nebo registry EZ-USB.

5. Přípravky pro první pokusy s AN2131

V této kapitole budou předvedeny přípravky potřebné pro první pokusy s obvodem AN2131. Konstrukce jsou voleny tak, aby je zvládl i začínající amatér. Hotové přípravky lze také objednat u autora (kontakt je uveden na konci článku).

Plošné spoje byly navrženy ve vývojovém prostředí **Eagle 4.12** v edici Light.

Přípravek USB2131KIT

Před stavbou tohoto přípravku jsme zvažovali, kterou z variant AN2131 použít. Původně se jevila jako nejschůdnější varianta **AN2131SC** (nejmenší pouzdro), bohužel se nepodařilo obstarat tyto procesory v kusovém množství. Tak padla volba na variantu **AN2131QC**, sehnání tohoto mikrořadiče bylo poměrně jednoduché.

S ohledem na možnosti mikrořadiče AN2131QC byl vývojový kit vybaven konektory pro připojení přípravků na porty **PA**, **PB**, **PC**. Dále se zdálo výhodné mít k dispozici port **I²C**, proto je osazen další konektor. Tyto čtyři konektory mají 10 vývodů, kromě napájení (GND a 3,3 V - mikrořadiče AN2131 pracují se sníženým napětím) je k dispozici celý 8bitový port (pochopitelně u sběrnice I²C pouze signály SDA a SCL).

Resetovací obvod mikrořadiče AN2131QC je řešen obvyklým článkem *RC* (10 kΩ/100 nF).

Možnost použít vnější paměť (programovou nebo datovou) či přenos s velkou rychlostí jsem neuvažoval. Vždy je možné vytvořit další variantu kitu, která takové schopnosti bude mít. Šlo o to, vytvořit kit co nejjednodušeji a nejlevněji.

Další důležitou součástí je objímka pro **E²PROM**, která je připojena na sběrnici I²C. Vložená paměť umožňuje volit hodnoty VID a PID odlišné od standardních hodnot (VID = 0x0547 a PID = 0x2131). Do paměti lze také umístit celý program pro mikrořadič, který se po připojení přípravku na sběrnici USB automaticky nahraje do jeho RAM. Typ paměti E²PROM se volí pomocí rezistorů **R6** a **R7** (aktivita signálu A0 rozhoduje, zda bude správně rozeznána vložená sériová paměť). Pokud zapájíte oba rezistory, pak vzhledem k velikosti jejich odporů bude linka A0 připojena na úroveň „log. 0". Při vypájení R7 je linka A0 v úrovni „log. 1".

Poslední neméně důležitou součástí je stabilizátor. Stabilizátor typu **LM1084IT-3.3** se nakonec po počátečních peripetiích podařilo sehnat (dnes se snad již běžně prodává). Tento stabilizátor zmenšuje napájecí napětí získané ze sběrnice USB na velikost 3,3 V. Na rozdíl od jiných podobných stabilizátorů má výhodu malého průchozího úbytku (Low Drop). Tzn., že správně funguje i při minimálním rozdílu napětí mezi vstupem a výstupem. Některé stabilizátory potřebují průchozí úbytek až 3 V. Tento typ stabilizátoru vystačí s 1,3 V, což je vzhledem k možnostem sběrnice USB dostačující (minimální velikost napájecího napětí získaného z USB je 4,4 V; mikrořadič AN2131 pracuje již od 3,0 V).

Tak byla shrnuta volba použitých součástek a realizace kitu. Všechny součástky jsou v současnosti snadno k dostání. Čtenáři tedy nemusí mít strach, že by se jim nepodařilo kit vyrobit!

A nyní k vlastnímu zapojení. Schéma zapojení vývojového kitu **USB2131KIT** je na obr. 5.1.

Zapojení konektorů **PA**, **PB**, **PC** a **I2C** odpovídá rozmístění vývodů, které bylo zvoleno v knihách o mikrořadičích **AT89C2051**, **AT89S8252** a **AVR** (viz obr. 5.11). Mnoho přípravků publikovaných v [1], [2] a [3] lze používat i v souvislosti s vývojovým kitem **USB2131KIT**.

Jediným problémem může být skutečnost, že napájecí napětí je 3,3 V a ne 5 V, jak tomu bylo dříve. V mnoha případech to nevadí. Řada obvodů je schopna pracovat i při zmenšeném napětí, vstupy mikrořadiče AN2131 jsou tolerantní k úrovním TTL (vydrží vstupní napětí až 5,7 V).

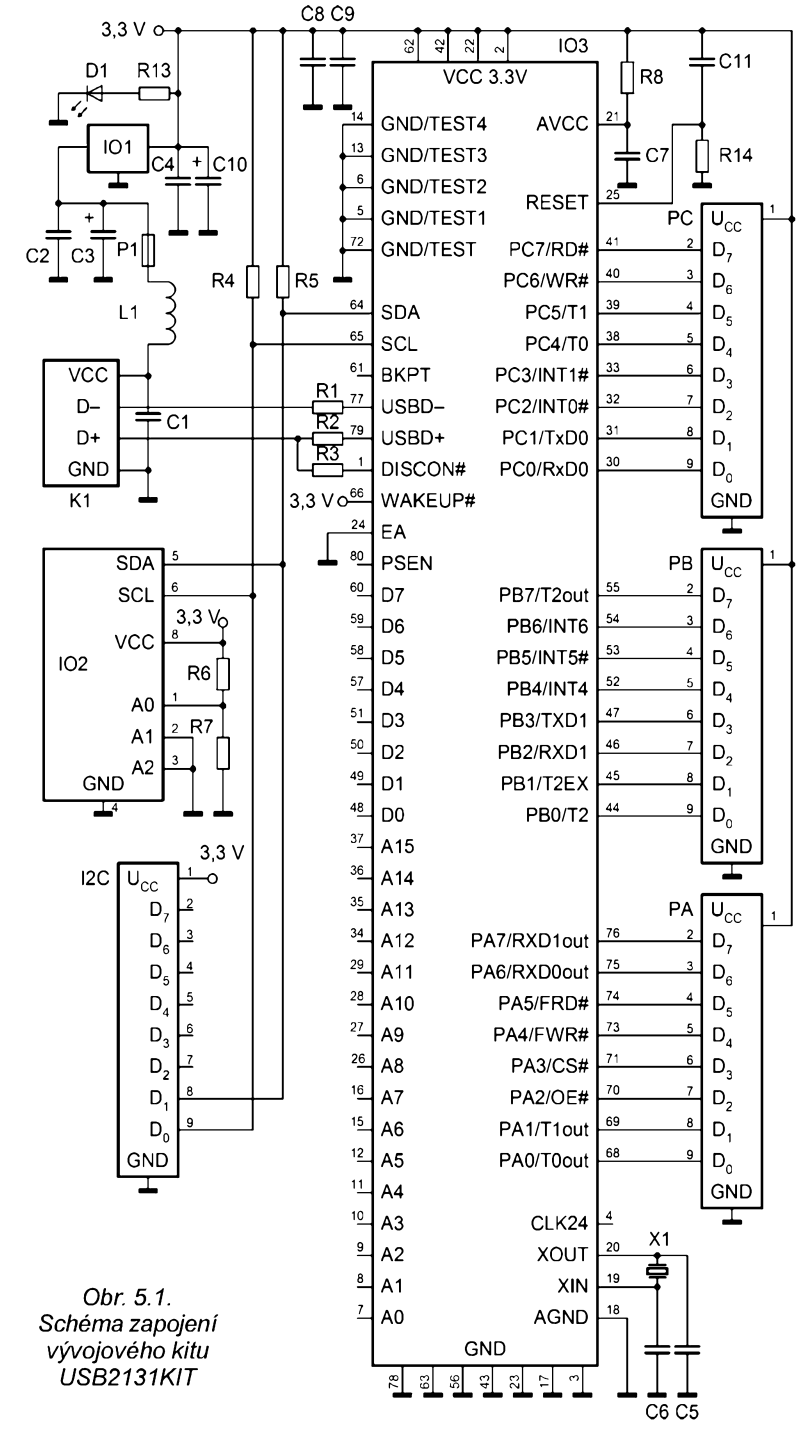
V konstrukci bylo nutné použít součástky pro povrchovou montáž (SMD). Důvod byl prostý - mikrořadiče AN2131 se vyrábí pouze v pouzdrech SMD. S ohledem na snahu zmenšit rozměry kitu byla i většina pasivních součástek (kondenzátrory a rezistory) zvolena v provedení SMD.

Všechny součástky jsou připájené na desce s jednostrannými plošnými spoji. Obrazec spojů je na obr. 5.2, rozmístění součástek na obou stranách desky je na obr. 5.3 a obr. 5.4.

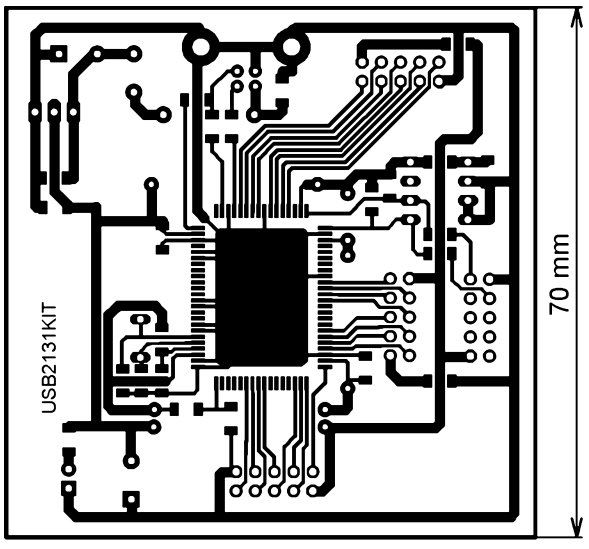
Při zapojování desky doporučujeme vyvrtat všechny díry a začít pájením součástek SMD. Nejprve zapájejte mikrořadič AN2131QC (je zapotřebí mikropáječka a trocha šikovnosti, aby vám cín nespojil sousední vývody) a poté další součástky. Nakonec osaďte běžné součástky (ty, co jsou umístěné ze strany součástek).

Tab. 4.3. Formát dotazu „Firmware upload" (čtení programu)

Bajt	Označení	Hodnota	Význam
0	bmRequestType	0xC0	vstup (ze směru hostitele)
1	bRequest	0xA0	download/upload
2	wValueL	AddrL	startovací adresa, dolní bajt
3	wValueH	AddrH	startovací adresa, horní bajt
4	wIndexL	0x00	nepoužito
5	wIndexH	0x00	nepoužito
6	wLengthL	LenL	počet čtených bajtů, dolní bajt
7	wLengthH	LenH	počet čtených bajtů, horní bajt



Obr. 5.1. Schéma zapojení vývojového kitu USB2131KIT



Obr. 5.2. Obrazec plošných spojů na desce vývojového kitu USB2131KIT (měř.: 1 : 1)

Seznam součástek
vývojového kitu USB2131KIT

R1, R2	27 Ω, SMD 1206
R3	1,5 kΩ, SMD 1206
R4 až R6	2,2 kΩ, SMD 1206
R7, R9 až R12	0 Ω, SMD 1206
R8	1 Ω, SMD 1206
R13	470 Ω, SMD 1206
R14	10 kΩ, SMD 1206
C1, C2, C4, C7 až C9, C11	100 nF/X7R, SMD 1206
C3, C10	470 µF/25 V, radiální
C5, C6	27 pF/NPO, SMD 1206
L1	33 µH, tlumivka axiální
X1	krystal 12,000 MHz
D1	LED červená, 5 mm, 200 mcd
IO1	LM1084IT-03,3
IO2	24LC01B–I/P
IO3	AN2131QC
P1	RXE025, vratná pojistka 0,25 A
I2C, PA, PB, PC	konektory MLW10G
K1	konektor USB1X90B PCB
objímka precizní DIP8 pro IO2	(1 ks)
chladič DO1A pro IO1	(1 ks)
deska s plošnými spoji USB2131KIT	

Pozn.: Rezistory R9 až R12 nejsou na schématu na obr. 5.1 nakresleny, jedná se o zkratovací propojky.

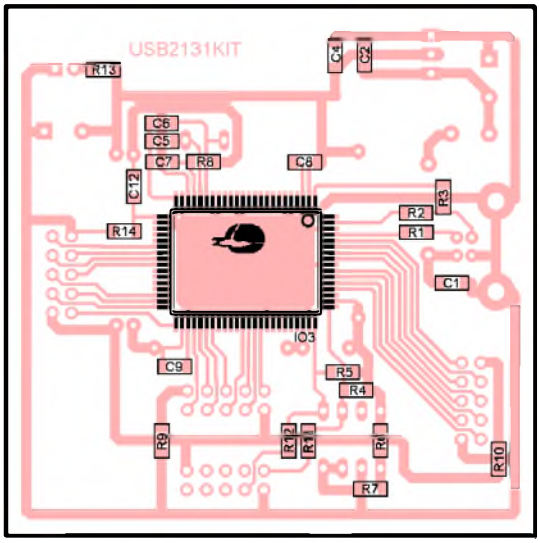
Přípravek AT8LED

Přípravek **AT8LED** patří nepochybně k legendám mé práce s mikrořadiči. Publikoval jsem jej již v [1]. Vlastně to byl první přípravek, který jsem kdy pro mikrořadiče vytvořil, i já jsem s ním začínal. Snad i proto si myslím, že jeho popis (přestože byl uveden již na mnoha místech) má svůj smysl.

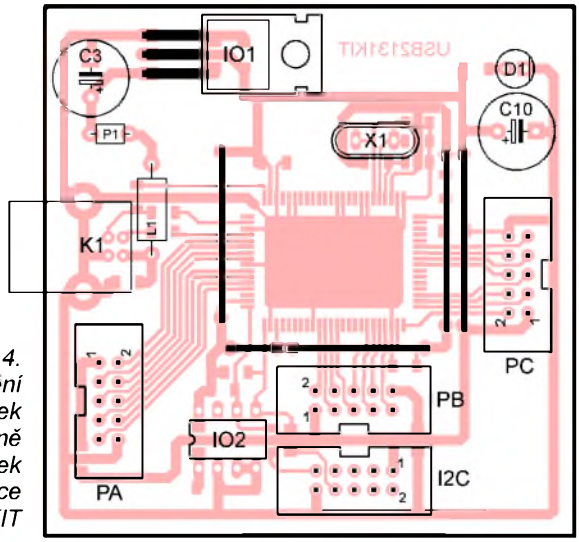
Jedná se o přípravek, který obsahuje skupinu osmi LED. Každá z LED není připojena na bit portu přímo, ale v cestě je budič sběrnice. Budič pracuje jako proudový zesilovač a díky tomu je zatížení vývodů portu minimální. V původní konstrukci se předpokládalo, že LED bude svítit při úrovni „log. 0“ (pro mikrořadiče vycházející z typu 8051 to má svůj význam). Z toho důvodu jsou LED připojeny anodami ke kladné napájecí sběrnici a katody pak sledují stav vývodů.

Zvolený budič sběrnice **74HCT245** je neinvertující, takže aby LED svítila, musí být na odpovídajícím vývodu skutečně stav „log. 0“. Pokud by se použil invertující budič (vstupní „log. 1“ by převedl na výstupní „log. 0“), svítila by LED při „log. 1“. Pokud uvažujete toto řešení, použijte místo budiče typu 74HCT245 typ **74HCT640**.

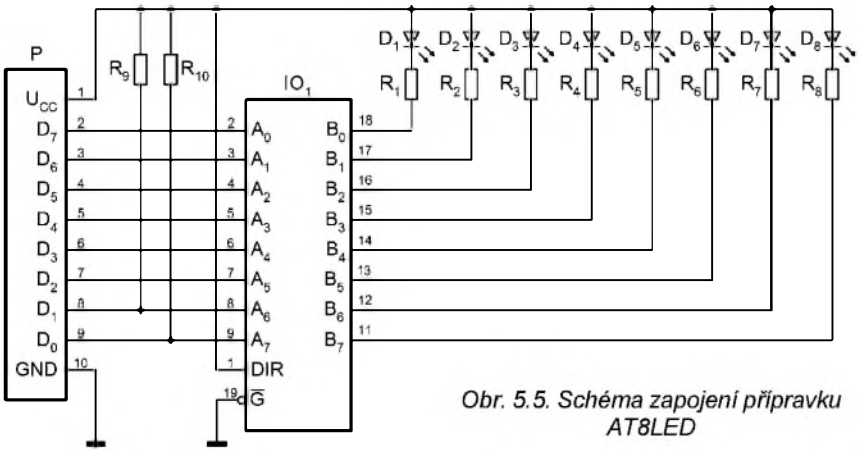
Zapojení přípravku již nebudu dále komentovat, na obr. 5.5 je schéma jeho zapojení (rezistory R9 a R10 pocházejí z původní konstrukce určené pro mikrořadič AT89C2051, ten totiž neobsahuje na vývodech P1.0 a P1.1 zdvihací rezistory).



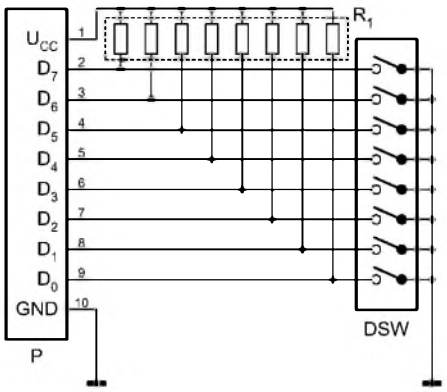
Obr. 5.3. Rozmístění součástek SMD na straně spojů na desce USB2131KIT



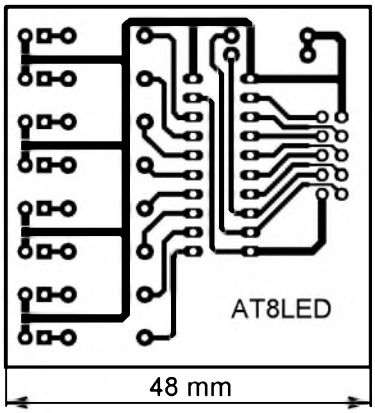
Obr. 5.4. Rozmístění součástek na straně součástek na desce USB2131KIT



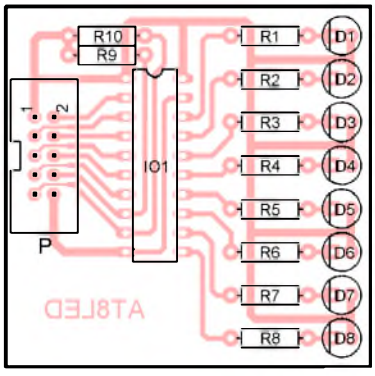
Obr. 5.5. Schéma zapojení přípravku AT8LED



Obr. 5.8. Schéma přípravku ATDIPSW2



Obr. 5.6. Obrazec spojů na desce přípravku AT8LED (měř.: 1 : 1)



Obr. 5.7. Rozmístění součástek na desce přípravku AT8LED

Přípravek je zkonstruován z běžných vývodových součástek na desce s jednostrannými plošnými spoji. Obrazec spojů je na obr. 5.6, osazovací plánek desky je na obr. 5.7.

Seznam součástek

vývojového kitu AT8LED

R1 až R8	330 Ω, miniaturní
R9, R10	10 kΩ, miniaturní
D1 až D8	LED červené, 5 mm, 200 mcd
IO1	74HCT245 (DIL20)
P	konektor MLW10G (nebo PSL10)

deska s plošnými spoji AT8LED

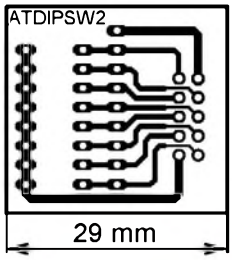
Přípravek ATDIPSW2

Přípravek **ATDIPSW2** má podobnou historii jako dříve popsaný přípravek **AT8LED**. Opět se jedná o starší konstrukci, je to jednoduchá skupina osmi spínačů umístěných v pouzdře DIP (**DIP8X**). Původně byla vynechána odporová síť **R1**, protože nebyla nutná (mikrořadiče typu 8051 mají vývody opatřené tzv. zdvihacími rezistory, které uvádějí vývod do stavu „log. 1"; pouze když vývod přímo spojíme s GND, bude na něm stav „log. 0").

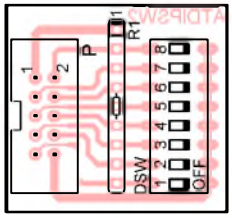
Mikrořadič AN2131 bohužel disponuje jiným typem portů než procesor 8051. Proto je odporová síť R1 nezbytná. Definuje totiž stav vývodu v případě, že je spínač rozpojen. Tehdy přečteme log. 1. Pokud je spínač sepnut (a připojí vývod na GND), přečteme z vývodu log. 0.

Kvůli této úpravě je přípravek z původního označení **ATDIPSW** přejmenován na **ATDIPSW2** (jako verze 2, tuto verzi můžete bez komplikací používat místo původního přípravku). Pokud máte přípravek **ATDIPSW** již zhotovený, můžete odporovou síť R1 poměrně snadno dopájet!

Schéma přípravku je na obr. 5.8. Je zkonstruován z vývodových součástek



Obr. 5.9. Obrazec spojů na desce přípravku ATDIPSW2 (měř.: 1 : 1)



Obr. 5.10. Rozmístění součástek na desce přípravku ATDIPSW2

na desce s jednostrannými plošnými spoji. Obrazec spojů je na obr. 5.9, rozmístění součástek na desce je na obr. 5.10.

Seznam součástek

vývojového kitu ATDIPSW2

R1	odporová síť 8x 10 kΩ
P	konektor MLW10G (nebo PSL10)
DSW	skupina osmi spínačů DIP8X

deska s plošnými spoji ATDIPSW2

Zapojení konektorů MLW10G na přípravku USB2131KIT

Pro lepší pochopení přidávám ještě na obr. 5.11 popis jednotlivých vývodů konektorů **MLW10G** (jsou to konektory PA, PB, PC a I2C na přípravku USB2131KIT).

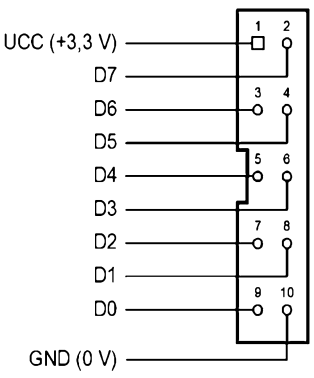
Pro konektor **I2C** platí, že jsou osazeny pouze vývody 1, 8, 9, 10 (tedy z datových vodičů jsou k dispozici pouze D0 a D1). Na vývodu D1 je signál **SDA** a na vývodu D0 pak signál **SCL**. Zapojení opět odpovídá přípravkům určeným pro obvody se sběrnicí I^2C, které byly popsány v [1] až [3].

Konstrukce propojovacího kablíku pro konektor MLW10G

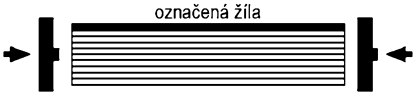
Konstrukce **propojovacího kablíku pro konektor MWL10G** je zřejmá z obr. 5.12.

Tento kablík je určen pro připojení přípravků AT8LED, ATDIPSW2 nebo jiných periferií k vývojovému kitu USB2131KIT.

Potřebujeme dva konektory **PFL10** a kousek (alespoň 15 cm) plochého kabelu s deseti žilami (AWG28-10). Oba konektory umístíme na kabel podle obrázku. Pak jeden konektor po druhém vložíme do svěráčku a stáhneme. Tím je mechanicky připevníme ke kabelu.



*Obr. 5.11. Popis vývodů konektorů **PA**, **PB**, **PC**, **I2C** na přípravku USB2131KIT*



Obr. 5.12. Konstrukce propojovacího kablíku pro konektor MWL10G

6. Výchozí ovladač EZUSB.SYS

Každé zařízení připojené k počítači (PC) nutně potřebuje **ovladač**. Ovladač je vlastně program vytvořený ve speciálním formátu (přípona SYS), který pomocí svých služeb zajišťuje ovládání zařízení.

V této kapitole si ukážeme použití standardního ovladače **EZUSB.SYS**, který poskytuje poměrně široký aparát služeb pro ovládání mikrořadiče AN2131. Dále budou popsány nejdůležitější funkce Windows API, které zajišťují použití zmíněného ovladače.

Ovladač a instalační soubor

„Když připojíte USB zařízení poprvé k počítači, stane se něco magického" (tolik citát z datasheetu mikrořadiče AN2131). Skutečně, systém Windows je tak inteligentní, že při připojení zařízení vyhledá odpovídající ovladač a zavede jej do paměti. Tak lze tedy volat služby ovladače a zajistit použití zařízení v systému.

Při prvním připojení zařízení, které dosud nebylo k počítači nikdy připojeno, musí proběhnout **instalace ovladače**. Uživatel v průběhu instalace vloží instalační CD–ROM (případně disketu), na kterém se nachází ovladač. V některých verzích (myslíme především Windows XP), může být také ovladač stažen z Internetu.

Jak však systém rozpozná, že příslušný soubor je skutečně tím správným ovladačem pro daný typ zařízení? K tomu slouží takzvaný **informační soubor instalace** (přípona INF).

Instalační soubor stanovuje, pro které zařízení je ovladač určen. Jedná se o textový soubor, který lze poměrně snadno vytvořit pomocí jednoduchého textového editoru (i když zápis některých částí se může být pro začátečníka zbytečně rozvláčný nebo krkolomný). Hlavní částí souboru je uvedení hodnot **VID** a **PID**, pro které lze příslušný ovladač používat. Pokud máme více typů zařízení, které používají stejný ovladač, může být v informačním souboru zapsáno více možných variant VID a PID. Níže jsou uvedeny dva příklady.

Nejprve je v tab 6.1 uvedena část informačního souboru, který zajišťuje instalaci ovladače pro zařízení s **VID = 0x0547** a **PID = 0x2131** (výchozí hodnoty pro mikrořadič AN2131).

Druhý příklad v tab. 6.2 ukazuje možnost, jak zapsat rozlišovací část informačního souboru pro více zařízení (všechna ale budou ovládána pomocí jediného ovladače). Každé zařízení se při instalaci může objevovat pod námi určeným označením, což určuje sekce **Strings**. Např. zařízení s **VID = 0x0547** a **PID = 0x0080** je označeno jako **DEV1**, při instalaci se pak v informačním okně zobrazí pod jménem **Zařízení 1**.

Shrňme to tedy. Pro instalaci je třeba mít minimálně dva soubory: **instalační soubor** (INF), který popisuje instalaci (a říká, pro které VID a PID je určen ovladač) a **samotný ovladač** (SYS).

Instalace ovladače EZUSB.SYS

Po tomto krátkém teoretickém úvodu (popis jednotlivých sekcí instalačního souboru by zabral několik stran, proto jej vynecháme) můžeme přistoupit k připojení vývojového kitu **USB2131KIT** (z kapitoly 5) k počítači.

Před tímto krokem důrazně doporučujeme následující úkony:

- zkontrolujte, zda jsou všechny součástky správně zapájeny,

*Tab. 6.1. Část informačního souboru, který zajišťuje instalaci ovladače pro zařízení s **VID = 0x0547** a **PID = 0x2131***

```
.
.
[Anchor]
%USB\VID_0547&PID_2131.DeviceDesc%=EZUSB, USB\VID_0547&PID_2131

[ControlFlags]
ExcludeFromSelect=USB\VID_0547&PID_2131
.
.
```

Tab. 6.2. Možnost, jak zapsat rozlišovací část informačního souboru pro více zařízení

```
.
.
[Anchor]
%USB\VID_0547&PID_0080.DeviceDesc%=DEV1, USB\VID_0547&PID_0080
%USB\VID_0547&PID_1002.DeviceDesc%=DEV2, USB\VID_0547&PID_1002
%USB\VID_0547&PID_2131.DeviceDesc%=DEV3, USB\VID_0547&PID_2131

[ControlFlags]
ExcludeFromSelect=*
.
.
[Strings]
Anchor="AnchorChips"
USB\VID_0547&PID_0080.DeviceDesc="Zařízení 1"
USB\VID_0547&PID_1002.DeviceDesc="Zařízení 2"
USB\VID_0547&PID_2131.DeviceDesc="Zařízení 3"
```

• ověřte ohmmetrem, zda mezi vývody konektoru USB není zkrat (mohl by se poškodit počítač).

Je-li vše v pořádku, musí se po připojení kitu **USB2131KIT** k počítači pomocí kabelu USB typu A-B objevit obvyklý dialog systému. Tento dialog (viz obr. 6.1) informuje uživatele o vyhledávání ovladače. Dále uvedené obrázky platí pro operační systém Windows 98.

Po stisku tlačítka **Další**, je zobrazen dialog podle obr. 6.2. V něm důrazně doporučujeme vybrat variantu, podle které musí adresář s ovladačem stanovit uživatel (např. operační systém Windows XP může v tomto případně instalaci řádně zkomplikovat).

Po stisku tlačítka **Další** je zobrazen dialog podle obr. 6.3. Po volbě varianty **Jiné umístění** (ostatní varianty doporučujeme zrušit) lze tlačítkem **Procházet** zvolit adresář s ovladačem (pomocný dialog pro výběr souboru není možno zavřít dříve, dokud nezvolíme adresář obsahující alespoň jeden instalační soubor).

V dialogu na obr. 6.3 je uveden adresář **USB2131KIT\OVLADACE\EZUSB**, který se nachází na mechanice CD-ROM označené jako disk **E**. Tato situace odpovídá uložení souborů na doprovodném CD-ROM k tomuto časopisu. **Vzhledem k cenové relaci časopisu není CD–ROM jeho součástí, můžete si jej objednat od autora - kontakt je uveden v závěru článku. Jinou možností je bezúplatné stažení obrazu CD-ROM z Internetu, adresa je opět uvedena v závěru článku.**

Po stisku tlačítka **Další** je zobrazen dialog dle obr. 6.4. V něm jej již zobrazeno jméno zařízení **USB2131KIT** (tak je zařízení označeno v instalačním souboru).

Po stisku tlačítka **Další** je instalace dokončena (viz obr. 6.5).

Pokud vše proběhlo v pořádku, najdeme nainstalované zařízení v dialogu **Systém** (vyvoláme jej pomocí nabídky Start: **Start|Nastavení|Ovládací panely|Systém|Správce zařízení**), viz obr. 6.6.

Po úspěšně provedené instalaci si můžeme říci, jak lze vyvolávat jednotlivé služby ovladače:

• Nejdříve je třeba získat handle zařízení pomocí funkce **CreateFile**,

• vlastní službu vyvoláme funkcí **DeviceloControl**,

• pro vyvolání služby je třeba znát její tzv. CTL kód, získáme jej makrem **CTL_CODE**, a pochopitelně znát, jak pro danou sloužbu vyplnit parametry funkce **DeviceloControl**,

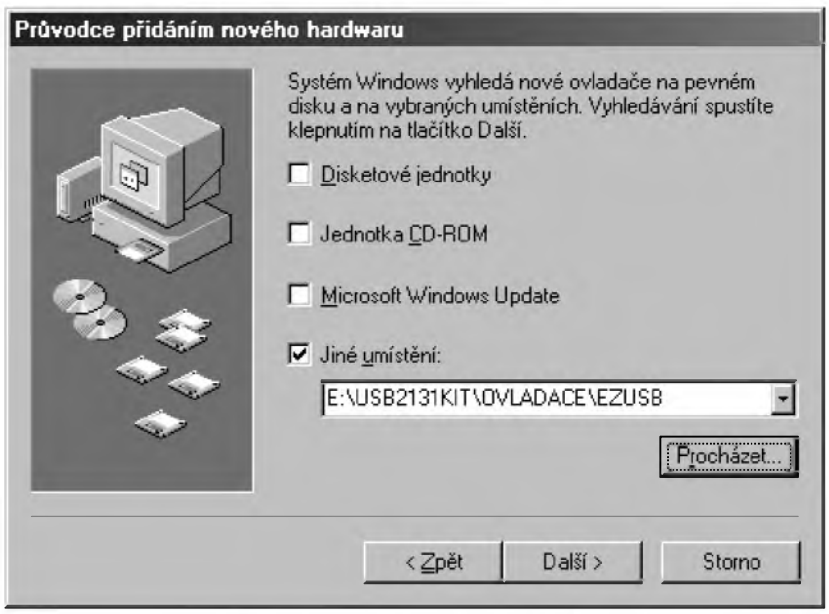
• nakonec handle zařízení odevzdáme systému funkcí **CloseHandle**.

Získání handle zařízení - CreateFile

Funkce **CreateFile** se v souvislosti s ovládáním hardware používá velmi

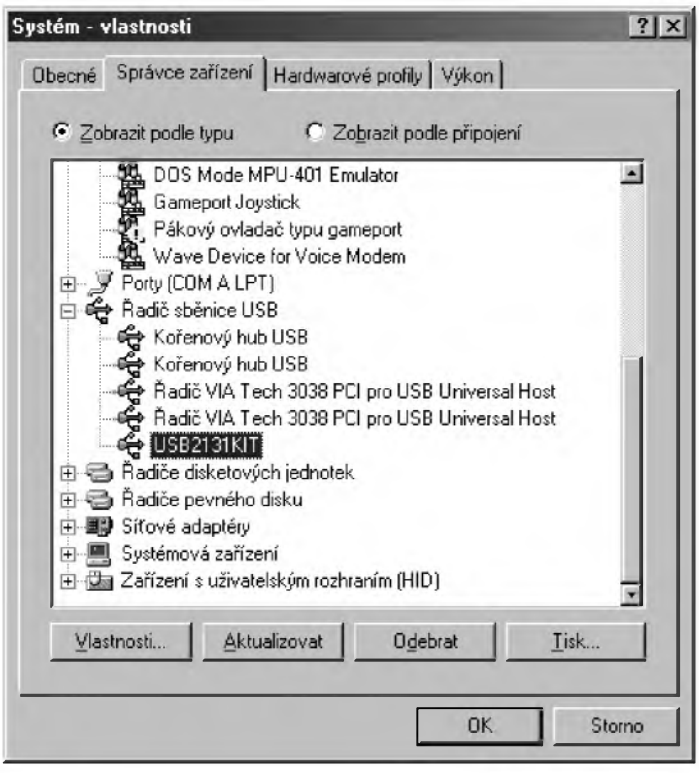
Obr. 6.1. Úvodní dialog instalace

Obr. 6.2. Uživatel vybere ovladač sám



Obr. 6.3. Volba adresáře s ovladačem

← *Obr. 6.4. Systém úspěšně nalezl správný ovladač*



↑ *Obr. 6.5. Instalace úspěšně dokončena*

← *Obr. 6.6. Dialog systém ukazuje zařízení **USB2131KIT***

často (je to zvyklost systému Windows, že např. sériové a paralení porty mapuje jako soubory).

Deklarace funkce z hlediska jazyka C:

HANDLE **CreateFile**(
LPCTSTR **lpFileName**,
DWORD **dwDesiredAccess**,
DWORD **dwShareMode**,
LPSECURITY_ATTRIBUTES **lpSecurityAttributes**,
DWORD **dwCreationDistribution**,
DWORD **dwFlagsAndAttributes**,
HANDLE **hTemplateFile**);

Parametry funkce **CreateFile** si vyložíme velmi stručně, v míře postačující pro naši potřebu:

- **lpFileName** - jméno otevíraného zařízení (souboru); ovladač **EZUSB.SYS** označuje první zařízení s mikrořadičem AN2131 jako **\\.\ezusb-0**, druhé zařízení má pak označení **\\.\ezusb-1** atd. Připomeňme, symbol **** musí být v zápisu řetězce zdvojen (platí pouze pro jazyk C),
- **dwDesiredAccess** - bitové pole definující požadovaný přístup k zařízení, pro plný přístup zadáme složení příznaků **GENERIC_READ|GENERIC_WRITE** (symbol **|** označuje v jazyce C bitový součet),
- **dwShareMode** - bitové pole definující režim sdílení; pokud zařízení nechceme sdílet se zbytkem systému (žádáme exkluzivní přístup) zapíšeme **0** (žádné sdílení),
- **lpSecurityAttributes** - ukazatel na strukturu bezpečnostních atributů (nelze používat pod Windows 98), většinou používáme hodnotu **NULL**, která toto zabezpečení ignoruje,
- **CreationDistribution** - způsob vytvoření (má smysl hlavně pro soubory), v případě otevírání zařízení zadáme **OPEN_EXISTING** (zajistí otevření existujícího zařízení, pokud je k dispozici),
- **FlagsAndAttributes** - atributy (souboru) a řídicí příznaky; pro zařízení nelze používat a proto zadáváme **0**,
- **hTemplateFile** - handle tzv. souborové šablony, podle které se soubor vytvoří; pro zařízení nemá smysl a proto zadáváme **NULL**,
- **návratová hodnota** - představuje handle zařízení, pokud zařízení není k dispozici (tedy nelze jej otevřít) je vrácena speciální hodnota **INVALID_HANDLE_VALUE** (testováním této hodnoty tedy zjistíme úspěšné otevření zařízení).

Zavření zařízení - CloseHandle

Funkce **CloseHandle** zavře zařízení, parametrem je handle zařízení získaný při volání funkce **CreateFile**.

Deklarace funkce z hlediska jazyka C:

BOOL **CloseHandle**(
HANDLE **hObject**);

Volání služeb ovladače

Funkce **DeviceIoControl** umožňuje volat služby ovladače přes handle získaného zařízení (viz **CreateFile**).

Deklarace funkce z hlediska jazyka C:

BOOL **DeviceIoControl**(
HANDLE **hDevice**,
DWORD **dwIoControlCode**,
LPVOID **lpInBuffer**,
DWORD **nInBufferSize**,
LPVOID **lpOutBuffer**,
DWORD **nOutBufferSize**,
LPDWORD **lpBytesReturned**,
LPOVERLAPPED **lpOverlapped**);

Parametry funkce **DeviceIoControl**:

- **hDevice** - handle zařízení získaný voláním funkce **CreateFile**,
- **dwIoControlCode** - řídicí kód operace získaný makrem **CTL_CODE** (viz dále),
- **lpInBuffer** - adresa bufferu obsahující data nutná pro provedení vstupní operace (například podrobnější stanovení příkazu, který má být proveden); pokud není potřeba, zadáme hodnotu **NULL**,
- **nInBufferSize** - velikost vstupního bufferu, který byl určen parametrem **lpInBuffer**,
- **lpOutBuffer** - adresa bufferu pro příjem výstupních dat (například pro načtení dat získaných ze zařízení); pokud není potřeba, zadáme hodnotu **NULL**,
- **nOutBufferSize** - velikost výstupního bufferu, který byl určen parametrem **lpOutBuffer**,
- **lpBytesReturned** - ukazatel na proměnnou typu **DWORD**, která indikuje počet bajtů skutečně přečtených ze zařízení (tak lze indikovat, zda se přečetlo **nOutBufferSize** bajtů či méně),
- **lpOverlapped** - ukazatel na strukturu **OVERLAPPED**, která se používá pro překryvné (asynchronní) operace; nebudeme používat, proto zadáme **NULL**,
- **návratová hodnota** - při úspěchu vrací hodnotu různou od **0**, v případě chyby vrací **0**.

Popis klíčových služeb ovladače EZUSB.SYS

Na tomto místě si popíšeme několik služeb ovladače **EZUSB.SYS**, které budeme používat:

- **GetDeviceDescriptor** - tato služba získá deskriptor zařízení; má v ovladači číslo 0x0801, jedná se o bufferovanou funkci; nemá vstupní parametry; na výstupu vrací 18 bajtů deskriptorů zařízení (viz tab. 6.3),
- **UpLoad** - tato služba čte jeden nebo několik bajtů (maximálně 1024) z vnější datové RAM; má v ovladači číslo 0x0805, jedná se o bufferovanou funkci; na vstupu je třeba předat 9bajtovou strukturu popisující čtenou část paměti (viz tab. 6.4); na výstupu vrací hodnoty přečtených bajtů,

Tab. 6.3. Struktura popisující zařízení (výchozí hodnoty)

Typ	Označení	Význam	Hodnota
BYTE	Length	délka deskriptoru	18
BYTE	DescriptorType	typ deskriptoru	1
WORD	bcdUSB	verze USB specifikace v BCD	0x0101
BYTE	DeviceClass	třída zařízení	0xFF
BYTE	SubDeviceClass	podtřída zařízení	0xFF
BYTE	DeviceProtocol	protokol zařízení	0xFF
BYTE	MaxPacketSize0	maximální velikost paketu pro endpoint 0	64
WORD	VID	Vendor ID	0x0547
WORD	PID	Product ID	0x2131
WORD	bcdDevice	číslo verze zařízení v BCD	????
BYTE	Manufacturer	index řetězce výrobce	0
BYTE	Product	index řetězce výrobku	0
BYTE	SerialNumber	index řetězce sériového čísla	0
BYTE	NumConfigurations	počet konfigurací dostupných pro dané rozhraní	1

Tab. 6.4. Formát vstupních parametrů pro službu UpLoad

Typ	Označení	Význam	Hodnota
BYTE	Request	požadavek (pro UpLoad)	0xA0
WORD	Value	počáteční adresa	Start
WORD	Index	nepoužito	0
WORD	Length	počet čtených bajtů	Délka
BYTE	Direction	čtení (1)	1
BYTE	Data	nepoužito	0

- **DownLoad** - tato služba zapisuje jeden nebo několik bajtů do vnější datové RAM; má v ovladači číslo 0x081B, jedná se o funkci předávající parametry nepřímo; na vstupu je třeba předat ukazatel na startovací adresu (v rozměru slova); funkce nemá výstup a na místo výstupních parametrů dosazujeme adresu pole zapisovaných bajtů.

Získání řídicího kódu služby

Pro volání funkce **DeviceIoControl** je nutno místo čísla služby znát tzv. řídicí kód. Ten je složením 4 parametrů:

- **DeviceType** - typ zařízení, některá zařízení jsou v systému předdefinována (např. FILE_DEVICE_BEEP odpovídá vestavěnému reproduktoru); pro případ ovladače EZUSB.SYS zadáme **FILE_DEVICE_UNKNOWN**,
- **Function** - číslo funkce, viz výše,
- **Method** - typ metody (bufferovaná - data se předávají přes buffer; nepřímá - data se předávají přes ukazatel),
- **Access** - přístup; obvykle **FILE_ANY_ACCESS** (libovolný přístup).

Pro sestavení kódu služby z výše uvedených parametrů slouží makro **CTL_CODE**, které je definováno v hlavičkovém souboru **WINIOCTL.H**.

#define **CTL_CODE**(**DeviceType**, **Function**, **Method**, **Access**) (\
((DeviceType) << 16) | ((Access) << 14) | ((Function) << 2) | (Method))

7. Ovládací jednotka EZUSB

V této kapitole je předvedena implementace ovládací jendotky **EZUSB**, která nám zajistí pohodlné ovládání přípravku **USB2131KIT**.

Jednotka je vytvořena ve vývojovém prostředí **C++ Builder** (programování v C++).

Univerzální ovládací rozhraní pro C++ Builder

Univerzální ovládací rozhraní nám zjednoduší vyvolávání jednotlivých služeb ovladače. Na zařízení budeme nahlížet jako na instanci třídy **TAN2131** a příslušné akce provedeme pouze voláním metod nebo použitím vlastností.

Hlavičkový soubor (EZUSB.H):

```
#ifndef EZUSBH
#define EZUSBH
//---------------------------
#include <winioctl.h>
//---------------------------
#pragma stack(push)
#pragma stack(1)
//deskriptor zařízení:
struct TDeviceDescriptor{
  BYTE Length;
  BYTE DescriptorType;
  WORD bcdUSB;
```

```
  BYTE DeviceClass;
  BYTE SubDeviceClass;
  BYTE DeviceProtocol;
  BYTE MaxPacketSize0;
  WORD VID;
  WORD PID;
  WORD bcdDevice;
  BYTE Manufacturer;
  BYTE Product;
  BYTE SerialNumber;
  BYTE NumConfigurations;
};
//CTL kód pro čtení deskriptoru:
const DWORD
  IOCTL_EZUSB_GET_DEVICE_DESCRIPTOR=
  CTL_CODE(
    FILE_DEVICE_UNKNOWN,
    0x0801,
    METHOD_BUFFERED,
    FILE_ANY_ACCESS);
//---------------------------------------
//vstupní dotaz:
struct TVendorRequestIn{
  BYTE Request;
  WORD Value;
  WORD Index;
  WORD Length;
  BYTE Direction;
  BYTE Data;
};
//CTL kód pro vstupní dotaz:
const DWORD IOCTL_EZUSB_UPLOAD=
  CTL_CODE(
    FILE_DEVICE_UNKNOWN,
    0x0805,
    METHOD_BUFFERED,
    FILE_ANY_ACCESS);
//---------------------------------------
//CTL kód pro RAM download indirect:
const DWORD IOCTL_EZUSB_DOWNLOAD=
  CTL_CODE(
    FILE_DEVICE_UNKNOWN,
    0x081B,
    METHOD_IN_DIRECT,
    FILE_ANY_ACCESS);
#pragma stack(pop)
//---------------------------------------
//USB registry:
const WORD CPUCS=0x7F92;
const WORD PORTACFG=0x7F93;
const WORD PORTBCFG=0x7F94;
const WORD PORTCCFG=0x7F95;
const WORD OUTA=0x7F96;
const WORD OUTB=0x7F97;
const WORD OUTC=0x7F98;
const WORD PINSA=0x7F99;
const WORD PINSB=0x7F9A;
const WORD PINSC=0x7F9B;
const WORD OEA=0x7F9C;
const WORD OEB=0x7F9D;
const WORD OEC=0x7F9E;
const WORD I2CS=0x7FA5;
const WORD I2DAT=0x7FA6;
//---------------------------------------
//pomocné funkce:
AnsiString __fastcall WordToHex(
  WORD Value);
AnsiString __fastcall ByteToHex(
  BYTE Value);
AnsiString __fastcall ByteToBin(
  BYTE Value);
AnsiString __fastcall BufferToHex(
  BYTE *Buffer, WORD Length);
BYTE __fastcall BinToByte(
  bool B7, bool B6, bool B5, bool B4,
  bool B3, bool B2, bool B1 ,bool B0);
bool __fastcall BitDecode(
  BYTE Value, int Index);
//---------------------------------------
//třída pro obsluhu AN2131:
class TAN2131{
private:
  DWORD d;
  BOOL b;
  HANDLE Handle;
  TDeviceDescriptor FDeviceDescriptor;
  TVendorRequestIn Request;
  BYTE __fastcall GetRAM(WORD Address);
  void __fastcall SetRAM(
    WORD Address,BYTE Data);
  void __fastcall Init(
    AnsiString DeviceName);
public:
  __fastcall TAN2131(AnsiString
    DeviceName="\\\\.\\ezusb-0");
  __fastcall TAN2131(int DeviceNumber);
  __fastcall ~TAN2131();
  __property TDeviceDescriptor
    DeviceDescriptor=
    {read=FDeviceDescriptor};
  bool __fastcall UpLoad(
    WORD StartAddress,
    WORD Length, BYTE *Data);
  bool __fastcall DownLoad(
    WORD StartAddress,
    WORD Length, BYTE *Data);
  int __fastcall DownLoadFromRes(
    WORD StartAddress,
    int Instance,
    const AnsiString ResName,
    char *ResType);
  __property BYTE RAM[WORD Address]=
    {read=GetRAM,write=SetRAM};
};
//---------------------------------------
#endif
Zdrojový kód (EZUSB.CPP):
#include <vcl.h>
#pragma hdrstop
#include "EZUSB.h"
//---------------------------------------
#pragma package(smart_init)
//---------------------------------------
//konstruktor s implicitním parametrem,
//zadává se jméno zařízení:
__fastcall TAN2131::TAN2131(
  AnsiString DeviceName)
{
  Init(DeviceName);
}
//---------------------------------------
//konstruktor s povinným parametrem,
//zadává se pořadové číslo zařízení:
__fastcall TAN2131::TAN2131(
  int DeviceNumber)
{
  Init(AnsiString("\\\\.\\ezusb-")
    +AnsiString(DeviceNumber));
}
//---------------------------------------
//provádí kódy obou konstruktorů:
void __fastcall TAN2131::Init(
  AnsiString DeviceName)
{
  //otevře zařízení přes ovladač:
  Handle=CreateFile(
    DeviceName.c_str(),
    GENERIC_READ|GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);
  //test úspěšnosti otevření:
  if(Handle==INVALID_HANDLE_VALUE)
    throw Exception("Zařízení "
      +DeviceName+" nelze otevřít");
  //zjistí deskriptor zařízení:
  DeviceIoControl(
    Handle,
    IOCTL_EZUSB_GET_DEVICE_DESCRIPTOR,
    NULL,
    0,
    &FDeviceDescriptor,
    18,
    &d,
    NULL);
}
//---------------------------------------
//destruktor:
__fastcall TAN2131::~TAN2131()
{
  //odevzdá handle zařízení systému:
  CloseHandle(Handle);
}
//---------------------------------------
//čte Length bajtů do Data
//počínaje StartAddress:
bool __fastcall TAN2131::UpLoad(
  WORD StartAddress, WORD Length, BYTE *Data)
{
  Request.Request=0xA0;
  //počáteční adresa:
  Request.Value=StartAddress;
  Request.Index=0;
  //počet čtených bajtů (max. 1024):
  Request.Length=Length;
  //čtení:
  Request.Direction=1;
  Request.Data=0x00;
  //vlastní čtení z RAM:
  b=DeviceIoControl(
    Handle,
    IOCTL_EZUSB_UPLOAD,
    &Request,
    sizeof(Request),
    Data,
    Length,
    &d,
    NULL);
  //má být přečteno Length bajtů:
  return (d==(DWORD)Length)&&(b);
}
//---------------------------------------
//zapíše Length bajtů z Data
//počínaje StartAddress:
bool __fastcall TAN2131::DownLoad(
  WORD StartAddress, WORD Length, BYTE *Data)
{
  //zápis do RAM:
  b=DeviceIoControl(
    Handle,
    IOCTL_EZUSB_DOWNLOAD,
    &StartAddress,
    sizeof(StartAddress),
    Data,
    Length,
    &d,
    NULL);
  //test úspěšnosti:
  return b;
}
//---------------------------------------
//zapíše obsah zdroje
//počínaje StartAddress do RAM,
//vrací délku zdroje:
```

```
int __fastcall TAN2131::DownLoadFromRes(
  WORD StartAddress,int Instance,
  const AnsiString ResName,char *ResType)
{
  TResourceStream *rs=NULL;
  BYTE *Data=NULL;
  BYTE *p;
  int v=-1;
  try{
    //otevře stream:
    rs=new TResourceStream(
      Instance,ResName,ResType);
    //zjistí délku dat:
    v=rs->Size;
    //nastaví velikosti pole Data:
    Data=new BYTE[v];
    //přetypuje ukazatel na typ *BYTE:
    p=(BYTE*)rs->Memory;
    //kopie dat z rs do Data:
    for(int i=0;i<v;i++)
      //kopie:
      Data[i]=*p++;
    //vlastní zápis:
    DownLoad(StartAddress,v,Data);
  }
  catch(Exception &E){
    //uvolní paměť při výjimce:
    if(Data)
      delete[] Data;
    if(rs)
      delete rs;
    //výjimku posílá dál:
    throw Exception(E);
  }
  //uvolní paměť mimo výjimku:
  if(Data)
    delete[] Data;
  if(rs)
    delete rs;
  //vrátí velikost zdroje:
  return v;
}
//---------------------------------------
//čtecí metoda pro RAM:
BYTE __fastcall TAN2131::GetRAM(
  WORD Address)
{
  BYTE Data;
  UpLoad(Address,1,&Data);
  return Data;
}
//---------------------------------------
//zapisovací metoda pro RAM:
void __fastcall TAN2131::SetRAM(
  WORD Address,BYTE Data)
{
  DownLoad(Address,1,&Data);
}
//---------------------------------------
//převod slova Value na řetězec
//hexačíslic:
AnsiString __fastcall WordToHex(
  WORD Value)
{
  return "0x"+IntToHex(Value,4);
}
//---------------------------------------
//převod bajtu Value na řetězec
//hexačíslic:
AnsiString __fastcall ByteToHex(
  BYTE Value)
{
  return "0x"+IntToHex(Value,2);
}
//---------------------------------------
//převod bajtu Value na řetězec
//binárních číslic:
AnsiString __fastcall ByteToBin(
  BYTE Value)
{
  AnsiString r;
  //vymaskování bitů:
  for(int i=7;i>-1;i--)
    r+=(Value>>i)&0x01;
  //výsledek:
  return r;
}
//---------------------------------------
//převod bufferu Buffer na řetězec
//hexačíslic:
AnsiString __fastcall BufferToHex(
  BYTE *Buffer,WORD Length)
{
  AnsiString r;
  //převod bajtů:
  for(WORD i=0;i<Length;i++){
    r+=ByteToHex(Buffer[i]);
    if(i<Length-1)
      r+=" ";
  }
  //výsledek:
  return r;
}
//---------------------------------------
//převod bitů na bajt:
BYTE __fastcall BinToByte(
  bool B7,bool B6,bool B5,bool B4,
  bool B3,bool B2,bool B1,bool B0)
{
  return (B7<<7)|(B6<<6)|(B5<<5)|(B4<<4)
    |(B3<<3)|(B2<<2)|(B1<<1)|B0;
}
//---------------------------------------
//vrátí hodnotu bitu Index z bajtu Value:
bool __fastcall BitDecode(
  BYTE Value,int Index)
{
  return Value&(0x01<<Index);
}
```

Konstruktory a destruktor:

• **__fastcall TAN2131(AnsiString DeviceName="\\\\.\\ezusb-0")** - konstruktor s implicitním parametrem (pokud parametr **DeviceName** nevyplníme, otevře se zařízení s označením \\.\ezusb-0), označení zařízení je dáno parametrem **DeviceName** (první je \\.\ezusb-0, následuje \\.\ezusb-1 atd.),

• **__fastcall TAN2131(int DeviceNumber)** - konstruktor pro otevření zařízení na základě pořadového čísla, např. **DeviceNumber = 0** otevře zařízení \\.\ezusb-0,

• **__fastcall ~TAN2131( )** - destruktor, zavře zařízení.

Metody:

• **bool __fastcall UpLoad(WORD StartAddress, WORD Length, BYTE *Data)** - načte z vnější RAM tolik bajtů, kolik je určeno parametrem **Length**. Starovací adresa je dána parametrem **StartAddress** (např. StartAddress = 0, Length = 10 načte prvních deset bajtů vnější RAM). Data jsou uložena do bufferu, jehož adresa se předává v parametru **Data** (ukazatel na pole bajtů, pole musí mít velikost minimálně Length bajtů),

• **bool __fastcall DownLoad(WORD StartAddress, WORD Length, BYTE *Data)** - zapíše do vnější RAM tolik bajtů, kolik je určeno parametrem **Length**. Startovací adresa je určena parametrem **StartAddress**. Data se berou z bufferu, na který ukazuje **Data** (pole bajtů),

• **int __fastcall DownLoadFromRes (WORD StartAddress, int Instance, const AnsiString ResName, char *Res Type)** - zapíše do RAM obsah zdroje programu (resource) od adresy **Start Address**; výskyt, název a typ zdroje je určen parametry **Instance**, **ResName** a **ResType**. Funkce vrací velikost zdroje, který byl zapisován (hodí se pro informaci o volném místě za koncem zapsaného bloku).

Vlastnosti:

• **TDeviceDescriptor DeviceDescriptor** (R/O) - vrací deskriptor zařízení podle tab. 6.3, nejdůležitější jsou patrně položky **VID** a **PID**,

• **BYTE RAM**[WORD Address] - umožňuje číst nebo zapisovat do zvolené buňky vnější RAM (narozdíl od metod **UpLoad** a **DownLoad** lze ovládat pouze jeden bajt). Deklarace v podobě vlastnosti typu pole dovoluje paměť indexovat podobně jako prosté pole bajtů. Pro přístup k základním USB registrům mikrořadiče AN2131 jsou zavedeny konstanty: **CPUCS**, **PORTACFG**, **PORTBCFG**, **PORTCCFG**, **OUTA**, **OUTB**, **OUTC**, **PINSA**, **PINSB**, **PINSC**, **OEA**, **OEB**, **OEC**, **I2CS**, **I2DAT** (viz také tab. 3.2).

Pomocné funkce:

• **AnsiString __fastcall WordToHex (WORD Value)** - konvertuje číselný údaj uložený v parametru **Value** (v rozměru slova) na řetězec hexadecimálních číslic; např. pro Value = 0x2131 dostaneme řetězec "0x2131",

• **AnsiString __fastcall ByteToHex (BYTE Value)** - konvertuje číselný údaj uložený v parametru **Value** (v rozměru bajtu) na řetězec hexadecimálních číslic; např. pro Value = 0xFF dostaneme řetězec "0xFF",

• **AnsiString __fastcall BufferToHex (BYTE *Buffer, WORD Length)** - konvertuje pole bajtů (Buffer je adresa pole, Length je jeho délka) na řetězec hexadecimálních číslic (podobně jako ByteTo Hex), každý bajt je oddělen mezerou,

• **AnsiString __fastcall ByteToBin (BYTE Value)** - konvertuje číselný údaj uložený v parametru **Value** (v rozměru bajtu) na řetězec binárních (dvojkových) číslic; například pro Value = 0xA5 dostaneme řetězec "10100101",

• **BYTE __fastcall BinToByte**(bool B7, bool B6, bool B5, bool B4, bool B3, bool B2, bool B1, bool B0) - složí 8 bitů **B0 až B7** (B0 má nejnižší váhu) do podoby bajtu; např. BinToByte(1, 0, 1, 0, 0, 1, 0, 1) vrátí 0xA5,

• **bool __fastcall BitDecode(BYTE Value, int Index)** - vrátí hodnotu určeného bitu (**Index** = 0 až 7, 0 značí bit s nejnižší váhou) bajtu **Value**; např. pro Value = 0xA5 a Index = 2 dostaneme 1 (protože to je hodnota bitu 2, 0xA5 = 10100101b).

8. Příklad č. 1
- snímání stavu portů - použití přípravku ATDIPSW2

Na tomto prvním příkladu je vysvětleno, jak používat třídu **TAN2131** a pomocné funkce z jednotky **EZUSB**.

Jak je uvedeno v nadpisu, zajišťuje program snímání stavu registrů **PINSA až PINSC**, tedy vstupní stav vývodů portů PA až PC. Čtení je prováděno periodicky (pomocí časovače) s periodou 100 ms. Pro tuto operaci není nutné, aby byl do mikrořadiče AN2131 zaveden nějaký program.

Návrhový formulář (obr. 8.1) byl vytvořen tak, že na plochu byly umístěny 3 komponenty typu GroupBox pod názvy: **gbPINSA**, **gbPINSB** a **gbPINSC** (každá odpovídá jednomu portu PINSA až PINSC). Do těchto komponent bylo umístěno po 8 komponentách typu CheckBox (např. pro skupinu **gbPINSA** jsou jména **cbPINSA7** až **cbPINSA0**). Dále jsou umístěny pomocné komponenty typu Label s popisky (MSB značí nejvýznamější bit, LSB pak nejméně významný bit).

Další komponentou je **Timer**, nastavení: Name = **Casovac**, Interval = = 100, Enabled = false.

Poslední komponentou je **ApplicationEvents** (z karty Additional), Name = **AplUdalosti**.

Dále byly vygenerovány události: **OnCreate** formuláře (**FormCreate**), **OnTimer** časovače (**Aktualizuj**), **OnDestroy** formuláře (**FormDestroy**) a **OnException** komponenty **AplUdalosti** (**Nezachyceno**).

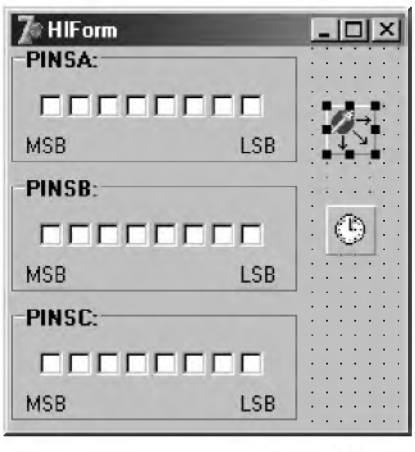
Inicializace

Pro řízení přípravku je nutno vytvořit instanci třídy **TAN2131**. Za tímto účelem je do privátní sekce deklarace třídy **THlForm** (představuje formulář) zapsána deklarace proměnné **AN2131**:

```
TAN2131 *AN2131;
```

Vlastní vytvoření instance probíhá v události **FormCreate** (OnCreate formuláře). Je využit implicitní konstruktor, takže v závorce není uvedena žádná inicializační hodnota (otevře se zařízení \\.\ezusb-0):

```
AN2131=new TAN2131;
```



Obr. 8.1. Návrhový formulář

Následně je do proměnné **dd** (typu **TDeviceDescriptor**) přečten deskriptor zařízení a z něj získané hodnoty VID a PID zobrazeny v titulku okna:

```
dd=AN2131->DeviceDescriptor   ;
Caption="VID="+WordToHex(dd.VID)
      +", PID="+WordToHex(dd.PID);
```

Nakonec je aktivován časovač. Časovač není záměrně aktivován již při návrhu formuláře, protože by mohly vzniknout problémy při rozběhu aplikace. Pokud totiž nebude zařízení k dispozici, nebude instance **AN2131** zkonstruována korektně. Obsluha časovače však instanci **AN2131** bere jako správně inicializovanou:

```
Casovac->Enabled=true;
```

Čtení stavu portů

Jak bylo již uvedeno, stav portů **PINSA** až **PINSC** je čten v události OnTimer časovače, která je nazvána **Aktualizuj**. Pro každý port je definována proměnná (symboly PINSA až PINSC jsou definovány v jednotce EZUSB):

```
BYTE vPINSA,vPINSB,vPINSC;
```

Čtení je prováděno vlastností **TAN2131::RAM**, např.:

```
vPINSA=AN2131->RAM[PINSA];
```

Nakonec je třeba zobrazit stav jednotlivých bitů v komponentách typu checkbox. Například pro nejvyšší bit portu **PINSA** je použit tento zápis (funkce **BitDecode** vybere z proměnné **vPINSA** hodnotu nejvyššího bitu, tomu odpovídá číslo 7):

```
cbPINSA7->Checked=BitDecode(vPINSA,7);
```

Uvolnění instance

Posledním krokem je destrukce instance **AN2131** na konci běhu aplikace. Je to provedeno pomocí události **FormDestroy** (OnDestroy formuláře):

```
delete AN2131;
```

Reakce na výjimky

Třída **TAN2131** je sestavena tak, že detekuje možné chyby komunikace s mikrořadičem AN2131. Pokud tedy nastane nesprávná odezva ze strany hardware, je generována výjimka **TAN2131Exception**. Jeden z možných způsobů ošetření výjimečného stavu byl proveden v události **FormCreate** (viz výše). Podobné ošetření by mělo být uvedeno i v události **Aktualizuj**. Jinak bude dialog s popisem výjimky neustále vypisován (například při odpojení přípravku za běhu programu). Pro ukázku jiných možností programování byla pro účel detekce výjimky použita aplikační událost typu **OnException** (reaguje na nezachycené výjimky) nazvaná **Nezachyceno**. Její kód zajišťuje zastavení časovače, zobrazení textu výjimky a předčasné ukončení aplikace:

```
Casovac->Enabled=false;
Application->MessageBox(
  E->Message.c_str(),
  "EZUSB",
  MB_ICONHAND);
Application->Terminate();
```

Výpis celého programu s komentáři:

VSTFORM.H:

```
#ifndef VSTFORMH
#define VSTFORMH

#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>

#include "EZUSB.h"
#include <ExtCtrls.hpp>
#include <AppEvnts.hpp>

//deklarace formuláře:
class THlForm : public TForm
{
__published:
  TGroupBox *gbPINSA;
  TCheckBox *cbPINSA7;
  TCheckBox *cbPINSA6;
  TCheckBox *cbPINSA5;
  TCheckBox *cbPINSA4;
  TCheckBox *cbPINSA3;
  TCheckBox *cbPINSA2;
  TCheckBox *cbPINSA1;
  TCheckBox *cbPINSA0;
  TGroupBox *gbPINSB;
  TCheckBox *cbPINSB7;
  TCheckBox *cbPINSB6;
  TCheckBox *cbPINSB5;
  TCheckBox *cbPINSB4;
  TCheckBox *cbPINSB3;
  TCheckBox *cbPINSB2;
  TCheckBox *cbPINSB1;
  TCheckBox *cbPINSB0;
  TGroupBox *gbPINSC;
  TCheckBox *cbPINSC7;
  TCheckBox *cbPINSC6;
  TCheckBox *cbPINSC5;
  TCheckBox *cbPINSC4;
  TCheckBox *cbPINSC3;
  TCheckBox *cbPINSC2;
  TCheckBox *cbPINSC1;
  TCheckBox *cbPINSC0;
  TLabel *Label1;
  TLabel *Label2;
  TLabel *Label3;
  TLabel *Label4;
  TLabel *Label5;
  TLabel *Label6;
  TTimer *Casovac;
  TApplicationEvents *AplUdalosti;
  void __fastcall AplUdalostiException(
    TObject *Sender,Exception *E);
  void __fastcall Aktualizuj(
    TObject *Sender);
  void __fastcall FormCreate(
    TObject *Sender);
  void __fastcall FormDestroy(
    TObject *Sender);
```

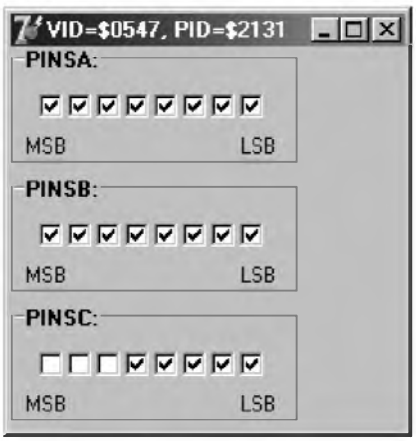
```
private:
    //proměnná pro přístup k AN2131:
    TAN2131 *AN2131;
public:         // User declarations
  __fastcall THlForm(TComponent* Owner);
};
extern PACKAGE THlForm *HlForm;
#endif
VSTFORM.CPP:
//---------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "VstForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------
__fastcall THlForm::THlForm(
  TComponent* Owner) : TForm(Owner)
{

}
//---------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
{
  TDeviceDescriptor dd;

  try{
    //vytvoření instance TAN2131:
    AN2131=new TAN2131;
    //čtení deskriptoru:
    dd=AN2131->DeviceDescriptor;
    //zobrazení VID a PID:
    Caption="VID="+WordToHex(dd.VID)
          +", PID="+WordToHex(dd.PID);
    //aktivace časovače:
    Casovac->Enabled=true;
  }
  catch(Exception &E){
    Application->MessageBox(
      E.Message.c_str(),
      "EZUSB",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
}
//---------------------------------------
void __fastcall THlForm::FormDestroy(
  TObject *Sender)
{
  if(AN2131)
    delete AN2131;
}
//---------------------------------------
void __fastcall THlForm::Aktualizuj(
  TObject *Sender)
{
  BYTE vPINSA,vPINSB,vPINSC;
  //čtení stavu portu PA:
  vPINSA=AN2131->RAM[PINSA];
  cbPINSA7->Checked=BitDecode(vPINSA,7);
  cbPINSA6->Checked=BitDecode(vPINSA,6);
  cbPINSA5->Checked=BitDecode(vPINSA,5);
  cbPINSA4->Checked=BitDecode(vPINSA,4);
  cbPINSA3->Checked=BitDecode(vPINSA,3);
  cbPINSA2->Checked=BitDecode(vPINSA,2);
  cbPINSA1->Checked=BitDecode(vPINSA,1);
  cbPINSA0->Checked=BitDecode(vPINSA,0);
  //čtení stavu portu PB:
  vPINSB=AN2131->RAM[PINSB];
  cbPINSB7->Checked=BitDecode(vPINSB,7);
  cbPINSB6->Checked=BitDecode(vPINSB,6);
  cbPINSB5->Checked=BitDecode(vPINSB,5);
  cbPINSB4->Checked=BitDecode(vPINSB,4);
  cbPINSB3->Checked=BitDecode(vPINSB,3);
  cbPINSB2->Checked=BitDecode(vPINSB,2);
  cbPINSB1->Checked=BitDecode(vPINSB,1);
  cbPINSB0->Checked=BitDecode(vPINSB,0);
  //čtení stavu portu PC:
  vPINSC=AN2131->RAM[PINSC];
  cbPINSC7->Checked=BitDecode(vPINSC,7);
  cbPINSC6->Checked=BitDecode(vPINSC,6);
  cbPINSC5->Checked=BitDecode(vPINSC,5);
  cbPINSC4->Checked=BitDecode(vPINSC,4);
  cbPINSC3->Checked=BitDecode(vPINSC,3);
  cbPINSC2->Checked=BitDecode(vPINSC,2);
  cbPINSC1->Checked=BitDecode(vPINSC,1);
  cbPINSC0->Checked=BitDecode(vPINSC,0);
}
//---------------------------------------
void __fastcall THlForm::ApiUdalostiException(
  TObject *Sender, Exception *E)
{
  Casovac->Enabled=false;
  Application->MessageBox(
    E->Message.c_str(),
    "EZUSB",
    MB_ICONHAND);
  //předčasné ukončení aplikace:
  Application->Terminate();
}
```

Vlastní test aplikace a přípravků

Před spuštěním aplikace připojte přípravek **USB2131KIT** na libovolný port USB (měl by být instalován ovladač, viz kap. 6). Přípravek **ATDIPSW2** pak pomocí kablíku připojte na jeden z konektorů **PA** až **PC**.

Po spuštění aplikace (je-li přípravek USB2131KIT funkční) lze sledovat, jak program reaguje na změnu stavu spínačů připojených na zvolený port. Pozor, rozpojený spínač odpovídá stavu „log. 1" (políčko zaškrtnuto) a sepnutý spínač pak log. 0 (políčko nezaškrtnuto). Na obr. 8.2 je byl přípravek ATDIPSW2 připojen na konektor PC (spínače připojené na tři nejvyšší bity byly sepnuty, ostatní byly rozpojeny).

Pokud při spuštění aplikace nebude přípravek **USB2131KIT** připojen (nebo nebude správně fungovat), zobrazí se dialog podle obr. 8.3 a aplikace bude předčasně ukončena. Podobný dialog (s textem: chyba komunikace) se zobrazí, pokud přípravek odpojíte v době běhu programu **VSTUPY.EXE**.



Obr. 8.2. Aplikace VSTUPY.EXE v akci

Obr. 8.3. Přípravek není připojen při spuštění aplikace VSTUPY.EXE

9. Příklad č. 2

- porty ve výstupním režimu - použití přípravku AT8LED

V tomto příkladu si již ukážeme, jak naprogramovat mikrořadič **AN2131Q** přímo v aplikaci (mikrořadič zůstane zapájený ve vývojovém kitu, programování proběhne přes USB). Dále si ukážeme, jak porty mikrořadiče používat ve výstupním směru.

Základní informace (idea realizace)

V předchozí kapitole (příklad č. 1) byla situace velmi jednoduchá. Program snímal stav vývodů portů (tedy sledoval vstupy) pouhým čtením registrů **PINSA až PINSC**. A daný stav dále zobrazoval pomocí políček.

Nebyl tedy nutný žádný program vložený do mikrořadiče. Příkaz **Firmware Upload** totiž poskytuje možnost číst libovolnou oblast vnější RAM bez ohledu na právě prováděný program. Protože jsou registry **PINSA až PINSC** mapovány právě do vnější RAM, je tato operace velmi jednoduchá.

Pokud budeme chtít použít porty jako výstupy, může se zdát, že program zrealizujeme podobně. Bohužel registry **OUTA až OUTC** (určují výstupní úroveň vývodů portů) a **OEA až OEC** (volí mezi vstupním a výstupním režimem vývodů) nelze nastavovat příkazem **Firmware Download**. Tyto porty mohou být ovládány výhradně samotným mikrořadičem.

Jedinou (jednoduchou) možností realizovat tento úkol je tedy uložit výstupní data pro porty do té části vnější RAM, kam je možno zapsat příkazem **Firmware Download**. Vložený program (do mikrořadiče jej nahrajeme opět příkazem Firmware Download) pak tyto hodnoty uloží do registrů **OUTA až OUTC** a tím vlastně zajistí námi požadované vybavení vývodů mikrořadiče.

Překlad zdrojového kódu

Pro vývoj programů pro mikrořadiče kompatibilní s 8051 se používá mnoho vývojových prostředí. Na tomto místě připomeneme freeware utilitky **ASM51** a **HEX2BIN**.

ASM51.EXE je překladačem zdrojových souborů v asembleru 8051 do formátu Intel Hex. Program se vyvolává z příkazové řádky. Pro správnou funkci

je nutno zadat název souboru ve formátu 8.3 (tento dosový program nepodporuje dlouhá jména souborů). Překládaný soubor musí být ve stejném adresáři, jako program ASM51.EXE. První řádek programu musí definovat procesor, použijte zápis **$MODxx51** (pro tento případ musí být v adresáři ještě soubor MODxx51).

HEX2BIN.EXE převede výsledný soubor ve formátu Intel Hex do binární podoby (což je výhodnější pro download). Pro použití opět platí, že konvertovaný soubor musí mít dosové jméno a musí být ve stejném adresáři jako tento program.

Použití obou programů je ukázáno dále. Programy najdete na doprovodném **CD-ROM** v adresáři **ASM51**.

Pokud vám uváděné programy nevyhovují, můžete pochopitelně používat jiné překladače kompatibilní s řadou 8051.

Program pro mikrořadič AN2131 (VYSTUPY.ASM)

Hlavní idea programu byla popsána v úvodu této kapitoly.

Připomeňme, že program musí přepnout porty do výstupního směru. Což provede zápisem samých jedniček (hexadecimálně FFH) do registrů **OEA až OEC**.

Potom bude program procházet nekonečnou smyčkou, v níž bude neustále přenášet hodnoty zapsané do zvolených buňek vnější RAM (zapíše je tam ovládací program počítače přes USB) na porty ***OUTA až OUTC***. Pochopitelně, pokud se hodnoty v paměti nebudou měnit (program nebude odesílat nová data), nebude se ani měnit stav portů.

Tolik obecně, nyní si probereme jednotlivé kroky trochu podrobněji.

1. Nejdříve je nutno zapsat hodnotu FFH do registrů OEA až OEC. Pozor, nejedná se o běžné registry umístěné ve vnitřní RAM (tam bychom použili instrukci **MOV**). Tyto registry jsou umístěny ve vnější RAM a přístup k nim musí být prováděn pomocí instrukce **MOVX**. Pro méně znalé připomeňme, že do registru **DPTR** musí být vložena adresa příslušné buňky vnější RAM (např. pro první port adresa **OEA**) a v akumulátoru (registru A) pak hodnota, kterou chceme zapsat. Takže tyto 3 řádky jsou nutné pro nastavení všech bitů portu **OEA** do stavu „log.1" (tedy pro přepnutí všech vývodů do výstupního směru):

```
MOV DPTR,#OEA
MOV A,#0FFH
MOVX @DPTR,A
```

Zápis do dalších registrů by mohl probíhat podobně. Operaci si však můžeme zjednodušit. Registr **OEB** následuje v paměti za **OEA**, za **OEB** je zase **OEC**. Stačí tedy obsah registru **DPTR** zvětšit o 1 (instrukcí **INC**) a zapsat obsah akumulátoru (předchozí instrukce jej nezměnily) do nové buňky vnější RAM:

```
INC DPTR
MOVX @DPTR,A
INC DPTR
MOVX @DPTR,A
```

2. Po této inicializaci se spustí hlavní smyčka. Jejím úkolem je přenést hodnoty uložené počítačem do vnější RAM na porty **OUTA až OUTC**. Pro čtení/zápis ve vnější RAM se opět používá instrukce **MOVX**. Problém ale spočívá ve skutečnosti, že se musí registr **DPTR** neustále přepínat. Při čtení dat uložených počítačem obsahuje **DPTR** jinou adresu, než při zápisu na port. V realizaci je však využito skutečnosti, že mikrořadič AN2131Q disponuje **dvojitým registrem** DPTR (přepíná se instrukcí **INC DPS**). Nejdříve tedy nahrajeme adresu odpovídající výstupnímu registru portu A:

```
MOV DPTR,#OUTA
```

Potom přepneme DPTR pomocí instrukce:

```
INC DPS
```

Dále vložíme do DPTR adresu buňky vnější RAM, do které počítač zapsal data pro port A:

```
MOV DPTR,#DATAA
```

Údaj přeneseme do akumulátoru:

```
MOVX A,@DPTR
```

Pak přepneme DPTR zpět:

```
INC DPS
```

A obsah akumulátoru (s daty poslanými počítačem) zapíšeme na port A:

```
MOVX @DPTR,A
```

Zápis hodnot pro zbylé porty bude probíhat podobně. Po zápisu na port C se program vrací zpět na návěští **SM** a opakuje tak přenos dat v nekonečné smyčce.

3. Pro uložení dat poslaných z počítače jsou vyhrazeny buňky označené jako DATAA, DATAB a DATAC. Pro snadnou úpravu programu jsou tyto buňky umístěny těsně za samotným programem (vnější RAM je použitelná nejen pro zápis dat, ale i pro uložení řídicího programu mikrořadiče). Pozice těchto proměnných se sice

Tab. 9.1. Výpis programu VYSTUPY.ASM

```
            $MODxx51

            ;definice registrů:
CPUCS       EQU 7F92H
PORTACFG    EQU 7F93H
PORTBCFG    EQU 7F94H
PORTCCFG    EQU 7F95H
OUTA        EQU 7F96H
OUTB        EQU 7F97H
OUTC        EQU 7F98H
PINSA       EQU 7F99H
PINSB       EQU 7F9AH
PINSC       EQU 7F9BH
OEA         EQU 7F9CH
OEB         EQU 7F9DH
OEC         EQU 7F9EH
I2CS        EQU 7FA5H
I2DAT       EQU 7FA6H
DPS         EQU 86H

            ;začátek programu:
START:      MOV DPTR,#OEA     ;DPTR na OEA
            MOV A,#0FFH       ;vše výstupní
            MOVX @DPTR,A      ;PORTA je výstupní
            INC DPTR          ;DPTR na OEB
            MOVX @DPTR,A      ;PORTB je výstupní
            INC DPTR          ;DPTR na OEC
            MOVX @DPTR,A      ;PORTC je výstupní
            ;hlavní smyčka:
SM:         MOV DPTR,#OUTA    ;DPTR na OUTA
            INC DPS           ;přepne DPTR
            ;PORTA:
            MOV DPTR,#DATAA   ;DPTR na DATAA
            MOVX A,@DPTR      ;čte z DATAA
            INC DPS           ;přepne DPTR
            MOVX @DPTR,A      ;zapíše A na OUTA
            ;PORTB:
            INC DPTR          ;DPTR na OUTB
            INC DPS           ;přepne DPTR
            INC DPTR          ;DPTR na DATAB
            MOVX A,@DPTR      ;čte z DATAB
            INC DPS           ;přepne DPTR
            MOVX @DPTR,A      ;zapíše A na OUTB
            ;PORTC:
            INC DPTR          ;DPTR na OUTC
            INC DPS           ;přepne DPTR
            INC DPTR          ;DPTR na DATAC
            MOVX A,@DPTR      ;čte z DATAC
            INC DPS           ;přepne DPTR
            MOVX @DPTR,A      ;zapíše A na OUTC
            LJMP SM           ;zpět do smyčky

            ;výpočet adres dat portů:
DATAA       EQU $             ;PORTA
DATAB       EQU $+1           ;PORTB
DATAC       EQU $+2           ;PORTC

            END
```

změní, když upravíme program (např., když se změní jeho délka), to však není na škodu. Návěští **DATAA až DATAC** generovaná překladačem se této upravené situaci přizpůsobí:

```
DATAA   EQU $
DATAB   EQU $+1
DATAC   EQU $+2
```

4. Program počítače musí zajistit download programu a dále ukládání nových hodnot v případě, že uživatel žádá změnu stavu vývodů. Nové hodnoty se zapíšou do odpovídající buňky vnější RAM v rozmezí **DATAA až DATAC**.

Výpis celého programu VYSTUPY.ASM je v tab. 9.1.

Překlad programu pomocí ASM51

Najděte adresář s překladačem (pochopitelně není možno spouštět jej z CD-ROM, musíte si jej zkopírovat na pevný disk). Do tohoto adresáře také zkopírujte překládaný soubor. Napište na příkazovou řádku:

```
ASM51.EXE VYSTUPY.ASM
```

Je-li vše v pořádku, vzniknou soubory **VYSTUPY.HEX** (přeložená forma) a **VYSTUPY.LST** (protokol o překladu). Proveďte převod do binární podoby tím, že na příkazovou řádku zapíšete:

```
HEX2BIN.EXE VYSTUPY.HEX
```

Je-li vše v pořádku vznikne soubor **VYSTUPY.BIN** . Jeho výpis v hexadecimální podobě je následující:

VYSTUPY.BIN (41 bajtů)

```
90 7F 9C 74 FF F0 A3 F0 A3 F0 90 7F 96 05 86 90
00 29 E0 05 86 F0 A3 05 86 A3 E0 05 86 F0 A3 05
86 A3 E0 05 86 F0 02 00 0A
```

Program pro Windows

Poslední částí příkladu je realizace ovládacího programu pro počítač (PC).

Program musí zavést soubor **VYSTUPY.BIN** do vnější RAM mikrořadiče AN2131Q. Tato operace proběhne pouze jednou, při spuštění programu. Program zůstává v RAM do chvíle, než odpojíme přípravek **USB2131KIT** od počítače.

Dále je třeba, aby program zajistil přenos dat zadávaných uživatelem na příslušný port. Jedná se pouze o zápis bajtu na jednu z adres **DATAA až DATAC**.

Použijeme zdroj!

Jedním z větších problému realizace popsaných úkolů bude zavedení programu do RAM mikrořadiče. Tuto operaci lze provést několika způsoby:

- vytvořit **pole** do nějž vložíme operační kódy instrukcí, které realizují program (tedy opíšeme obsah souboru VYSTUPY.BIN); tento postup je dosti nevýhodný (je třeba opsat jednotlivé bajty souboru, při změně programu musíme vše provádět znovu); použito v [8],
- připojit do programu operace, které **načtou data ze souboru** VYSTUPY.BIN; tento postup je univerzální (výhodou je navíc, že při změně programu pro mikrořadič se prostě načte nový obsah souboru); použito v [8],
- připojit program pro mikrořadič do programu pro Windows - v terminilogii programátorů se mluví o **zdroji** (resource); tento způsob je sice komplikovaný (pro začátečníky) nikoli však na provedení, ale na pochopení (vysvětlíme):

- kromě instrukcí, které realizují program pro Windows, může být v EXE souboru uloženo mnoho dalších dat (např. kurzory nebo bitmapy, které program používá; může se jednat i o speciální formáty, tedy vlastně cokoli),
- zdroj se stane součástí EXE souboru (soubor VYSTUPY.BIN pak již není potřebný pro běh programu, pouze pro nový překlad); tím lze zjednodušit práci s programem (nemusíme kopírovat tolik souborů; program pro mikrořadič nemůže nikdo modifikovat bez nového překladu - to může být někdy chyba, jindy výhoda),
- funkce pro podporu downloadu programu mikrořadiče AN2131Q jsou součástí rozhraní **EZUSB**, jedná se o metodu **DownLoadFromRes** třídy **TAN2131**.

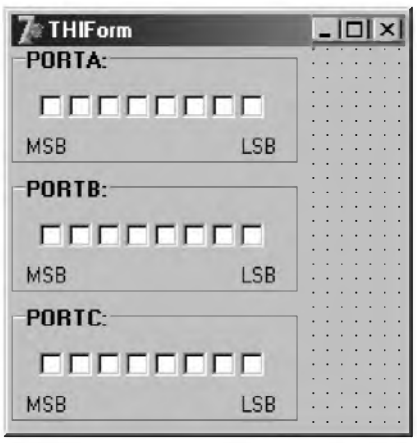
Návrhový formulář

Aplikace bude používat návrhový formulář (obr. 9.1) podobný formuláři z předchozího příkladu (z kapitoly 8). Opět jsou použity 3 komponenty typu GroupBox pod názvy: **gbPORTA**, **gbPORTB** a **gbPORTC**. Do těchto komponent bylo umístěno po 8 komponentách typu CheckBox (např. pro skupinu **gbPORTA** jsou jména **cbPORTA7** až **cbPORTA0**). Dále jsou umístěny pomocné komponenty typu Label s popisky (MSB značí nejvýznamnější bit, LSB pak nejméně významný bit).

Dále byly vygenerovány události: **OnCreate** formuláře (**FormCreate**), **OnDestroy** formuláře (**FormDestroy**) a **PORTClick** (OnClick všech komponent typu CheckBox; vyberou se všechny a přiřadí se jim společná událost).

Inicializace

V události **FormCreate** provedeme inicializaci programu. Nejdříve vytvoříme instanci třídy **TAN2131** (to již známe) a potom provedeme download programu pro mikrořadič AN2131Q.



Obr. 9.1. Návrhový formulář

Pro tento účel je třeba nejdříve mikrořadič zastavit (držet jej v resetu), zápisem hodnoty **01H** do registru **CPUCS**:

```
AN2131->RAM[CPUCS]=0x01;
```

Následně provedeme download. Metoda **TAN2131::DownLoadFromRes** má tyto parametry: startovací adresa (zde **0**, program bude zaveden na začátek RAM), instance aplikace (určuje, kde je uložen zdroj, může být uložen například v DLL; pro náš případ napíšeme **HInstance** - zdroj je přímo v našem programu), název zdroje (zvolil jsem **VYSTUPY**, což koresponduje s názvem programu) a typ zdroje (zvolil jsem **PROGRAM**):

```
AN2131->DownLoadFromRes(
  0,
  int(HInstance),
  "VYSTUPY",
  "PROGRAM");
```

Nakonec program spustíme zrušením resetu, zápisem hodnoty **00H** do registru **CPUCS**:

```
AN2131->RAM[CPUCS]=0x00;
```

Metoda **TAN2131::DownLoadFromRes** vrací délku zdroje (tedy vlastně délku programu). Protože je program zaveden od adresy 0, lze tento údaj s úspěchem použít pro výpočet adres buněk **DATAA až DATAC**.

Odesílání dat

Vlastní odesílání dat na porty je řešeno v události **PORTClick** společně pro libovolný port. Pomocí metody **BinToByte** (z jednotky EZUSB) se z jednotlivých stavů komponent CheckBox sestaví bajt, který se zapíše do příslušné buňky vnější RAM (např. do **DATAA** pro port A):

```
AN2131->RAM[DATAA]=BinToByte(
  cbPORTA7->Checked,
  cbPORTA6->Checked,
  cbPORTA5->Checked,
  cbPORTA4->Checked,
  cbPORTA3->Checked,
  cbPORTA2->Checked,
  cbPORTA1->Checked,
  cbPORTA0->Checked);
```

Výpis celého programu s komentáři:

VYSTFORM.H:

```
#ifndef VYSTFORMH
#define VYSTFORMH
//---------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------
#include "EZUSB.h"
#include <ExtCtrls.hpp>
#include <AppEvnts.hpp>
//---------------------------------------
class THlForm : public TForm
{
__published:
    TGroupBox *gbPORTA;
```

```
    //.
    //. zkráceno
    //.
  TCheckBox *cbPORTC0;
  void __fastcall FormCreate(
    TObject *Sender);
  void __fastcall FormDestroy(
    TObject *Sender);
  void __fastcall PortClick(
    TObject *Sender);
private:        // User declarations
  TAN2131 *AN2131;
  //adresy pro zápis dat na porty:
  int DATAA,DATAB,DATAC;
public:         // User declarations
  __fastcall THlForm(TComponent* Owner);
};
//---------------------------------------
extern PACKAGE THlForm *HlForm;
//---------------------------------------
#endif
```

VYSTFORM.CPP:

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

#include "VystForm.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------
__fastcall THlForm::THlForm(
  TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
{
  TDeviceDescriptor dd;
  try{
    //vytvoření instance TAN2131:
    AN2131=new TAN2131;
    //čtení deskriptoru:
    dd=AN2131->DeviceDescriptor;
    //zobrazení VID a PID:
    Caption="VID="+WordToHex(dd.VID)
          +", PID="+WordToHex(dd.PID);

    //zastaví procesor:
    AN2131->RAM[CPUCS]=0x01;
    //download programu
    //a výpočet adresy pro data portů:
    DATAA=AN2131->DownLoadFromRes(
      0,
      int(HInstance),
      "VYSTUPY",
      "PROGRAM");
    DATAB=DATAA+1;
    DATAC=DATAA+2;
    //rozběhne procesor:
    AN2131->RAM[CPUCS]=0x00;
  }
  catch(Exception &E){
    Application->MessageBox(
      E.Message.c_str(),
      "EZUSB",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
}
//---------------------------------------
void __fastcall THlForm::FormDestroy(
  TObject *Sender)
{
  if(AN2131)
    delete AN2131;
}
//---------------------------------------
void __fastcall THlForm::PortClick(
  TObject *Sender)
{
  //sestaví bajt pro DATAA:
  AN2131->RAM[DATAA]=BinToByte(
    cbPORTA7->Checked,
    cbPORTA6->Checked,
    cbPORTA5->Checked,
    cbPORTA4->Checked,
    cbPORTA3->Checked,
    cbPORTA2->Checked,
    cbPORTA1->Checked,
    cbPORTA0->Checked);

  //sestaví bajt pro DATAB:
  AN2131->RAM[DATAB]=BinToByte(
    cbPORTB7->Checked,
    cbPORTB6->Checked,
    cbPORTB5->Checked,
    cbPORTB4->Checked,
    cbPORTB3->Checked,
    cbPORTB2->Checked,
    cbPORTB1->Checked,
    cbPORTB0->Checked);

  //sestaví bajt pro DATAC:
  AN2131->RAM[DATAC]=BinToByte(
    cbPORTC7->Checked,
    cbPORTC6->Checked,
    cbPORTC5->Checked,
    cbPORTC4->Checked,
    cbPORTC3->Checked,
    cbPORTC2->Checked,
    cbPORTC1->Checked,
    cbPORTC0->Checked);
}
```

A teď ten zdroj

Pro přidání zdroje do programu aktivuje ve vývojovém prostředí C++ Builder položku menu **File|New**. V dialogu vyberte možnost **Text** (vytvoříte nový textový soubor). Soubor uložte pod jménem **PROGRAM.RC** (příponu zachovejte).

Nyní připojte tento soubor do projektu např. pomocí menu položkou **Project|Add To Project**.

Obsah souboru je uveden dále. Jedná se o jediný řádek, který definuje zdroj s názvem **VYSTUPY** a který je typu **PROGRAM**. Zbytek řádku stanovuje, že data zdroje tvoří soubor **VYSTUPY.BIN**.

PROGRAM.RC:

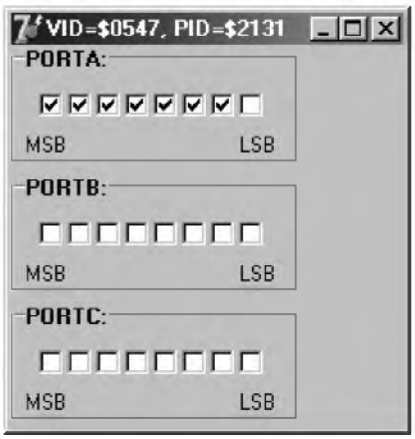
```
VYSTUPY PROGRAM VYSTUPY.BIN
```

Test programu

Zvolte port a připojte na něj kablíkem přípravek **AT8LED**. Spusťte program **VYSTUPY.EXE**.

Je-li vše v pořádku (kontroluje se úspěšnost otevření zařízení **USB2131KIT** a downloadu programu), lze kliknout na libovolné pole. LED přípravku **AT8LED** by se měly podle aktuálního stavu rozsvítit nebo zhasnout.

Připomeňme, že na přípravku **AT8LED** svítí LED při „log. 0" (pole nezaškrtnuto), při „log. 1" (pole zaškrtnuto) je LED zhasnuta.



Obr. 9.2. Aplikace VYSTUPY.EXE v akci

Stav podle obr. 9.2 odráží případ, ve kterém byl přípravek **AT8LED** připojen na port A. Na přípravku pak svítila pouze LED v nejnižším bitu. Ostatní byly zhasnuté.

10. Impulsní generátor

Po předchozích příkladech, určených především pro vysvětlení problematiky ovládání mikrořadiče AN2131 pomocí počítače, si ukážeme konstrukci prvního zařízení. Jedná se o impulsní generátor.

Základní parametry

Impulsní generátor vytváří obdélníkový signál. Obvykle je možno nastavovat kmitočet, střídu a případně i strmost vzestupné a sestupné hrany.

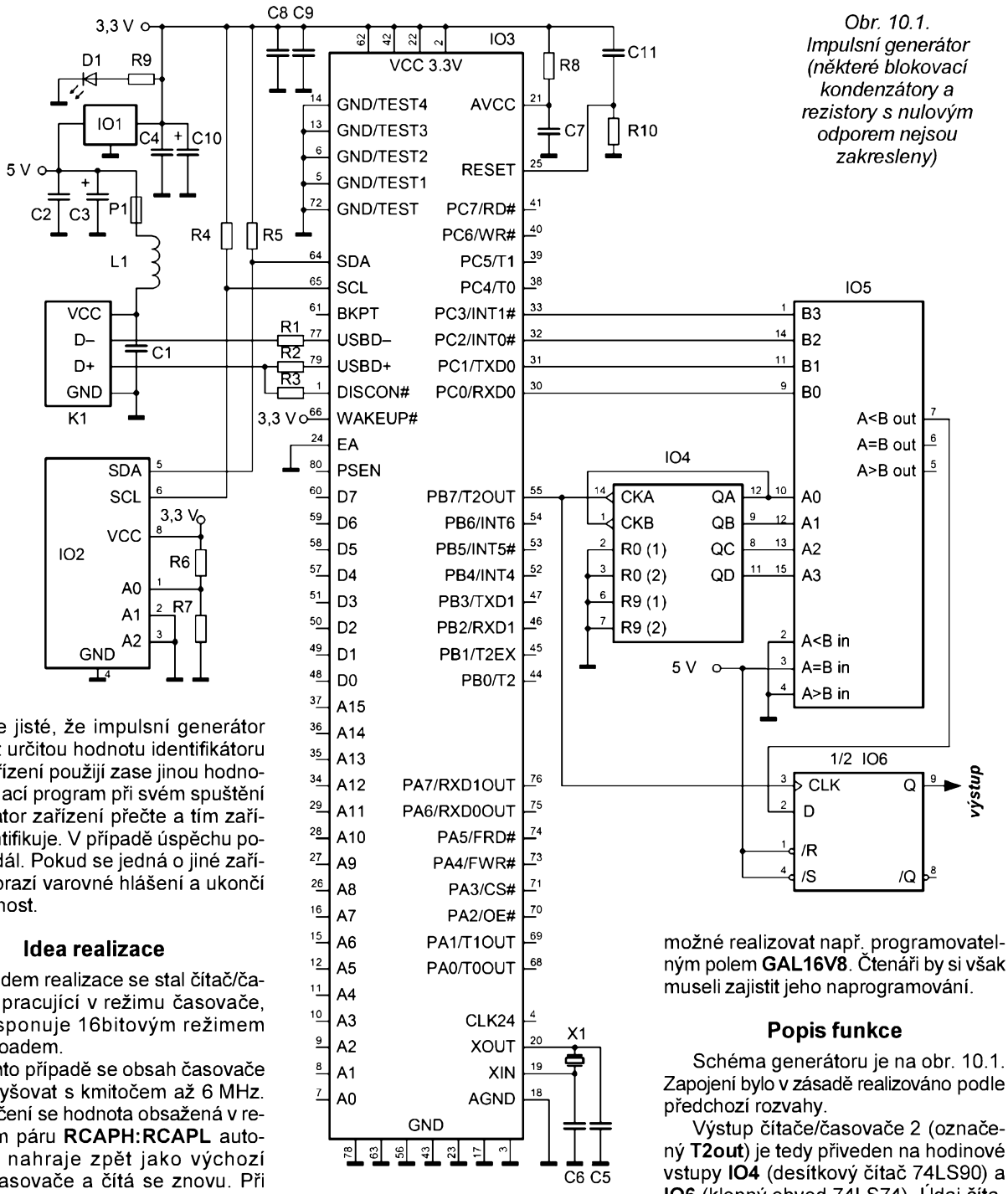
V našem případě vytvoříme impulsní generátor, který dovoluje nastavovat kmitočet a střídu, přičemž v převážné míře vystačíme s periferiemi obsaženými v samotném mikrořadiči AN2131.

Obrovskou výhodou naší realizace je skutečnost, že generátor je napájen z USB. Máme-li tedy volný port, stačí generátor jen připojit k počítači a spustit ovládací program. Lze vytvořit i takovou variantu ovládacího programu, která mění parametry signálu automaticky - např. přeladuje kmitočet (to však nebudeme uvažovat).

Identifikace zařízení

V následujících kapitolách bude předvedena realizace dalších zařízení. Zatím víme, že ovladač přiřazuje přístrojům na bázi mikrořadiče AN2131 jména automaticky (liší se koncovým číslem). Nyní je třeba rozeznat, zda připojené zařízení je skutečně impulsní generátor a ne něco jiného. Případná záměna by mohla vést třeba i k poškození přístroje.

Problém identifikace zařízení můžeme řešit nejsnáze pomocí konfigurační E^2PROM. Podle tab. 2.5 lze 5. a 6. bajt použít k uložení identifikátoru zařízení



Obr. 10.1. Impulsní generátor (některé blokovací kondenzátory a rezistory s nulovým odporem nejsou zakresleny)

(DID). Je jisté, že impulsní generátor bude mít určitou hodnotu identifikátoru a jiná zařízení použijí zase jinou hodnotu. Ovládací program při svém spuštění identifikátor zařízení přečte a tím zařízení identifikuje. V případě úspěchu pokračuje dál. Pokud se jedná o jiné zařízení, zobrazí varovné hlášení a ukončí svou činnost.

Idea realizace

Základem realizace se stal čítač/časovač 2 pracující v režimu časovače, který disponuje 16bitovým režimem s autoreloadem.

V tomto případě se obsah časovače může zvyšovat s kmitočtem až 6 MHz. Po přetečení se hodnota obsažená v registrovém páru **RCAPH:RCAPL** automaticky nahraje zpět jako výchozí obsah časovače a čítá se znovu. Při každém přetečení se na vývodu **T2out** (viz obr. 2.2) vygeneruje impuls o délce jednoho hodinového cyklu (hodinový kmitočet je 24 MHz). Dělicí poměr poskytovaný časovačem 2 obsáhne rozsah výstupního kmitočtu zhruba 100 Hz až 6 MHz. Dolní mez je ještě možné rozšířit programově.

Jako generátor pulsů s proměnnou šířkou (pulsně-šířkový modulátor) můžeme použít vnější čítač, jehož obsah se porovnává s předem zvolenou konstantou. Je-li obsah čítače nižší než stanovená konstanta, je na výstupu „log. 1". V opačném případě pak „log. 0". Budeme-li požadovat střídu v rozsahu 0 až 1 s krokem 0,1, lze použít desítkový čítač. Pokud např. zvolíme konstantu 6, bude prvních 6 cyklů čítače (pro jeho obsah 0 až 5) na výstupu „log. 1". Po zbylé 4 cykly (pro obsah čítače 6 až 9) je na výstupu „log. 0". Pro zvolenou konstantu dostaneme tedy střídu 0,6. Podobně můžeme odvodit chování pro ostatní hodnoty konstant.

Jako desítkový čítač byl použit klasický obvod **74LS90** a jako komparátor stavu čítače také klasický obvod **74LS85**. Vzhledem k tomu, že čítač je asynchronního typu a komparátor produkuje hazardní stavy (prakticky zjištěno) je třeba signál ještě upravit. Hazardní stavy lze nejjednodušeji odfiltrovat klopným obvodem D, který vzorkuje signál v protifázi. Protože má čítač 74LS90 hodinový vstup citlivý na sestupnou hranu, byl použit klasický obvod **74LS74** (klopný obvod D s hodinovým vstupem citlivým na náběžnou hranu).

K volbě součástek lze říci, že byla silná snaha zvolit takové součástky, které lze snadno obstarat. Jinak je jasné, že celý pulsně-šířkový modulátor je možné realizovat např. programovatelným polem **GAL16V8**. Čtenáři by si však museli zajistit jeho naprogramování.

Popis funkce

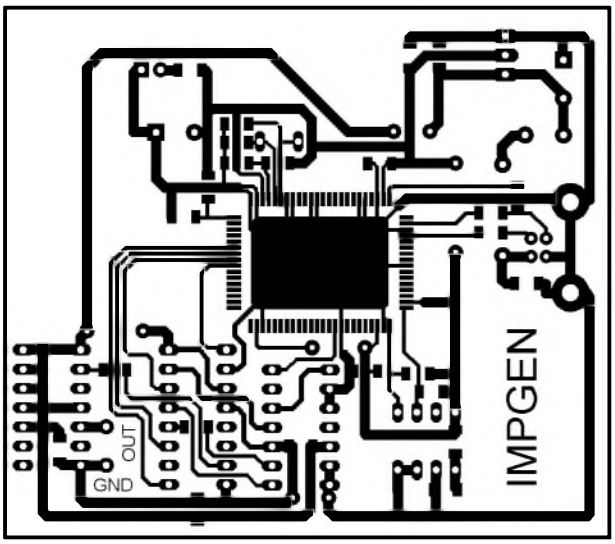
Schéma generátoru je na obr. 10.1. Zapojení bylo v zásadě realizováno podle předchozí rozvahy.

Výstup čítače/časovače 2 (označený **T2out**) je tedy přiveden na hodinové vstupy **IO4** (desítkový čítač 74LS90) a **IO6** (klopný obvod 74LS74). Údaj čítače **IO4** se porovnává se 4bitovou konstantou nastavenou na vývodech **PC0 až PC3** ve velikostním komparátoru **IO5** (74LS85). Výstupní signál velikostního komparátoru je zbaven zákmitů v klopném obvodu **IO6**.

Vzhledem k použití desítkového čítače je jasné (čítač dělí signál generovaný mikrořadičem deseti), že výstupní kmitočet impulzního generátoru je v rozsahu 10 Hz až 600 kHz. Střída je nastavitelná po krocích 0,1 v rozsahu 0 až 1.

Konstrukce

Při realizaci generátoru byly použity většinou součástky SMD a deska s jednostrannými plošnými spoji. Pomocné propojky jsou realizovány buď ze strany součástek nebo ze strany spojů (některé také pomocí rezistorů SMD s nulovým odporem).



Obr. 10.2. Obrazec plošných spojů na desce impulsního generátoru (měř.: 1 : 1, vodorovný rozměr desky je 80 mm)

Obrazec spojů je na obr. 10.2, rozmístění součástek a propojek na obou stranách desky je na obr. 10.3 a obr. 10.4.

Před pájením součástek doporučuji nejdříve vyvrtat všechny díry a potom na straně spojů osadit součástky SMD. Pak osadíme vývodové součástky a propojky na straně součástek. Nakonec se zapojí propojky na straně spojů.

Seznam součástek

impulsního generátoru

R1, R2	27 Ω. SMD 1206
R3	1, 5 k Ω, SMD 1206
R4 až R6	2,2 kΩ, SMD 1206
R7, R11, R12	0 Ω, SMD 1206
R8	1 Ω, SMD 1206
R9	470 Ω, SMD 1206
R10	10 kΩ, SMD 1206
C1, C2, C4, C7 až C9, C11 až C14	100 nF/X7R, SMD 1206
C3, C10	470 µF/25 V, radiální
C5, C6	27 pF/NPO, SMD 1206
L1	33 µH, tlumivka axiální
X1	krystal 12,000 MHz
D1	LED červená, 5 mm, 200 mcd
IO1	LM1084IT-03,3
IO2	24LC01B–I/P
IO3	AN2131QC
IO4	74LS90 (74HCT90)
IO5	74LS85 (74HCT85)
IO6	74LS74 (74HCT74)
P1	RXE025, vratná pojistka
K1	konektor USB1X90B PCB

objímka precizní DIP8 pro IO2 (1 ks)
chladič DO1A pro IO1 (1 ks)
deska s plošnými spoji IMPGEN

Naprogramování konfigurační E²PROM

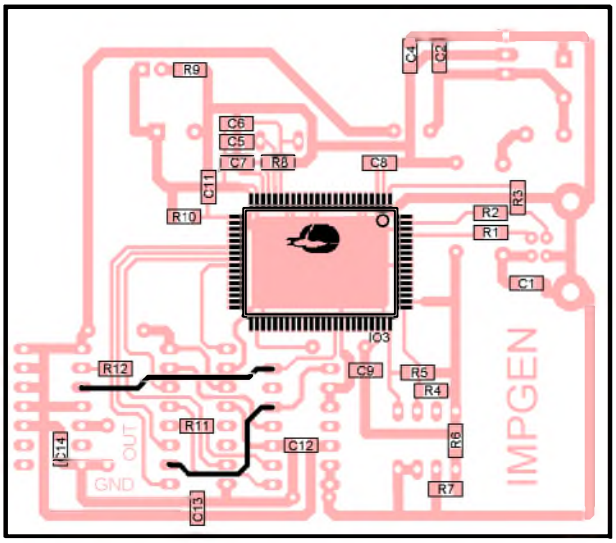
Jak již bylo naznačeno, pro úspěšné rozeznání impulsního generátoru je třeba zapsat do 5. a 6. bajtu identifikátor zařízení. Zvolme např. hodnoty 0x49 a 0x47, kde 0x49 odpovídá písmenu I a 0x47 zase písmenu G (IG jako impulzní generátor). Takže jednotlivé bajty jsou: **0xB0, 0x47, 0x05, 0x31, 0x21, 0x49, 0x47**.

Pro vlastní naprogramování bude vhodné vytvořit si zvláštní program, ve kterém lze zadávat alespoň hodnoty identifikátoru zařízení a uložit je do konfigurační E²PROM. Do prvních pěti bajtů se zapisují stále stejné hodnoty: **0xB0, 0x47, 0x05, 0x31, 0x21**. Naprogramování můžeme zajistit přímo v impulzním generátoru při jeho oživování.

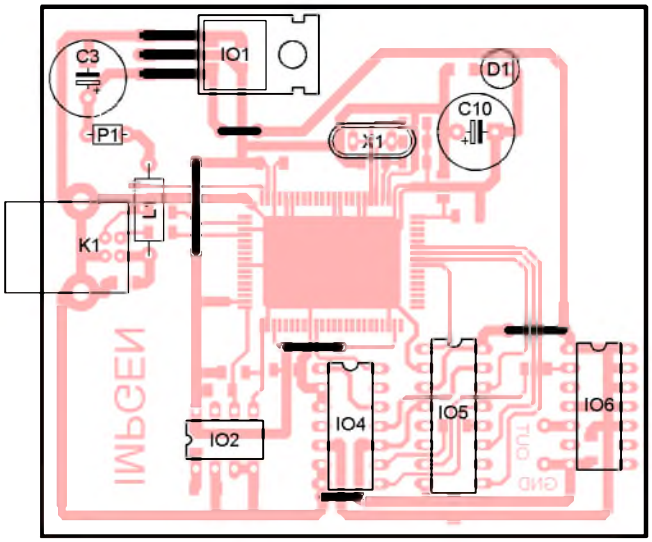
Dále následuje zdrojový text programu mikrořadiče pro zápis do E²PROM a rovněž jsou uvedeny výpisy zdrojových souborů ovládacího programu.

WRITE.ASM:

```
        $MODxx51
I2CS    EQU 7FA5H
I2DAT   EQU 7FA6H

        ;hlavní program:
ZAPIS:  ACALL WRITE
        SJMP $
        ;vyšle start:
START:  MOV DPTR,#I2CS
        MOV A,#80H
        MOVX @DPTR,A
        RET
        ;vyšle adresu paměti pro zápis:
WADDR:  MOV DPTR,#I2DAT
        MOV A,#0A0H
        MOVX @DPTR,A
        AJMP READY
        ;čeká na provedení operace:
READY:  MOV DPTR,#I2CS
        MOVX A,@DPTR
        ANL A,#01H
        CJNE A,#01H,READY
        RET
        ;pošle data/adresu podle A:
SEND:   MOV DPTR,#I2DAT
        MOVX @DPTR,A
        AJMP READY
        ;vyšle stop:
STOP:   MOV DPTR,#I2CS
        MOV A,#40H
        MOVX @DPTR,A
        AJMP READY
        ;ustálení:
DELAY:  MOV R7,#0
DEL1:   MOV R6,#0
DEL2:   DJNZ R6,DEL2
        DJNZ R7,DEL1
        RET
        ;zapíše údaj z MDATA na adr. MADDR:
WRITE:  ACALL START     ;start
        ACALL WADDR     ;pošli adr. E2PROM
        MOV DPTR,#MADDR ;načti adr. buňky
        MOVX A,@DPTR    ;a ulož do A
```



Obr. 10.3. Rozmístění součástek SMD a drátových propojek na straně spojů na desce impulsního generátoru



Obr. 10.4. Rozmístění součástek a drátových propojek na straně součástek na desce impulsního generátoru

```
        ACALL SEND      ;pošli adr. buňky
        MOV DPTR,#MDATA ;načti data
        MOVX A,@DPTR    ;a ulož do A
        ACALL SEND      ;pošli data
        ACALL STOP      ;stop
        ACALL DELAY     ;prodleva
        RET

MDATA   EQU $
MADDR   EQU $+1

        END
```

E2PROGF.H:

```
#ifndef E2PROGFH
#define E2PROGFH
//---------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------
#include "EZUSB.h"
#include <ExtCtrls.hpp>
#include <AppEvnts.hpp>
//---------------------------------------
class THlForm : public TForm
{
__published:    // IDE-managed Components
  TEdit *Edit1;
  TEdit *Edit2;
  TButton *btnProg;
  TLabel *Label1;
  TLabel *Label2;
  void __fastcall FormCreate(TObject *Sen-
der);
   void __fastcall btnProgClick(TObject
*Sender);
private:        // User declarations
  TAN2131 *AN2131;
  //data a adresa pro E2PROM:
  int MDATA,MADDR;
public:         // User declarations
  __fastcall THlForm(TComponent* Owner);
};
//---------------------------------------
extern PACKAGE THlForm *HlForm;
//---------------------------------------
#endif
```

E2PROGF.CPP:

```
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "E2PROGF.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------
__fastcall THlForm::THlForm
(TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
{
  TDeviceDescriptor dd;
  char ret[3];
  try{
    //vytvoření instance TAN2131:
    AN2131=new TAN2131;
    //čtení deskriptoru:
    dd=AN2131->DeviceDescriptor;
    //zobrazení VID a PID:
    ret[2]=0;
    ret[0]=dd.bcdDevice&0xFF;
    ret[1]=(dd.bcdDevice)>>8;
    Caption="DID="+WordToHex(dd.bcdDevice)
      +" \""+ret+"\"";
    //zastav procesor:
    AN2131->RAM[CPUCS]=0x01;
    //download programu:
    MDATA=AN2131->DownLoadFromRes(
      0,
      int(HInstance),
      "WRITE",
      "PROGRAM");
    MADDR=MDATA+1;
  }
  catch(Exception &E){
    Application->MessageBox(
      E.Message.c_str(),
      "EZUSB",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
}
//---------------------------------------
void __fastcall THlForm::btnProgClick(
  TObject *Sender)
{
  //programovací data:
  BYTE Data[7]=
    {0xB0,0x47,0x05,0x31,0x21,0x00,0x00};
  //načte údaje z edit. polí:
  Data[5]=Edit1->Text[1];
  Data[6]=Edit2->Text[1];
  //programování 7 bajtů:
  for(BYTE i=0;i<7;i++){
    //uloží data a adresu:
    AN2131->RAM[MDATA]=Data[i];
    AN2131->RAM[MADDR]=i;
    //rozběhne procesor:
    AN2131->RAM[CPUCS]=0x00;
    //počká:
    Sleep(100);
    //zastaví procesor:
    AN2131->RAM[CPUCS]=0x01;
  }
  //ukončí aplikaci:
  MessageBox(Handle,
    "Odpoj a připoj zařízení",
    "E2PROG",MB_ICONHAND);
  Application->Terminate();
}
```

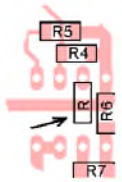
PROGRAM.RC:

```
WRITE PROGRAM WRITE.BIN
```

Přípravek **IMPGEN** připojíme s vymazanou pamětí IO2 k počítači a spus-

Obr. 10.5. Programátor E²PROM

Obr. 10.6. Obsah E²PROM odpovídá impulsnímu generátoru



Obr. 10.7. Zablokování zápisu do E²PROM

tíme program **E2PROG.EXE**. Údaje zadáme podle obr. 10.5 a stiskneme tlačítko **Programuj**. Potom se zobrazí hláška požadující odpojit přípravek od počítače. Současně se program automaticky ukončí.

Po novém připojení přípravku a opětovném spuštění programu **E2PROG.EXE** se musí v titulku okna zobrazit hodnoty podle obr. 10.6.

Abychom zabránili přepisu obsahu E²PROM, je možno připojit vývod WP (zábrana zápisu) na úroveň „log. 1". Můžeme to velice snadno provést připájením rezistoru SMD s odporem 0R mezi vývod 7 IO2 a sběrnici kladného napájecího napětí (viz obr. 10.7).

Program pro mikrořadič

Mikrořadič musí ze strany počítače přijímat 3 bajty (označené jako DATA1 až DATA3). První dva bajty představují obsah registrového páru **RCAP2H:RCAP2L**, poslední bajt odpovídá nastavení konstanty pro pulsně-šířkový modulátor (připojí se na vývody **PC0 až PC3**).

Realizace programu bude dosti podobná jako v příkladu v kapitole 9. V RAM je tedy třeba vytvořit 3 proměnné, do kterých ovládací program přes USB zapíše tyto tři bajty. Hlavní program pak přenáší hodnoty těchto proměnných do registrů **RCAPH2H**, **RCAP2L** a **OUTC**.

Důležité je také konfigurovat čítač/časovač 2 do režimu časovače s autoreloadem a nastavit bit **T2M** (v registru **CKCON**). Dále je nutné konfigurovat vývody **PC0 až PC3** (pomocí registru **OUTC**) jako výstupy a nakonec konfigurovat vývod **PB7** jako **T2out** (nastavit nejvyšší bit registru **PORTBCFG**).

Výpis programu IMPGEN.ASM pro mikrořadič je v tab. 10.1.

Ovládací program pro Windows

Ovládací program je vytvořen na podobném základu, jako v předchozích příkladech. Nebudeme tedy komentovat použité komponenty (viz obr. 10.10).

Větší změny byly provedeny v události **FormCreate**. Nyní není direktivně otevíráno první zařízení EZ-USB, ale vyhledávají se dostupná zařízení a testuje se, zda se jedná o impulsní generátor.

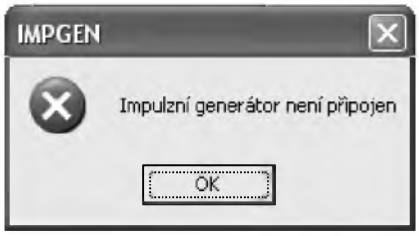
Tab. 10.1. Výpis programu IMPGEN.ASM

```
                $MODxx51
CPUCS           EQU 7F92H
PORTACFG        EQU 7F93H
PORTBCFG        EQU 7F94H
PORTCCFG        EQU 7F95H
OUTA            EQU 7F96H
OUTB            EQU 7F97H
OUTC            EQU 7F98H
PINSA           EQU 7F99H
PINSB           EQU 7F9AH
PINSC           EQU 7F9BH
OEA             EQU 7F9CH
OEB             EQU 7F9DH
OEC             EQU 7F9EH
I2CS            EQU 7FA5H
I2DAT           EQU 7FA6H
DPS             EQU 86H
T2CON           EQU 0C8H
RCAP2H          EQU 0CBH
RCAP2L          EQU 0CAH
TH2             EQU 0CDH
TL2             EQU 0CCH
CKCON           EQU 8EH

                ;inicializace:
START:          MOV DPTR,#OEC       ;DPTR na OEC
                MOV A,#00FH         ;PC0..PC3 výstupní
                MOVX @DPTR,A
                MOV DPTR,#PORTBCFG  ;DPTR na PORTBCFG
                MOV A,#080H         ;PB7 jako T2out
                MOVX @DPTR,A
                MOV DPTR,#OEB       ;DPTR na OEB
                MOV A,#080H         ;PB7 výstupní
                MOVX @DPTR,A
                ;nastavení č/č 2:
                MOV T2CON,#00000100B;autoreload
                MOV CKCON,#00100000B;T2M=1

                ;hlavní smyčka:
SM:             MOV DPTR,#DATA1     ;DPTR na DATA1
                MOVX A,@DPTR        ;údaj pro RCAP2H
                MOV RCAP2H,A        ;nastav
                MOV DPTR,#DATA2     ;DPTR na DATA2
                MOVX A,@DPTR        ;údaj pro RCAP2L
                MOV RCAP2L,A        ;nastav
                MOV DPTR,#DATA3     ;DPTR na DATA3
                MOVX A,@DPTR        ;údaj pro PC0 až PC3
                MOV DPTR,#OUTC      ;DPTR na OUTC
                MOVX @DPTR,A        ;nastav
                LJMP SM             ;zpět do smyčky

DATA1           EQU $               ;RCAP2H
DATA2           EQU $+1             ;RCAP2L
DATA3           EQU $+2             ;PC0 až PC3

                END
```

Pro tento účel se přečte z deskriptoru otevřeného zařízení hodnota **DID** (**bcdDevice**), která musí být rovna **4749H**. Pokud se nepodaří žádné zařízení najít, zobrazí se hláška podle obr. 10.8 a program je ukončen

Další pozoruhodností je skutečnost, že program si ukládá konfigurační údaje do souboru **IMPGEN.INI**, který vznikne v adresáři, ze kterého program spustíme. Při novém spuštění tak dostaneme předchozí nastavení, což může být užitečné.

Vzhledem k poměrně velkému rozmezí generovaných kmitočtů je celé pásmo rozděleno do pěti rozsahů (obr. 10.9). Takto je přeladování jednodušší.



Obr. 10.8. Generátor se nepodařilo najít

Obr. 10.9. Aplikace v akci

Zvolená střída je zobrazována jednak jako textová informace, ale také jako náčrtek generovaného signálu, což dává velmi dobrou představu o skutečném tvaru výstupního signálu (obr. 10.9). Jednotlivé obrázky jsou do programu vloženy jako zdroje (viz následující výpis souboru **PROGRAM.RC**).

Následuje výpis ovládacích programů pro Windows:

IMPGENF.H:

```
#ifndef IMPGENFH
#define IMPGENFH
//---------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------
#include "EZUSB.h"
#include <ExtCtrls.hpp>
//---------------------------------------
class THlForm : public TForm
{
__published:    // IDE-managed Components
  TScrollBar *sbStrida;
  TScrollBar *sbKmitocet;
  TLabel *lbKmitocet;
  TLabel *lbStrida;
  TRadioGroup *rgRozsah;
  TImage *imObr;
  void __fastcall FormCreate(
    TObject *Sender);
  void __fastcall FormDestroy(
    TObject *Sender);
  void __fastcall sbStridaChange(
    TObject *Sender);
  void __fastcall sbKmitocetChange(
    TObject *Sender);
  void __fastcall rgRozsahClick(
    TObject *Sender);
private:        // User declarations
  //generátor:
  TAN2131 *AN2131;
  //adresy pro zápis dat:
  int RCAP2H,RCAP2L,OUTC;
public:         // User declarations
  __fastcall THlForm(TComponent* Owner);
};
//---------------------------------------
extern PACKAGE THlForm *HlForm;
//---------------------------------------
#endif
```

IMPGENF.CPP:

```
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "IMPGENF.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------
__fastcall THlForm::THlForm(
  TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
```

```
{
  TDeviceDescriptor dd;
  int c=0;
   AN2131=NULL;
  try{
    do{
      //zruš předchozí instanci:
      if(AN2131)
        delete AN2131;
      //vytvoření instance TAN2131:
      AN2131=new TAN2131(c++);
      //zjištění deskriptoru:
      dd=AN2131->DeviceDescriptor;
    }while(dd.bcdDevice!=0x4749); //test
    //zastaví procesor:
    AN2131->RAM[CPUCS]=0x01;
    //download programu
    //a výpočet adresy pro data portů:
    RCAP2H=AN2131->DownLoadFromRes(
      0,
      int(HInstance),
      "IMPGEN",
      "PROGRAM");
    RCAP2L=RCAP2H+1;
    OUTC=RCAP2H+2;
    //rozběhne procesor:
    AN2131->RAM[CPUCS]=0x00;
  }
  catch(Exception &E){
    Application->MessageBox(
      "Impulzní generátor není připojen",
      "IMPGEN",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
  //inicializace:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\IMPGEN.INI");
  sbStrida->Position=ini->ReadInteger(
    "NASTAVENI","Strida",5);
  rgRozsah->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Rozsah",4);
  rgRozsahClick(NULL);
  sbStridaChange(NULL);
  sbKmitocet->Position=ini->ReadInteger(
    "NASTAVENI","Kmitocet",200);
  sbKmitocetChange(NULL);
  delete ini;
}
//---------------------------------------
void __fastcall THlForm::FormDestroy(
  TObject *Sender)
{
  //uložení parametrů:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\IMPGEN.INI");
  ini->WriteInteger("NASTAVENI","Strida",
    sbStrida->Position);
  ini->WriteInteger("NASTAVENI","Rozsah",
    rgRozsah->ItemIndex);
    ini->WriteInteger("NASTAVENI","Kmitocet",
    sbKmitocet->Position);
  delete ini;
  if(AN2131)
    delete AN2131;
}
//---------------------------------------
//změna střídy:
void __fastcall THlForm::sbStridaChange(
  TObject *Sender)
{
  //texty:
  char *Text[11]=
    {"log.0","1:9","1:4","3:7","2:3","1:1",
    "3:2","7:3","4:1","9:1","log. 1"};
  //odešle střídu:
  AN2131->RAM[OUTC]=sbStrida->Position;
  //zobrazí doprovodný text:
  lbStrida->Caption=
     FormatFloat("0.0",sbStrida->Position/
10.0)
    +" ("+Text[sbStrida->Position]+")";
  //a ještě obrázek:
  imObr->Picture->Bitmap->LoadFromResource-
ID(
    int(HInstance),sbStrida->Position+100);
}
//---------------------------------------
//jemné ladění ve zvoleném rozsahu:
void __fastcall THlForm::sbKmitocetChange(
  TObject *Sender)
{
  //výpočet kmitočtu:
  double f=600000.0/
    (65536-sbKmitocet->Position);
  //odešle nastavenou hodnotu:
  AN2131->RAM[RCAP2H]=
    (sbKmitocet->Position>>8)&0xFF;
  AN2131->RAM[RCAP2L]=
    (sbKmitocet->Position)&0xFF;
  //úprava textu pro zobrazení:
  if(f>=1e3)
    lbKmitocet->Caption=
      FormatFloat("0.000 kHz",f/1e3);
  else
    lbKmitocet->Caption=
      FormatFloat("0.000 Hz",f);
}
//---------------------------------------
//změna kmitočtového pásma:
void __fastcall THlForm::rgRozsahClick(
  TObject *Sender)
{
  //nutné před změnou nastavení skrolbaru:
  sbKmitocet->Min=0;sbKmitocet->Max=65535;
  //podle pásma zadej min a max:
  switch(rgRozsah->ItemIndex){
    case 0:  //x10 Hz
      sbKmitocet->Min=5536;
      sbKmitocet->Max=59535;
      sbKmitocet->LargeChange=10000;break;
    case 1:  //x100 Hz
      sbKmitocet->Min=59536;
      sbKmitocet->Max=64935;
      sbKmitocet->LargeChange=1000;break;
    case 2:  //x1 kHz
      sbKmitocet->Min=64936;
      sbKmitocet->Max=65475;
      sbKmitocet->LargeChange=100;break;
    case 3:  //x10 kHz
      sbKmitocet->Min=65476;
      sbKmitocet->Max=65529;
      sbKmitocet->LargeChange=10;break;
    case 4:  //x100 kHz
      sbKmitocet->Min=65530;
      sbKmitocet->Max=65535;
      sbKmitocet->LargeChange=1;break;
  }
  //definovaná pozice skrolbaru:
  sbKmitocet->Position=sbKmitocet->Max;
}
```

PROGRAM.RC:

```
IMPGEN PROGRAM IMPGEN.BIN
100 BITMAP OBR_00.BMP
101 BITMAP OBR_01.BMP
102 BITMAP OBR_02.BMP
103 BITMAP OBR_03.BMP
104 BITMAP OBR_04.BMP
105 BITMAP OBR_05.BMP
106 BITMAP OBR_06.BMP
107 BITMAP OBR_07.BMP
108 BITMAP OBR_08.BMP
109 BITMAP OBR_09.BMP
110 BITMAP OBR_10.BMP
```

11. Čítač

V této kapitole je uvedena konstrukce čítače, který měří kmitočet v rozmezí zhruba 1 Hz až 30 MHz.

Základní informace

Většina čítačů měří kmitočet tzv. přímou měřicí metodou. Ta spočívá v tom, že po určitou pevnou dobu (např. po dobu 1 s) počítáme impulsy vstupního signálu. Jelikož je kmitočet definován jako počet změn signálu za jednotku času, změříme skutečně hodnotu kmitočtu vstupního signálu. Pokud je doba měření 1 s, odpovídá námi zjištěný počet impulsů kmitočtu v Hz. Místo pojmu doba měření obvykle spíše používáme pojem **doba otevření hradla čítače**.

V některých případech je nutné dobu otevření hradla čítače měnit. Např. se může stát, že kmitočet je příliš velký a během měření čítač přeteče. Pokud používáme 16bitové čítače, je jejich maximální obsah roven číslu 65535. Je-li tedy kmitočet vyšší než asi 65 kHz, mohou tyto čítače přetéct. Pokud však dobu otevření hradla čítače zkrátíme na 0,1 s, napočítá se pouze desetina počtu impulsů a danou metodu můžeme použít zhruba až do kmitočtu 650 kHz. Načítaný údaj pak musíme před zobrazením vynásobit deseti.

Jindy zase dobu otevření hradla čítače prodlužujeme. To má smysl především při měření signálů s nízkým kmitočtem - pod 10 Hz. Pokud totiž měříme 1 s, dostaneme méně než 10 impulsů a ztrácíme tak přesnost měření. Pokud ale budeme měřit 10 s, napočítáme desetkrát větší počet impulsů a přesnost se zase zvětší. Načítaný údaj pak musíme před zobrazením dělit deseti.

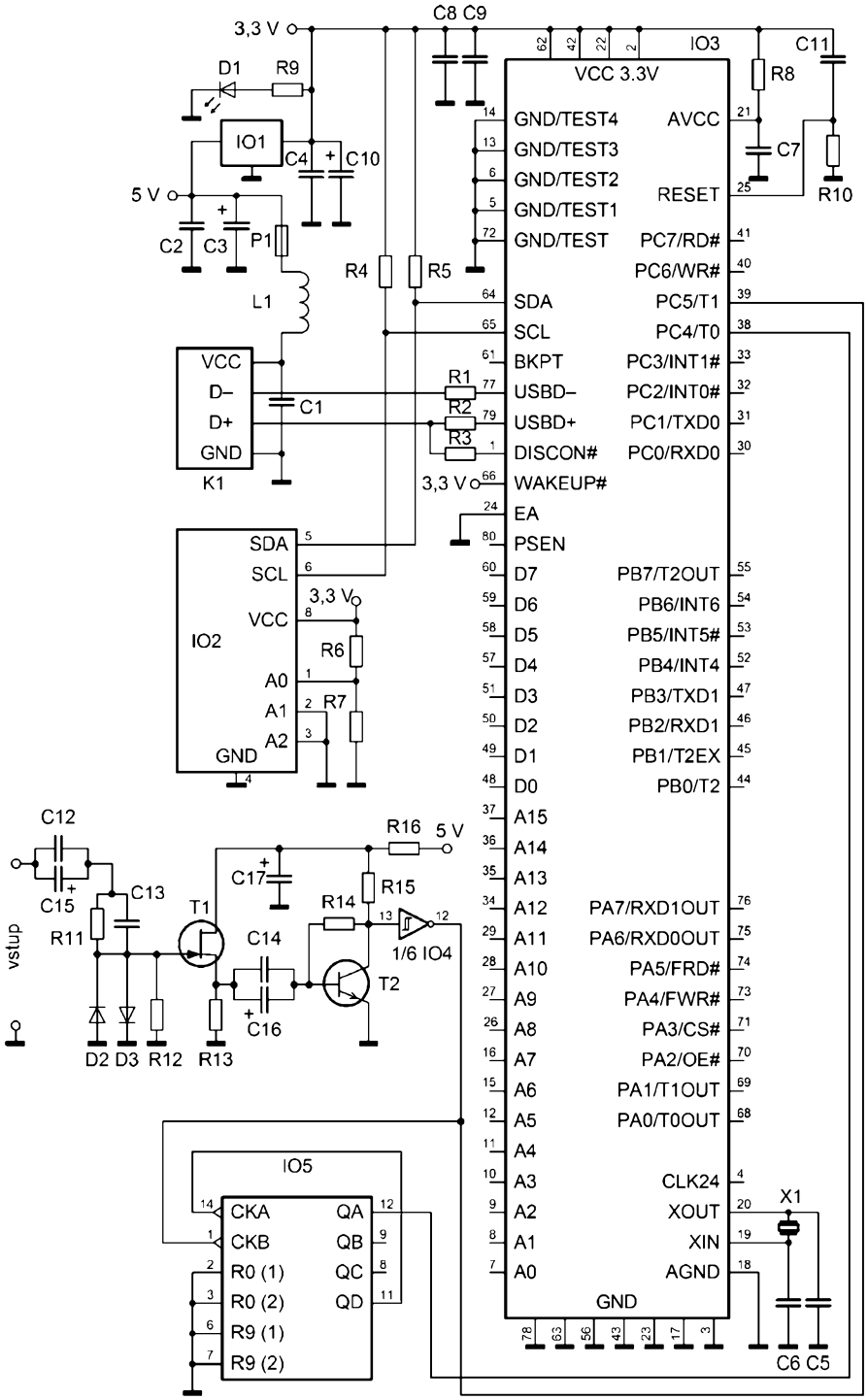
Posledním problémem je skutečnost, že každý čítač má stanoven mezní kmitočet. Pokud na jeho hodinový vstup přivedeme rychlejší signál, nebude čítač pracovat správně. Čítače v mikrořadiči **AN2131** mají mezní kmitočet 6 MHz. Pokud potřebujeme měřit vyšší kmitočty, musíme předřadit vnější předděličku. Pro kmitočty do 30 MHz vystačíme s běžnou děličkou deseti **74LS90**. Načítaný údaj pak musíme před zobrazením vynásobit deseti.

Identifikace zařízení

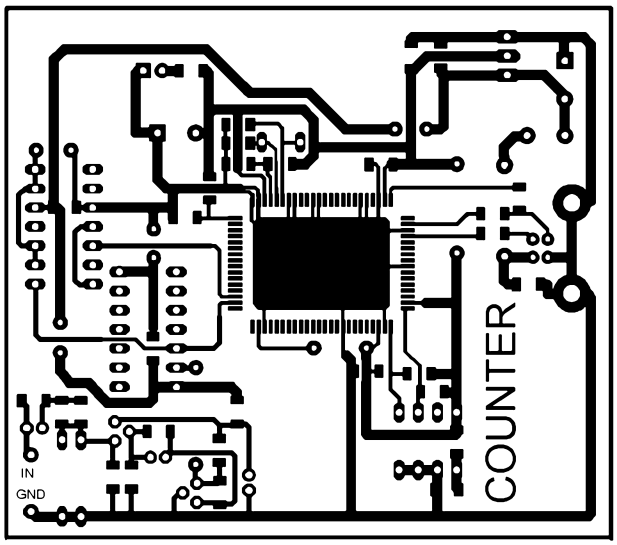
Pro identifikaci zařízení budeme opět používat metodu popsanou v kapitole 10. Čítač bude identifikován znaky **CT**, které uložíme do konfigurační E^2PROM. Tyto znaky představují hexadecimální číslo **5443H**.

Popis funkce

Schéma zapojení vychází z uvedeného popisu funkce obecného čítače a je na obr. 11.1.



↑ *Obr. 11.1. Schéma zapojení čítače (nejsou zakresleny některé blokovací kondenzátory)*



← *Obr. 11.2. Obrazec plošných spojů na desce čítače (měř.: 1 : 1, vodorovný rozměr desky je 80 mm)*

Měřený signál prochází přes vazební kondenzátory C12 a C15 a přes ochranný rezistor R11 na hradlo tranzistoru T1. Kondenzátor C13 zvětšuje citlivost na vysokých kmitočtech, diody D2 a D3 omezují vstupní napětí a rezistor R12 definuje vstupní odpor čítače.

Dále je signál zesílen tranzistorem T2 a nakonec převeden na číslicové úrovně Schmittovým klopným obvodem (hradlem z obvodu IO**4** typu **74LS14**).

Abychom se vyhnuli nutnosti vybírat signál přímý nebo dělený deseti, používáme dva čítače. Přímý signál přichází na vstup **T1** čítače/časovače 1, signál dělený deseti děličkou IO5 (**74LS90**) je přiveden na vstup **T0** čítače/časovače 0.

Určitou zajímavostí může být zapojení děličky IO5. Obvykle se oba stupně děličky propojují do kaskády tak, že výstup QA se připojí na vstup CKB a signál dělený deseti je pak na výstupu QD. Bohužel takto generovaný signál nemá střídu 1 : 1, takže čítač v mikrořadiči by jej nemusel správně měřit. Proto je v naší konstrukci přiveden vstupní signál na vstup CKB a výstup QD je připojen na vstup CKA. Dělený signál je pak na výstupu QA.

Použitý vstupní díl s tranzistory byl poprvé popsán v [5] a je převzat z článku od **Miloše Zajíce**, který byl publikován v časopisu PE 5/97.

Na závěr si ještě uvědomme, s jakými dobami otevření hradla čítače (GATE) musí naše konstrukce pracovat, aby obsáhla rozsah kmitočtů 10 Hz až 30 MHz. Tuto informaci nám podává tab. 11.1. Některá omezení dostáváme s ohledem na délku čítače (maximální údaj 65535), jiná jsou dána mezním kmitočtem čítače v mikrořadiči AN2131 (6 MHz), nakonec i předdělička má mezní kmitočet (asi 30 MHz).

Konstrukce

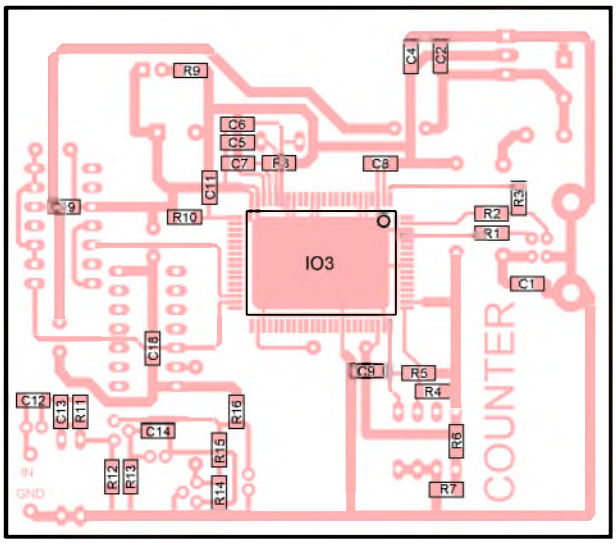
Čítač byl opět realizován se součástkami SMD i vývodovými na desce s jednostrannými plošnými spoji. Pomocné propojky jsou tvořeny krátkými drátky, které jsou umístěny na straně součástek.

Obrazec spojů je na obr. 11.2, rozmístění součástek a propojek na obou stranách desky je na obr. 11.3 a obr. 11.4.

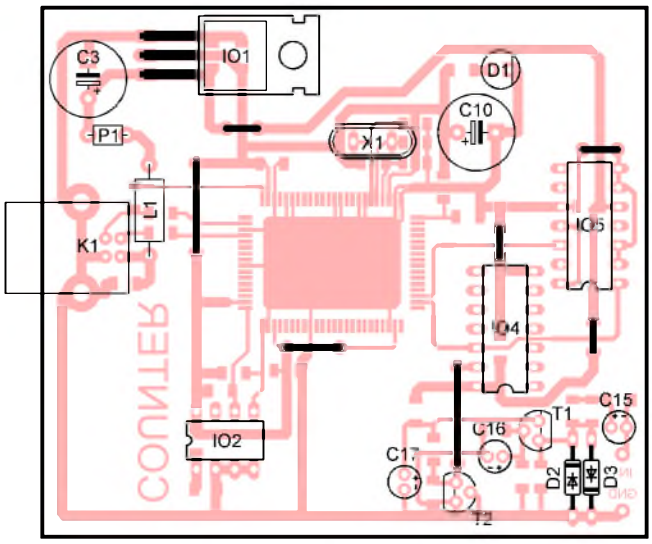
Před pájením součástek doporučuji vyvrtat všechny díry, potom na straně spojů osadit součástky SMD a nakonec zapájet vývodové součástky.

Tab. 11.1. Měřicí rozsahy čítače (f_{max}) v závislosti na době otevření hradla (GATE) a použití předděličky

GATE	předdělička	f_{max}
10 s	ne	6553 Hz
10 s	ano	65535 Hz
1 s	ne	65535 Hz
1 s	ano	655 kHz
0,1 s	ne	655 kHz
0,1 s	ano	6,55 MHz
0,01 s	ne	6 MHz
0,01 s	ano	30 MHz



Obr. 11.3. Rozmístění součástek SMD na straně spojů na desce čítače



Obr. 11.4. Rozmístění součástek a drátových propojek na straně součástek na desce čítače

Seznam součástek

čítače

R1, R2	27 Ω, SMD 1206
R3	1,5 kΩ, SMD 1206
R4 až R6	2,2 kΩ, SMD 1206
R7	0 Ω, SMD 1206
R8	1 Ω, SMD 1206
R9	470 Ω, SMD 1206
R10	10 kΩ, SMD 1206
R11, R13, R15	1 kΩ, SMD 1206
R12	1 MΩ, SMD 1206
R14	56 kΩ, SMD 1206
R16	47 Ω, SMD 1206
C1, C2, C4, C7 až C9, C11, C12, C14, C18, C19	100 nF/X7R, SMD 1206
C3, C10	470 µF/25 V, radiální
C5, C6	27 pF/NPO, SMD 1206
C13	47 pF/NPO, SMD 1206
C15 až C17	47 µF/16 V, radiální
L1	33 µH, tlumivka axiální
X1	krystal 12,000 MHz
D1	LED červená, 5 mm, 200 mcd
D2, D3	1N4148
T1	BF245
T2	BF199
IO1	LM1084IT-03,3
IO2	24LC01B-I/P
IO3	AN2131QC
IO4	74LS14 (74HCT14)
IO5	74LS90 (74HCT90)
P1	RXE025, vratná pojistka
K1	konektor USB1X90B PCB

objímka precizní DIP8 pro IO2 (1 ks)
chladič DO1A pro IO1 (1 ks)
deska s plošnými spoji COUNTER

Naprogramování konfigurační E²PROM

Přípravek **COUNTER** připojíme s vymazanou pamětí IO2 k počítači a spustíme program **E2PROG.EXE**. Údaje zadáme podle obr. 11.5 a stiskneme tlačítko **Programuj**. Potom se zobrazí hláška požadující odpojit přípravek od počítače. Současně se program automaticky ukončí.

Abychom zabránili přepisu obsahu E²PROM, je možné zapojit vývod WP (zábrana zápisu) na úroveň „log. 1". Postup byl již uveden v předchozí kapitole 10 (obr. 10.7).

Program pro mikrořadič

Realizace programu pro mikrořadič AN2131 je poměrně jednoduchá.

Nejdříve je třeba nakonfigurovat oba čítače/časovače do režimu 16bitového čítače vnějšího signálu. Vzhledem k tomu, že vývody **T0** a **T1** jsou na portu C, musíme také povolit alternativní funkci těchto vývodů nastavením bitů 4 a 5 registru **PORTCCFG**.

Potom se již rozbíhá smyčka odměřující dobu otevření hradla čítače. Nejdříve je třeba načíst požadovanou dobu z proměnných **GATEL** (spodní bajt) a **GATEH** (horní bajt), doba otevření hradla čítače je zadána jako 16bitová proměnná proto, aby bylo možné obsáhnout poměrně velký rozsah požadovaných časů. Údaj obsažený v páru **GATEH:GATEL** představuje dobu otevření hradla čítače v ms (tedy 10 pro 0,01s; 100 pro 0,1 s; 1000 pro 1 s a 10000 pro 10 s). Před spuštěním odměru musí být vynulovány obsahy čítačů a příznaky přetečení (mohly by v nich být nenulové údaje z předchozího měření). Nakonec povolíme čítání do obou čítačů. Potom se již spustí čekací smyčka.

Realizace čekací smyčky byla poněkud náročnější, protože u každé instrukce musela být zjištěna doba vykonání. Odměřování času zajišťují proměnné **ODM1, ODM2** (kopie údaje z **GATEL, GATEH**) a **CEK1, CEK2** (odměr doby 993 µs, doladění na 1000 µs je zajištěno několika instrukcemi **NOP**).

Po odměru se zablokuje čítání a načítané hodnoty se uloží do proměnných **CTL0, CTH0, CTL1, CTH1**. Rovněž se přečte stav příznaků přetečení a uloží do proměnné **OVER**.

Ovládací program vlastně pouze uloží hodnoty GATEL, GATEH a vyčká stanovenou dobu otevření hradla čítače. Potom si přečte obsahy ostatních proměnných a podle nich zobrazí naměřený kmitočet.

Výpis programu COUNTER.ASM pro mikrořadič je v tab. 11.2.

Ovládací program pro Windows

Ovládací program pro čítač byl vytvořen na základě programu z kapitoly 10.

Při spuštění se opět pomocí deskriptoru zařízení zjišťuje, zda je čítač připojen. Potom se do mikrořadiče uloží program a přečte konfigurace ze souboru **COUNTER.INI** v aktuálním adresáři.

Ovládací program umožňuje měnit dobu otevření hradla čítače a použít předděličku.

Při změně nastavené doby otevření hradla čítače se vyvolá událost **rgGateClick**. Musí být zajištěno odeslání nových hodnot pro **GATEL** a **GATEH** a reset procesoru (tak předčasně ukončíme předchozí měření). Na konec měření se čeká pomocí časovače.

Po proběhnutí intervalu nastaveného v časovači je vyvolána událost **Aktualizuj**. V ní je třeba přečíst obsahy obou čítačů a převést je do zobrazitelné formy. Podle nastavené doby otevření hradla čítače se zjištěný údaj čítače vynásobí nebo vydělí a dále se uvažuje použití předděličky. Nakonec se ještě zformátuje text pro zobrazení do řádů Hz, kHz nebo MHz.

Obr. 11.5. Zadání identifikátoru čítače do E²PROM

Tab. 11.2. Výpis programu COUNTER.ASM

```
            $MODxx51
PORTCCFG    EQU 7F95H
            ;registry pro časování:
CEK1        EQU 4
CEK2        EQU 5
ODM1        EQU 6
ODM2        EQU 7

            ;inicializace:
INIC:       MOV A,#01010101B
            MOV TMOD,A          ;režim 16b čítačů
            MOV A,#00110000B
            MOV DPTR,#PORTCCFG
            MOVX @DPTR,A        ;T0, T1 k dispozici

            ;smyčka měření:
START:      MOV DPTR,#GATEL
            MOVX A,@DPTR        ;načti GATEL
            MOV ODM1,A          ;nastav do ODM1
            MOV DPTR,#GATEH
            MOVX A,@DPTR        ;načti GATEH
            MOV ODM2,A          ;nastav do ODM2
            CLR A
            MOV TL0,A           ;nulování
            MOV TH0,A           ;obsahu čítačů
            MOV TL1,A
            MOV TH1,A
            CLR TF0             ;nulování příznaků
            CLR TF1
            SETB TR0            ;spuštění odměru
            SETB TR1
            AJMP ODMER1         ;korekce 7 cyklů
ODMER:      NOP                 ;7xNOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
ODMER1:     NOP                 ;11xNOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP                 ;celkem 18 NOP,
            NOP                 ;tedy 3 µs
            MOV CEK1,#LOW(331)
            MOV CEK2,#HIGH(331) ;ještě 1 µs
            ;celkem 993 µs:
ODMSM:      CLR C
            MOV A,CEK1
            SUBB A,#1
            MOV CEK1,A          ;CEK2:CEK1-1
            MOV A,CEK2
            SUBB A,#0
            MOV CEK2,A
            ORL A,CEK1          ;test CEK2:CEK1=0
            JNZ ODMSM
            ;1ms*ODM2:ODM1
            CLR C
            MOV A,ODM1
            SUBB A,#1
            MOV ODM1,A          ;ODM2:ODM1-1
            MOV A,ODM2
            SUBB A,#0
            MOV ODM2,A
            ORL A,ODM1          ;test ODM2:ODM1=0
            JNZ ODMER
            ;uložení výsledků:
            CLR TR0             ;zastavení odměru
            CLR TR1
            MOV DPTR,#CTL0
            MOV A,TL0
            MOVX @DPTR,A  ;ulož TL0
            INC DPTR
            MOV A,TH0
            MOVX @DPTR,A  ;ulož TH0
            INC DPTR
            MOV A,TL1
            MOVX @DPTR,A  ;ulož TL1
            INC DPTR
            MOV A,TH1
            MOVX @DPTR,A  ;ulož TH1
            INC DPTR
            MOV A,TCON
            ANL A,#10100000B
            MOVX @DPTR,A  ;ulož příznaky
            AJMP START    ;a nový odměr

            ;řídicí proměnné:
GATEL       EQU $
GATEH       EQU $+1
CTL0        EQU $+2
CTH0        EQU $+3
CTL1        EQU $+4
CTH1        EQU $+5
OVER        EQU $+6
            END
```

Zajímavostí je použití komponenty **Obr** (typu Shape), která indikuje provedený odměr (bliká jako LED u klasických čítačů). Tak uživatel neztrácí přehled, kdy byl odměr proveden.

Pohled na okno aplikace v akci je na obr. 11.6.

Následuje výpis ovládacích programů pro Windows:

Obr. 11.6. Aplikace COUNTER (čítač) v akci

COUNTERF.H:

```
#ifndef COUNTERFH
#define COUNTERFH
//---------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------
#include "EZUSB.h"
#include <ExtCtrls.hpp>
//---------------------------------------
class THlForm : public TForm
{
__published:
  TLabel *lbKmitocet;
  TRadioGroup *rgGate;
  TTimer *Casovac;
  TRadioGroup *rgPreddel;
  TShape *Obr;
  void __fastcall FormCreate(
```

```
    TObject *Sender);
  void __fastcall FormDestroy(
    TObject *Sender);
  void __fastcall rgGateClick(
    TObject *Sender);
  void __fastcall Aktualizuj(
    TObject *Sender);
private:        // User declarations
  //generátor:
  TAN2131 *AN2131;
  //adresy pro komunikaci:
  int GATEL,GATEH,
      CTL0,CTH0,CTL1,CTH1,OVER;
public:         // User declarations
  __fastcall THlForm(TComponent* Owner);
};
//---------------------------------------
extern PACKAGE THlForm *HlForm;
//---------------------------------------
#endif
```

COUNTERF.CPP:

```
#include <vcl.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "COUNTERF.h"
//---------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------
__fastcall THlForm::THlForm(
  TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
{
  TDeviceDescriptor dd;
  int c=0;
  AN2131=NULL;
  try{
    do{
      //zruš předchozí instanci:
      if(AN2131)
        delete AN2131;
      //vytvoření instance TAN2131:
      AN2131=new TAN2131(c++);
      //zjištění deskriptoru:
      dd=AN2131->DeviceDescriptor;
    }while(dd.bcdDevice!=0x5443); //test
    //zastaví procesor:
    AN2131->RAM[CPUCS]=0x01;
    //download programu
    //a výpočet adresy pro data portů:
    GATEL=AN2131->DownLoadFromRes(
      0,
      int(HInstance),
      "COUNTER",
      "PROGRAM");
    GATEH=GATEL+1;
    CTL0=GATEL+2;
    CTH0=GATEL+3;
    CTL1=GATEL+4;
    CTH1=GATEL+5;
    OVER=GATEL+6;
    //rozběhne procesor:
    AN2131->RAM[CPUCS]=0x00;
  }
  catch(Exception &E){
    Application->MessageBox(
      "Čítač není připojen",
      "COUNTER",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
  //inicializace:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\COUNTER.INI");
  rgPreddel->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Preddel",0);
  rgGate->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Gate",0);
  rgGateClick(NULL);
  delete ini;
  Casovac->Enabled=true;
}
//---------------------------------------
void __fastcall THlForm::FormDestroy(
  TObject *Sender)
{
  //uložení parametrů:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\COUNTER.INI");
  ini->WriteInteger(
    "NASTAVENI","Preddel",rgPreddel->ItemIndex);
  ini->WriteInteger(
    "NASTAVENI","Gate",rgGate->ItemIndex);
  delete ini;
  if(AN2131)
    delete AN2131;
}
//---------------------------------------
//změna GATE:
void __fastcall THlForm::rgGateClick(
  TObject *Sender)
{
  int vGate[4]={10,100,1000,10000};
  //přečte hodnotu gate z pole:
  int Gate=vGate[rgGate->ItemIndex];
  //zastaví procesor:
  AN2131->RAM[CPUCS]=0x01;
  //nastaví gate:
  AN2131->RAM[GATEL]=Gate&0xFF;
  AN2131->RAM[GATEH]=Gate>>8;
  //nastaví časovač:
  Casovac->Interval=Gate*1.2;
  //ustálení:
  Sleep(50);
  //rozběhne procesor:
  AN2131->RAM[CPUCS]=0x00;
}
//---------------------------------------
void __fastcall THlForm::Aktualizuj(
  TObject *Sender)
{
  //údaje čítačů:
  int T0=AN2131->RAM[CTL0]
       +(AN2131->RAM[CTH0]<<8);
  int T1=AN2131->RAM[CTL1]
       +(AN2131->RAM[CTH1]<<8);
  double f;
  bool over;
  //rozsvítí LED:
  Obr->Brush->Color=clRed;
  //výpočet kmitočtu dle volby čítače:
  switch(rgPreddel->ItemIndex){
    case 0:f=T1;
      over=AN2131->RAM[OVER]&0x80;break;
    case 1:f=T0*10;
      over=AN2131->RAM[OVER]&0x20;break;
  }
  //uvážení gate do celkového výsledku:
  switch(rgGate->ItemIndex){
    case 0:f=f*100;break;
    case 1:f=f*10;break;
    case 2:f=f;break;
    case 3:f=f/10;break;
  }
  //naformátování výsledku:
  if(over)
    lbKmitocet->Caption="! OVER !";
  else if(f>=10e6)
    lbKmitocet->Caption=FormatFloat(
      "00.00 MHz",f/1e6);
  else if(f>=1e6)
    lbKmitocet->Caption=FormatFloat(
      "0.000 MHz",f/1e6);
  else if(f>=100e3)
    lbKmitocet->Caption=FormatFloat(
      "000.0 kHz",f/1e3);
  else if(f>=10e3)
    lbKmitocet->Caption=FormatFloat(
      "00.00 kHz",f/1e3);
  else if(f>=1e3)
    lbKmitocet->Caption=FormatFloat(
      "0.000 kHz",f/1e3);
  else
    lbKmitocet->Caption=FormatFloat(
      "0 Hz",f);
  //zhasne LED:
  Application->ProcessMessages();
  Sleep(500);
  Obr->Brush->Color=clBlack;
}
```

PROGRAM.RC:

```
COUNTER PROGRAM COUNTER.BIN
```

12. Programovatelný generátor

Poslední a určitě i nejzajímavější konstrukcí je programovatelný generátor.

Základní informace

Programovatelný generátor dovoluje na svém výstupu vytvářet signál libovolného tvaru. S ohledem na omezené možnosti uvedené konstrukce zajistíme pouze vytvoření klasických průběhů: harmonický, obdélníkový a pilovitý. Rozšířením ovládacího programu lze definovat prakticky libovolný průběh.

Jádro konstrukce programovatelného generátoru je vždy stejné: Tvoří jej paměť, ve které jsou uloženy vzorky generovaného signálu. Tyto vzorky se pomocí převodníku D/A převádějí na napětí. Dále musí existovat způsob, jak do paměti uložit vzorky požadovaného signálu v režimu programování a číst je z paměti požadovanou rychlostí v režimu generování.

V naší konstrukci se opět snažíme co nejvíce zmenšit počet použitých součástek. Hlavní snahou je vyhnout se použití vnější paměti. Ta vždy dosti zkomplikuje zapojení. Je totiž nutné připojit ještě adresní čítač, který vytváří cyklickou posloupnost adres jednotlivých vzorků.

Naštěstí je skutečně možné vnější paměť vynechat. Vystačíme s pamětí v mikrořadiči. Převodník D/A totiž připojíme na datovou sběrnici (vývody D0 až D7) a vzorky generovaného signálu budeme odesílat pomocí rychlého přenosového režimu (viz kapitolu 2).

Identifikace zařízení

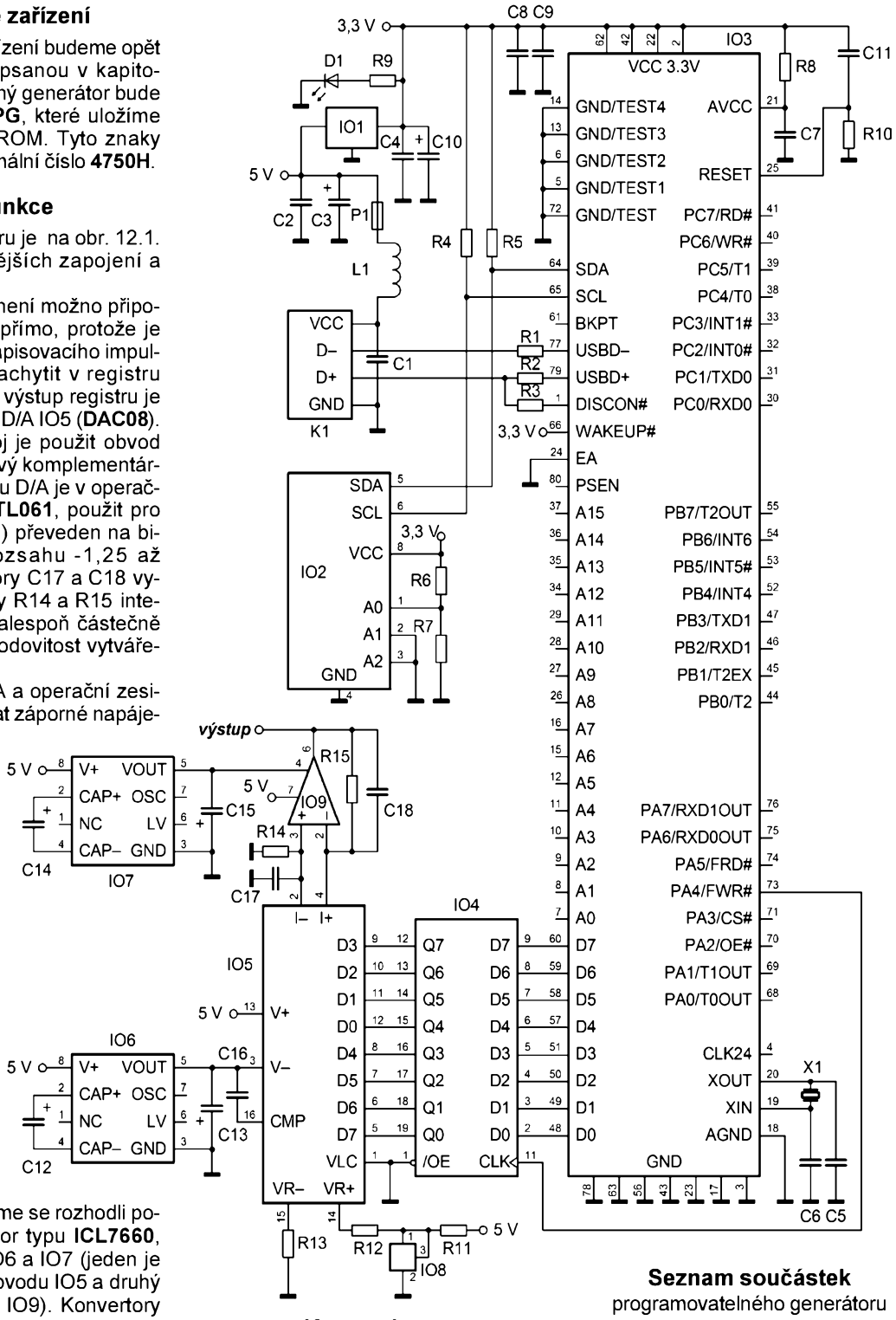
Pro identifikaci zařízení budeme opět používat metodu popsanou v kapitole 10. Programovatelný generátor bude identifikován znaky **PG**, které uložíme do konfigurační E^{2}PROM. Tyto znaky představují hexadecimální číslo **4750H**.

Popis funkce

Schéma generátoru je na obr. 12.1. Opět vychází z dřívějších zapojení a z předchozí rozvahy.

Datovou sběrnici není možno připojit k převodníku D/A přímo, protože je aktivní jen po dobu zapisovacího impulsu. Údaj se musí zachytit v registru IO4 (**74HCT574**). Na výstup registru je už připojen převodník D/A IO5 (**DAC08**). Jako referenční zdroj je použit obvod IO8 (**TL431**). Proudový komplementární výstup z převodníku D/A je v operačním zesilovači IO9 (**TL061**, použit pro svou malou spotřebu) převeden na bipolární výstup v rozsahu -1,25 až +1,25 V. Kondenzátory C17 a C18 vytváří spolu s rezistory R14 a R15 integrační články, které alespoň částečně omezí zákmity a schodovitost vytvářeného průběhu.

Pro převodník D/A a operační zesilovač bylo nutné získat záporné napáje-



Obr. 12.1. Schéma zapojení programovatelného generátoru (nejsou zakresleny některé blokovací kondenzátory)

cí napětí. Nakonec jsme se rozhodli použít klasický konvertor typu **ICL7660**, jedná se o obvody IO6 a IO7 (jeden je určen pro napájení obvodu IO5 a druhý pro napájení obvodu IO9). Konvertory poskytují naprázdno napětí přibližně -5 V. Vzhledem k saturačním napětím výstupního operačního zesilovače IO9 nelze zajistit příliš velký rozkmit výstupního signálu, který může být maximálně -1,25 až +1,25 V. Pro většinu praktických použití to však stačí.

Zajímavé je určitě i kmitočtové pásmo. Tento údaj je však závislý na zvoleném počtu vzorků. Pro 64 vzorků (nejkvalitnější signál) je pásmo generovaných kmitočtů 4,7 Hz až 6,25 kHz. Pokud požadujeme vyšší kmitočet, můžeme použít 32 vzorků v periodě a dostaneme pásmo 9,4 Hz až 12,5 kHz. Konečně pro 16 vzorků (již poměrně nekvalitní signál) dostáváme pásmo 18,7 Hz až 25 kHz.

Konstrukce

Stejně jako předchozí konstrukce je i programovatelný generátor zkonstruován ze součástek SMD i vývodových na desce s jednostrannými plošnými spoji. Pomocné propojky jsou realizovány krátkými drátky umístěnými na straně součástek i na straně spojů.

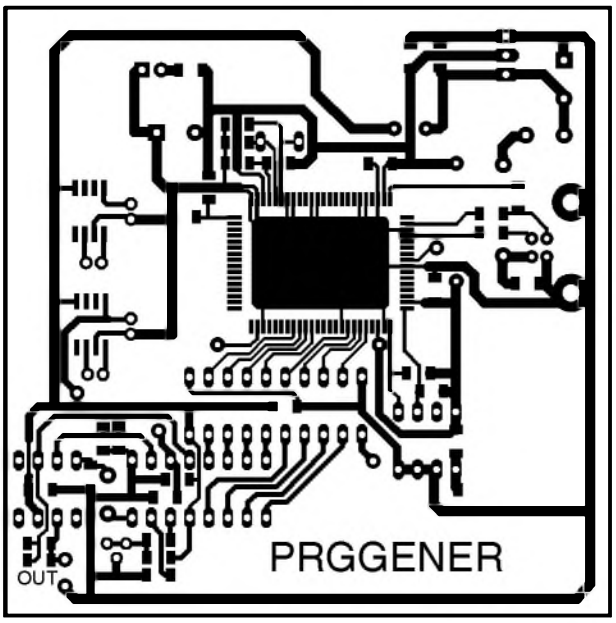
Obrazec spojů je na obr. 12.2, rozmístění součástek a propojek na obou stranách desky je na obr. 12.3 a obr. 12.4.

Před pájením součástek doporučuji nejdříve vyvrtat všechny díry a potom na straně spojů osadit součástky SMD. Pak se osadí vývodové součástky a propojky na straně součástek. Nakonec se zapojí tři propojky (izolovanými vodiči) na straně spojů.

Seznam součástek

programovatelného generátoru

R1, R2	27 Ω, SMD 1206
R3	1,5 kΩ, SMD 1206
R4 až R6	2,2 kΩ, SMD 1206
R7	0 Ω, SMD 1206
R8	1 Ω, SMD 1206
R9	470 Ω, SMD 1206
R10	10 kΩ, SMD 1206 SMD
R11	180 Ω, SMD 1206
R12, R13	2,0 kΩ/1 %, SMD 1206
R14, R15	1,0 kΩ/1 %, SMD 1206
C1, C2, C4, C7 až C9, C11, C19 až C23	100 nF/X7R, SMD 1206
C3, C10	470 µF/25 V, radiální
C5, C6	27 pF/NPO, SMD 1206
C12 až C15	10 µF/50 V, radiální
C16	68 pF/500 V, keramický



Obr. 12.2. Obrazec plošných spojů na desce programovatelného generátoru (měř.: 1 : 1, vodorovný rozměr desky je 80 mm)

C17, C18	1 nF/NPO, SMD 1206
L1	33 µH, tlumivka axiální
X1	krystal 12,000 MHz
D1	LED červená, 5 mm, 200 mcd
IO1	LM1084IT-03,3
IO2	24LC01-I/P
IO3	AN2131QC
IO4	74HCT574
IO5	DAC08
IO6, IO7	ICL7660, SMD
IO8	TL431
IO9	TL061
P1	RXE025, vratná pojistka
K1	konektor USB1X90B PCB

objímka precizní DIP8 pro IO2 (1 ks)
chladič DO1A pro IO1 (1 ks)
deska s plošnými spoji PRGGENER

Naprogramování konfigurační E²PROM

Přípravek **PRGGENER** připojíme s vymazanou pamětí IO2 k počítači a spustíme program **E2PROG.EXE**. Údaje zadáme podle obr. 12.5 a stiskneme tlačítko **Programuj**. Potom se zobrazí hláška požadující odpojit přípravek od počítače. Současně se program automaticky ukončí.

Abychom zabránili přepisu obsahu E²PROM, můžeme zapojit vývod WP (zábrana zápisu) na úroveň „log. 1". Postup byl uveden v kapitole 10 (viz obr. 10.7).

Program pro mikrořadič

Program pro mikrořadič musí především zajistit přenos vzorků generovaného signálu na datovou sběrnici požadovanou rychlostí. Z toho důvodu bylo definováno 16 rutin, takový počet totiž dovoluje velmi jemné přeladění. Rutiny jsou označeny jako **GEN00 až GEN15**.

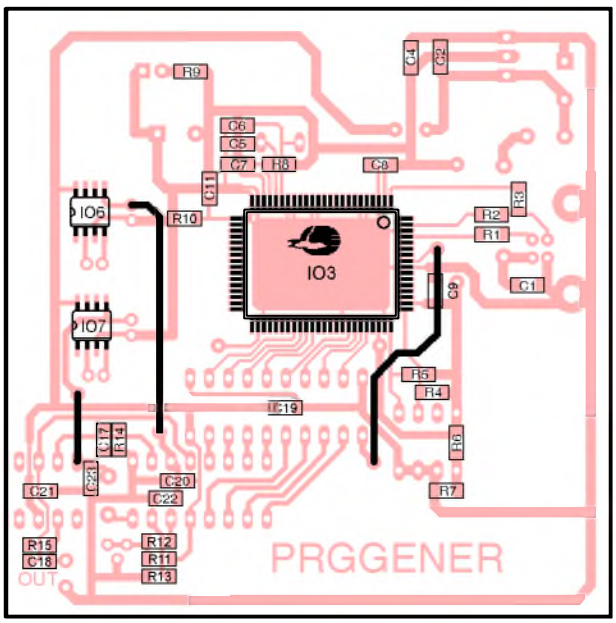
Základem přenosové funkce je makro **GENER**. Pro urychlení používá oba registry **DPTR**, které se mezi sebou musí přepínat. Pro urychlení přesunu ukazuje první **DPTR** na registr **AUTODATA** (využívá se automatické inkrementace) a druhý pak na **AUTOPRTL** (spodní bajt adresy autopointeru). Druhý ukazatel je nutný proto, aby se sekvence stále opakovala. Pomocí instrukcí **ANL** a **ORL** se zajišťuje procházení adres v rozsahu 16, 32 nebo 64 vzorků (takový počet vzorků má i generovaný signál). Pro pomalejší signály je třeba zařadit čekací funkci, k tomu je zavedeno makro **CEKEJ**.

Při inicializaci se nejdříve z proměnných **MOVXDEL** (určuje prodlevu provádění instrukce MOVX), **GENXX** (určuje jednu ze zvolených generovacích smyček), **ANDMASK** (AND maska pro GENER), **ORMASK** (OR maska pro **GENER**) a **DELL** a **DELH** (dolní a horní bajt pro čekací smyčku) nahrají žádané hodnoty. Povolí se funkce vývodu **FWR** (zapisovací impuls do registru IO4) a rychlý přenosový režim. Autopointer se nastaví na buffer hromadného endpointu 1.

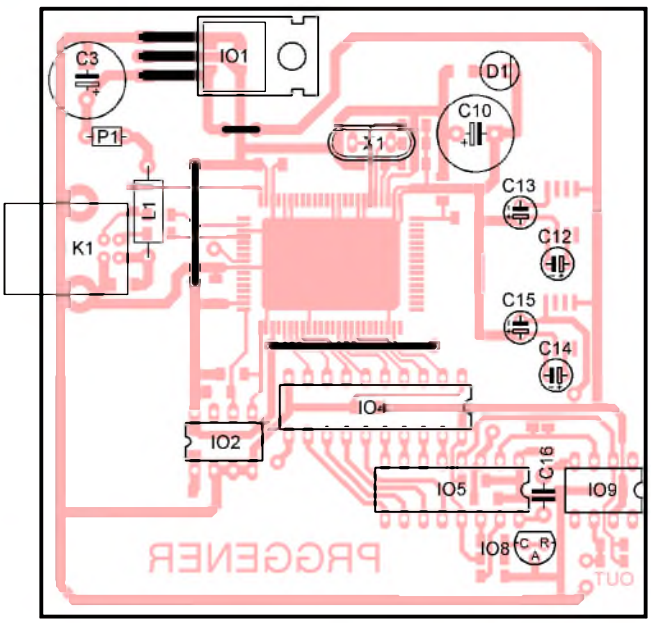
Vlastní generovací smyčka se spustí instrukcí **RET**. Úvodní inicializační část totiž v tabulce rutin **TABADR** vyhledala adresu požadované generovací funkce a uložila ji do zásobníku. Takže instrukce **RET** tuto adresu ze zásobníku vyjme a přejde na zvolenou rutinu. Tím se generování skutečně spustí.

Výpis programu PRGGENER.ASM pro mikrořadič je v tab. 12.1.

Obr. 12.5. Zadání identifikátoru programovatelného generátoru do E²PROM



Obr. 12.3. Rozmístění součástek SMD a drátových propojek na straně spojů na desce programovatelného generátoru



Obr. 12.4. Rozmístění součástek a drátových propojek na straně součástek na desce programovatelného generátoru

Tab. 12.1. Výpis programu PRGGENER.ASM

```
            $MODxx51
PORTACFG    EQU 7F93H
AUTOPTRH    EQU 7FE3H
AUTOPTRL    EQU 7FE4H
AUTODATA    EQU 7FE5H
DPS         EQU 86H
CKCON       EQU 8EH
FASTXFR     EQU 7FE2H
OUT07VAL    EQU 7FDFH
OUT1BUF     EQU 7E40H
            ;vlastní smyčka
GENER       MACRO
            MOVX A,@DPTR        ;odešli na DATA
            INC DPS             ;přepni DPTR
            MOVX A,@DPTR        ;čti AUTOPTRL
            ANL A,R0            ;omez na 40H
            ORL A,R1            ;až 7FH
            MOVX @DPTR,A        ;ulož AUTOPTRL
            INC DPS             ;přepni DPTR
ENDM
            ;čekací smyčka
CEKEJ       MACRO CEK
            MOV A,R2
            MOV R4,A
            MOV A,R3
            MOV R5,A
CEK:        MOV A,R4
            CLR C
            SUBB A,#1
            MOV R4,A
            MOV A,R5
            SUBB A,#0
            MOV R5,A
            ORL A,R4
            JNZ CEK
ENDM
START:      ;základní inicializace:
            MOV DPTR,#MOVXDEL
            MOVX A,@DPTR
            MOV CKCON,A         ;MOVX prodleva
            MOV DPTR,#PORTACFG
            MOV A,#00010000B
            MOVX @DPTR,A
            ;výběr smyčky:
            MOV DPTR,#GENXX
            MOVX A,@DPTR        ;A=číslo rutiny
            ADD A,ACC           ;A=A*2
            PUSH ACC            ;ulož A
            MOV DPTR,#TABADR
            MOVC A,@A+DPTR
            MOV R0,A            ;R0-horní bajt
            POP ACC             ;obnov A
            INC ACC             ;nižší bajt
            MOVC A,@A+DPTR
            PUSH ACC            ;ulož nižší bajt
            PUSH 0              ;ulož vyšší bajt
            MOV DPTR,#ANDMASK
            MOVX A,@DPTR
            MOV R0,A            ;maska pro ANL
            MOV DPTR,#ORMASK
            MOVX A,@DPTR
            MOV R1,A            ;maska pro ORL
            MOV DPTR,#DELL
            MOVX A,@DPTR
            MOV R2,A            ;R2=DELL
            MOV DPTR,#DELH
            MOVX A,@DPTR
            MOV R3,A            ;R3=DELH
            ;rychlý přenos. režim:
            MOV A,#00010000B    ;FWR aktivní
            MOVX @DPTR,A
            MOV DPTR,#OUT07VAL  ;povolí
            MOV A,#00000010B    ;endpoint 1
            MOVX @DPTR,A
            MOV DPTR,#FASTXFR   ;povolí
            MOV A,#01000000B    ;rychlý přenosový
            MOVX @DPTR,A        ;režim
            ;inic. před spuštěním:
            INC DPS             ;druhý DPTR
            MOV DPTR,#AUTOPTRL  ;ukazuje
            INC DPS             ;na AUTOPTRL
            MOV DPTR,#AUTOPTRH  ;nastav
            MOV A,#HIGH OUT1BUF ;AUTOPTR
            MOVX @DPTR,A
            MOV DPTR,#AUTOPTRL  ;na OUT1BUF
            MOV A,#LOW OUT1BUF
            MOVX @DPTR,A
            MOV DPTR,#AUTODATA  ;DPTR=AUTODATA
            ;spuštění:
            RET
            ;15,18,21,24,27,30,33,36
GEN00:      GENER
            SJMP GEN00
            ;16,19,22,25,28,31,34,37
GEN01:      GENER
            DB 0
            SJMP GEN01
            ;17,20,23,26,29,32,35,38
GEN02:      GENER
            DB 0,0
            SJMP GEN02
            ;19+13*R3:R2
GEN03:      GENER
            CEKEJ CEK03
            SJMP GEN03
            ;20+13*R3:R2
GEN04:      GENER
            CEKEJ CEK04
            DB 0
            SJMP GEN04
            ;21+13*R3:R2
GEN05:      GENER
            CEKEJ CEK05
            DB 0,0
            SJMP GEN05
            ;22+13*R3:R2
GEN06:      GENER
            CEKEJ CEK06
            DB 0,0,0
            SJMP GEN06
            ;23+13*R3:R2
GEN07:      GENER
            CEKEJ CEK07
            DB 0,0,0,0
            SJMP GEN07
            ;24+13*R3:R2
GEN08:      GENER
            CEKEJ CEK08
            DB 0,0,0,0,0
            SJMP GEN08
            ;25+13*R3:R2
GEN09:      GENER
            CEKEJ CEK09
            DB 0,0,0,0,0,0
            SJMP GEN09
            ;26+13*R3:R2
GEN10:      GENER
            CEKEJ CEK10
            DB 0,0,0,0,0,0,0
            SJMP GEN10
            ;27+13*R3:R2
GEN11:      GENER
            CEKEJ CEK11
            DB 0,0,0,0,0,0,0,0
            SJMP GEN11
            ;28+13*R3:R2
GEN12:      GENER
            CEKEJ CEK12
```

Tab. 12.1. Výpis programu PRGGENER.ASM (dokončení)

```
                DB 0,0,0,0,0,0,0,0,0
                SJMP GEN12
                ;29+13*R3:R2
GEN13:          GENER
                CEKEJ CEK13
                DB 0,0,0,0,0,0,0,0,0,0
                SJMP GEN13
                ;30+13*R3:R2
GEN14:          GENER
                CEKEJ CEK14
                DB 0,0,0,0,0,0,0,0,0,0,0
                SJMP GEN14
                ;31+13*R3:R2
GEN15:          GENER
                CEKEJ CEK15
                DB 0,0,0,0,0,0,0,0,0,0,0,0
                SJMP GEN15
TABADR:         DW GEN00
                DW GEN01
                DW GEN02
                DW GEN03
                DW GEN04
                DW GEN05
                DW GEN06
                DW GEN07
                DW GEN08
                DW GEN09
                DW GEN10
                DW GEN11
                DW GEN12
                DW GEN13
                DW GEN14
                DW GEN15
                ;komunikační proměnné:
ANDMASK         EQU $
ORMASK          EQU $+1
MOVXDEL         EQU $+2
DELH            EQU $+3
DELL            EQU $+4
GENXX           EQU $+5
                END
```

Ovládací program pro Windows

Ovládací program pro programovatelný generátor byl vybudován na základě programu z kapitoly 10.

Vzhledem k poměrně značné složitosti a omezenému rozsahu časopisu není možno program komentovat ještě mimo vlastní výpis, který je uveden dále. Zastavme se alespoň u funkce **Prehod**, která slouží k přehození bitů mezi datovou sběrnicí mikrořadiče a datovou sběrnicí D/A převodníku. Ze schématu na obr. 12.1 je patrné, že bity jsou vzájemně zpřeházeny. Toto řešení bylo zvoleno proto, aby byly plošné spoje co nejjednodušší.

Popišme si alespoň základní možnosti ovládání programovatelného generátoru (viz obr. 12.6):

- Předně je tu skupina pro volbu tvaru signálu (harmonický, obdélníkový a pilovitý).
- Dále je možné volit počet vzorků v periodě mezi 64, 32 a 16 vzorky. Čím větší počet vzorků zvolíme, tím je generovaný signál kvalitnější. Ovšem pro 64 vzorků je maximální kmitočet pouze 6,25 kHz, kdežto pro 16 vzorků již 25 kHz.
- Pro snadnější nastavování kmitočtu byla přidána skupina kmitočtové pásmo. Lze tedy volit kmitočet řádově v Hz nebo kHz. Vlastní nastavení kmitočtu probíhá skrolbarem. Nad ním je zobrazena nastavená hodnota kmitočtu.

Pohled na okno aplikace v akci je na obr. 12.6.

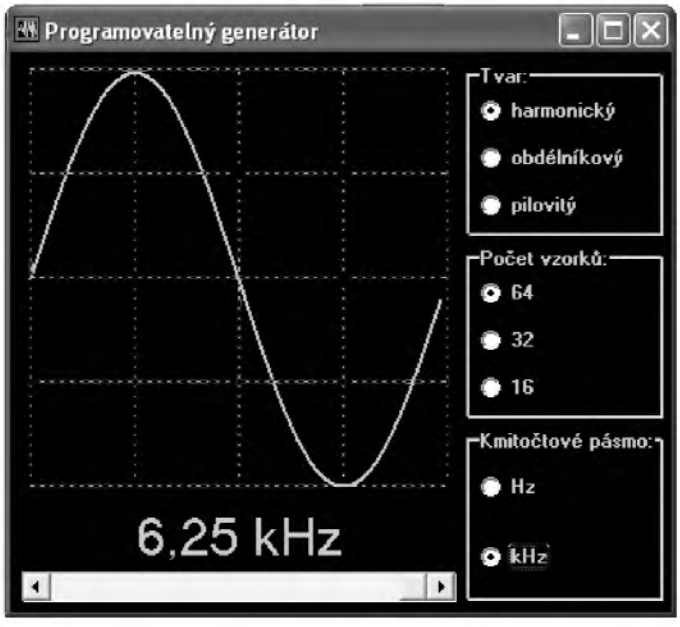
Následuje výpis ovládacích programů pro Windows:

PRGGENER.H:

```
#ifndef PRGGENERFH
#define PRGGENERFH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
#include "EZUSB.h"
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
const WORD Buffer=0x1e40;
//---------------------------------------------------------------------------
class THlForm : public TForm
{
__published:
  TRadioGroup *rgTvar;
  TRadioGroup *rgPocetVzorku;
  TScrollBar *sbKmitocet;
  TImage *imUkazka;
  TLabel *lbKmitocet;
  TRadioGroup *rgPasmo;
  void __fastcall FormCreate(
    TObject *Sender);
  void __fastcall FormDestroy(
    TObject *Sender);
  void __fastcall Aktualizuj(
    TObject *Sender);
  void __fastcall rgPasmoClick(
    TObject *Sender);
private: // User declarations
  //generátor:
  TAN2131 *AN2131;
  //adresy pro zápis dat:
  int ANDMASK,ORMASK,MOVXDEL,DELH,DELL,GENXX;
public:          // User declarations
  __fastcall THlForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
BYTE __fastcall Prehod(BYTE a);
//---------------------------------------------------------------------------
extern PACKAGE THlForm *HlForm;
//---------------------------------------------------------------------------
#endif
```

PRGGENER.CPP:

```
#include <vcl.h>
#include <math.h>
#include <inifiles.hpp>
#pragma hdrstop
#include "PRGGENERF.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlForm *HlForm;
//---------------------------------------------------------------------------
__fastcall THlForm::THlForm(
  TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall THlForm::FormCreate(
  TObject *Sender)
{
  TDeviceDescriptor dd;
  int c=0;
  AN2131=NULL;
  try{
    do{
      //zruš předchozí instanci:
      if(AN2131)
        delete AN2131;
      //vytvoření instance TAN2131:
      AN2131=new TAN2131(c++);
      //zjištění deskriptoru:
      dd=AN2131->DeviceDescriptor;
    }while(dd.bcdDevice!=0x4750); //test
    //zastaví procesor:
    AN2131->RAM[CPUCS]=0x01;
    //download programu
    //a výpočet adresy pro data portů:
    ANDMASK=AN2131->DownLoadFromRes(
      0,
      int(HInstance),
      "PRGGENER",
      "PROGRAM");
    ORMASK=ANDMASK+1;
    MOVXDEL=ANDMASK+2;
    DELH=ANDMASK+3;
    DELL=ANDMASK+4;
```



Obr. 12.6. Aplikace PROGGENER v akci

```
    GENXX=ANDMASK+5;
    //rozběhne procesor:
    AN2131->RAM[CPUCS]=0x00;
  }
  catch(Exception &E){
    Application->MessageBox(
      "Programovatelný generátor není připojen",
      "PRGGENER",
      MB_ICONHAND);
    //předčasné ukončení aplikace:
    Application->Terminate();
  }
  //inicializace:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\PRGGENER.INI");
  rgTvar->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Tvar",rgTvar->ItemIndex);
  rgPocetVzorku->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Pocet",rgPocetVzorku->ItemIndex);
  rgPasmo->ItemIndex=ini->ReadInteger(
    "NASTAVENI","Pasmo",rgPasmo->ItemIndex);
  //nastav rozsah scrollbaru:
  rgPasmoClick(NULL);
  sbKmitocet->Position=ini->ReadInteger(
    "NASTAVENI","Kmitocet",sbKmitocet->Position);
  delete ini;
  Aktualizuj(NULL);
}
//---------------------------------
void __fastcall THlForm::FormDestroy(
  TObject *Sender)
{
  //uložení parametrů:
  TIniFile *ini=new TIniFile(
    GetCurrentDir()+"\\PRGGENER.INI");
  ini->WriteInteger(
    "NASTAVENI","Tvar",rgTvar->ItemIndex);
  ini->WriteInteger(
    "NASTAVENI","Pocet",rgPocetVzorku->ItemIndex);
  ini->WriteInteger(
    "NASTAVENI","Pasmo",rgPasmo->ItemIndex);
  ini->WriteInteger(
    "NASTAVENI","Kmitocet",sbKmitocet->Position);
  delete ini;
  if(AN2131)
    delete AN2131;
}
//---------------------------------
void __fastcall THlForm::Aktualizuj(
  TObject *Sender)
{
  int a;
  //pole počtu vzorků:
  int pPocet[3]={64,32,16};
  //podklady pro výpočet kmitočtu:
  BYTE vMOVXDEL,vDELH,vDELL,vGENXX;
  //vzorky signálu a jejich počet:
  BYTE Vzorky[64];
  int Pocet=pPocet[rgPocetVzorku->ItemIndex];
  //násobicí faktor pro ukázku:
  int fakt=256/Pocet;
  //pole hodnot masek AND:
  BYTE pANDMASK[3]={0x7f,0x5f,0x4f};
  double f;
  int vDEL;
  //počet taktů:
  int pos=-sbKmitocet->Position+15;
  //bitmapa s rastrem:
  Graphics::TBitmap *b=new Graphics::TBitmap;
  b->Width=imUkazka->Width;
  b->Height=imUkazka->Height;
  b->Canvas->Brush->Color=clBlack;
  b->Canvas->Pen->Color=clLime;
  b->Canvas->FillRect(TRect(0,0,
    imUkazka->Width,imUkazka->Height));
  b->Canvas->Pen->Style=psDot;
  for(a=0;a<=256;a+=64){
    b->Canvas->MoveTo(4,258-a);
    b->Canvas->LineTo(4+256,258-a);
  }
  for(a=0;a<=256;a+=64){
    b->Canvas->MoveTo(4+a,4);
    b->Canvas->LineTo(4+a,256);
  }
  b->Canvas->Pen->Color=clLime;
  b->Canvas->Pen->Width=2;
  //generování vzorků posloupnosti:
  switch(rgTvar->ItemIndex){
    case 0:  //harmonický
      for(a=0;a<Pocet;a++)
        Vzorky[a]=127+127*sin(
          a/double(Pocet)*2*M_PI);
        break;
    case 1: //obdélníkový
      for(a=0;a<Pocet;a++)
        Vzorky[a]=(a<Pocet/2)?255:0;
        break;
    case 2: //pilovitý
      for(a=0;a<Pocet;a++)
        Vzorky[a]=255*a*256/double(Pocet);
        break;
  }
  //stanovení dělicího poměru:
  if(pos<=36){
    if((pos-15)%3==0){
      vGENXX=0;
      vMOVXDEL=(pos-15)/3;
    }
    else if((pos-16)%3==0){
      vGENXX=1;
      vMOVXDEL=(pos-16)/3;
    }
    else if((pos-17)%3==0){
      vGENXX=2;
      vMOVXDEL=(pos-17)/3;
    }
  }
  else{
    vDEL=(pos-19)/13;
    vDELL=vDEL&0xFF;
    vDELH=vDEL>>8;
    vGENXX=pos+3-19-vDEL*13;
    vMOVXDEL=0;
  }
  //výpočet kmitočtu:
  switch(vGENXX){
  case  0:f=6e6/double(Pocet*(15+3*vMOVXDEL));break;
  case  1:f=6e6/double(Pocet*(16+3*vMOVXDEL));break;
  case  2:f=6e6/double(Pocet*(17+3*vMOVXDEL));break;
  case  3:f=6e6/double(Pocet*(19+13*vDEL));break;
  case  4:f=6e6/double(Pocet*(20+13*vDEL));break;
  case  5:f=6e6/double(Pocet*(21+13*vDEL));break;
  case  6:f=6e6/double(Pocet*(22+13*vDEL));break;
  case  7:f=6e6/double(Pocet*(23+13*vDEL));break;
  case  8:f=6e6/double(Pocet*(24+13*vDEL));break;
  case  9:f=6e6/double(Pocet*(25+13*vDEL));break;
  case 10:f=6e6/double(Pocet*(26+13*vDEL));break;
  case 11:f=6e6/double(Pocet*(27+13*vDEL));break;
  case 12:f=6e6/double(Pocet*(28+13*vDEL));break;
  case 13:f=6e6/double(Pocet*(29+13*vDEL));break;
  case 14:f=6e6/double(Pocet*(30+13*vDEL));break;
  case 15:f=6e6/double(Pocet*(31+13*vDEL));break;
  }
  //uložení dat do procesoru:
  //zastaví procesor:
  AN2131->RAM[CPUCS]=0x01;
  //uloží vzorky:
  for(a=0;a<Pocet;a++){
    if(a==0)
      b->Canvas->MoveTo(4+a*fakt,258-Vzorky[a]);
    else
      b->Canvas->LineTo(4+a*fakt,258-Vzorky[a]);
    AN2131->RAM[a+Buffer]=Prehod(Vzorky[a]);
  }
  //řídicí proměnné:
  AN2131->RAM[MOVXDEL]=vMOVXDEL;
  AN2131->RAM[GENXX]=vGENXX;
  AN2131->RAM[DELH]=vDELH;
  AN2131->RAM[DELL]=vDELL;
  AN2131->RAM[ANDMASK]=
    pANDMASK[rgPocetVzorku->ItemIndex];
  AN2131->RAM[ORMASK]=0x40;
  //rozběhne procesor:
  AN2131->RAM[CPUCS]=0x00;
  //nakreslí ukázku:
  imUkazka->Canvas->Draw(0,0,b);
  delete b;
  //vypíše hodnotu kmitočtu:
  if(f>=1e3)
    lbKmitocet->Caption=FormatFloat("0.00 kHz",f/1e3);
  else
    lbKmitocet->Caption=FormatFloat("0.0 Hz",f);
}
//---------------------------------
void __fastcall THlForm::rgPasmoClick(
  TObject *Sender)
{
  //nutné před změnou nastavení skrolbaru:
  sbKmitocet->Min=-20000;sbKmitocet->Max=0;
  //podle pásma zadej min a max:
  switch(rgPasmo->ItemIndex){
    case 0:  //Hz
      sbKmitocet->Min=-20000;
      sbKmitocet->Max=-361;
      sbKmitocet->LargeChange=1000;break;
    case 1:  //kHz
      sbKmitocet->Min=-360;
      sbKmitocet->Max=0;
      sbKmitocet->LargeChange=10;break;
  }
  //definovaná pozice skrolbaru:
  sbKmitocet->Position=sbKmitocet->Max;
  if(Sender)
    Aktualizuj(NULL);
}
//---------------------------------
BYTE __fastcall Prehod(BYTE a)
{
  //prohodí bity do správného pořadí:
  return (a&0x80)>>7|(a&0x40)>>5|(a&0x20)>>3|
    (a&0x10)>>1|(a&0x0f)<<4;
}
```

Závěr

Účelem tohoto článku bylo seznámit čtenáře s populárním mikrořadičem **AN2131**. Jedná se o poměrně podrobný popis (ne však tak obsáhlý, jako původní katalogový list) s úvodními příklady. Kromě toho jsou zařazeny návody na výrobu užitečných zařízení. Tyto návody se mohou stát vodítkem pro realizaci vlastních typů zařízení.

Jako výchozí literaturu jsem používal [9] a velmi mi pomohla i [8]. Příklady uvedené v kapitolách 8 a 9 vycházejí z [8], ale vlastní provedení je odlišné. Pozornost jsem především věnoval podrobnějšími výkladu činnosti procesoru a prakticky zaměřeným konstrukcím.

Kde sehnat součástky a hotové moduly

Mikrořadiče **AN2131QC** lze v naší zemi zakoupit, se sháněním však mo-

hou být určité potíže (mnoho prodejců požaduje odebrat zboží v poměrně vysokém počtu kusů, což je značně nevýhodné). Z toho důvodu nabízím možnost koupě mikrořadičů, méně zkušenější amatéři (nechtějí pájet SMD) si mohou objednat dokonce hotové moduly. Přehled aktuálních cen najdete na:

www.mujweb.cz/efs-prodej/PE2005

Z této stránky je možné stáhnout doprovodné soubory (případně si je můžete objedat vypálené na CD-ROM, cena 30 Kč + poštovné).

Objednávat lze buď pomocí e-mailu:

matousekd@quick.cz

nebo písemně:

Ing. David Matoušek
Vysoká škola Polytechnická
Tolstého 16
JIHLAVA
586 01

Co obsahuje CD-ROM

Obsah CD-ROM je rozdělen do několika adresářů:

- ASM51 - **překladač ASM51.EXE a pomocné soubory,**
- DATASHEET - **katalogové listy nejdůležitějších součástek,**
- OVLADACE - **ovladač EZUSB,**
- PROGRAMY - **zdrojové texty i přeložená forma jednotlivých příkladů z kapitol 8 až 12,**
- SPOJE - **plošné spoje uvedených přípravků vytvořené v programu Eagle 4.11,**
- **UTILS** - pomocný program pro monitorování USB portů USBVIEW.EXE.

Literatura

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

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

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

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

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

[6] *Matoušek, D.*: USB prakticky s obvody FTDI. BEN - technická literatura, Praha 2003.

[7] *Matoušek, D.*: Udělejte si z PC v Delphi 1. díl. BEN - technická literatura, Praha 2003.

[8] *Kainka, B.*: Měření, řízení a regulace pomocí sběrnice USB. BEN - technická literatura, Praha 2003.

[9] Cypress: Katalogový list mikrořadiče AN2131.

Podrobnější informace o doporučené literatuře

Pro lepší představu připojuji krátké anotace knih doporučených v seznamu literatury.

Práce s mikrokontroléry ATMEL AT89C2051

V knize kromě podrobného popisu mikrořadiče AT89C2051 najdete konstrukci programátoru a vývojového kitu.

Dále je uvedeno poměrně velké množství přípravků, jejich používání je vysvětleno na příkladech (např. vícesegmentové displeje, vestavěný komparátor, převodníky A/D a D/A, budič M5451B7, TLC549, PCF8591, TL7705, LM331).

Na doprovodném CD-ROM naleznete zdrojové texty všech 23 publikovaných příkladů a klišé plošných spojů všech 16 realizovaných přípravků.

Práce s mikrokontroléry ATMEL AT89S8252

Kniha je zaměřena na popis mikrořadiče AT89S8252 včetně tří desítek jeho zajímavých aplikací. V knize je publikován vývojový kit kombinovaný s programátorem.

Pozornost je věnována především periferním obvodům SAA1064 (budič displeje), PCD3312 (DTMF), TDA8444 (8kanálový D/A převodník). Ukázáno je také ovládání disple aaaavvvvje LCD a klasické klávesnice k PC.

Je připojeno mnoho příkladů použití sériového portu v souvislosti s tímto mikrořadičem.

Práce s mikrokontroléry ATMEL AVR

Kniha je zaměřena na popis mikrořadičů AVR AT90S1200, AT90S2313, AT90S2343, AT90S4433, AT90S8515 a AT90S8535. Jsou v ní publikovány vývojové kity kombinované s programátorem.

V knize najdete příklady zaměřené na použití čítačů/časovačů, jednotek WatchDog, Input Capture, Output Compare, PWM. Najdete zde i příklady realizace převodníků A/D a D/A (např. MCP3002), použití LCD, SPI sběrnice, sériového kanálu. Je uveden i příklad použití teplotního čidla SMT160-30.

Udělejte si z PC, 1. díl

V knize je uveden popis stavby několika zařízení:

- impulsní generátor do 1 MHz,
- čítač do 16 MHz,
- programátor obvodů GAL,
- univerzální měřicí deska (převodníky A/D a D/A, číslicové vstupy/výstupy),
- programovatelný generátor do 100 kHz.

Udělejte si z PC, 2. díl

V knize je uveden popis stavby několika zařízení:

- testovací přípravky pro sériový a paralelní port,

- čítač do 50 MHz,
- napájecí zdroj řízený počítačem,
- zdokonalená univerzální měřicí deska (převodníky A/D a D/A, číslicové vstupy/výstupy),
- zdokonalený programovatelný generátor.

USB prakticky s obvody FTDI, 1. díl

Kniha se věnuje popisu a praktickému použití obvodu FT232BM, obvod pracuje jako konvertor signálů sběrnice USB na signály asynchronního sériového kanálu včetně linek modemu.

V knize je publikováno mnoho konstrukcí na bázi tohoto obvodu: programátor AT89C2051, vývojové kity pro AT90S2313, zdroj s proudovým omezením řízený počítačem, měřicí deska pro USB, konvertory USB na RS-232.

Udělejte si z PC v Delphi, 1. díl

Kniha ukazuje používání vývojového prostředí Delphi v souvislosti s paralelním a sériovým portem počítače. Nechybí příklad použití USB.

Nejdříve jsou uvedeny jednodušší příklady: LPTLCD (ovládání LCD displeje), použití teplotního čidla SMT160-30, COM4021 (8bitový vstupní port), COM1320 (levný 8bitový D/A převodník se sběrnicí I^2C).

Na závěr jsou publikovány pokročilejší konstrukce: LPTUNI (univerzální deska pro paralelní port), COMOSC (2kanálový osciloskop), USBMC (univerzální měřicí karta pro USB).

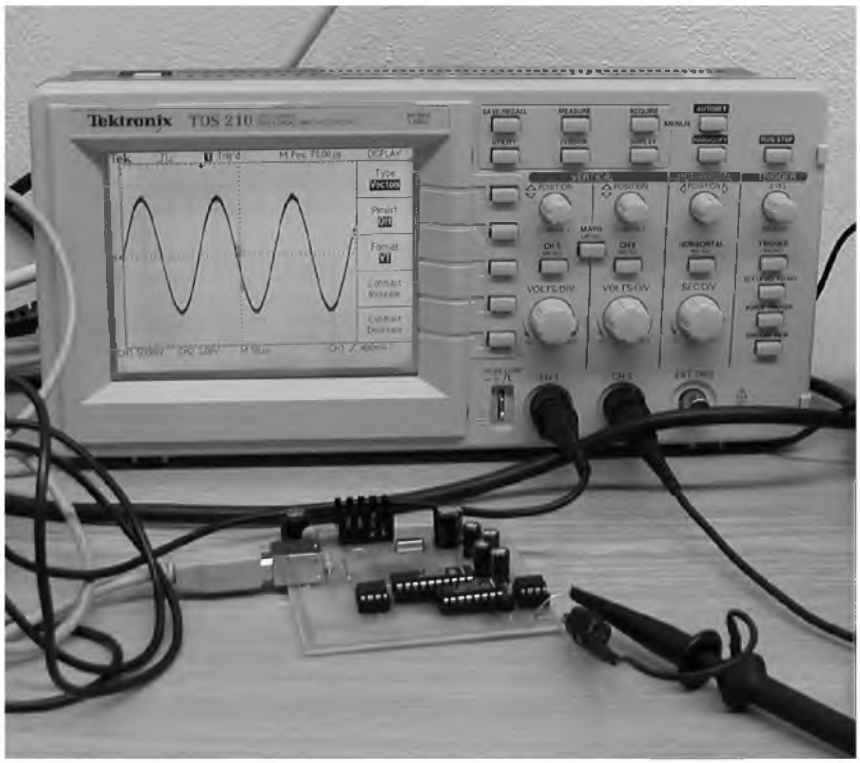
USB - měření, řízení a regulace pomocí sběrnice USB

Tato kniha zpřístupňuje USB i pro poloprofesionální aplikace. Nabízí jednoduché metody přístupu a účinnou podporu a ulehčuje tak čtenářům první praktický kontakt s USB.

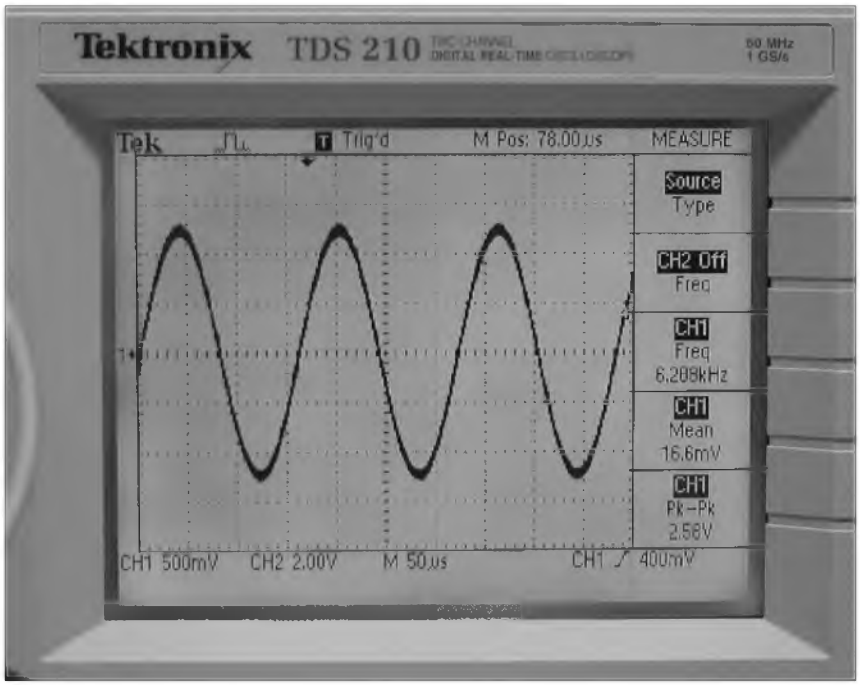
Krok za krokem jsou čtenáři seznamováni s vývojem hardware a software vhodného pro USB.

Na příkladech je uvedena práce s mikrořadiči CY7C63000 a AN2131.

Současně jsou položeny i nezbytné základy programování na straně PC. Jako programovací jazyky jsou používány Visual Basic a Delphi.



Měření sinusového signálu na výstupu programovatelného generátoru, popisovaného na straně 33



Detail průběhu sinusového signálu na výstupu programovatelného generátoru. Je vidět schodovitý charakter signálu, který by bylo možné dokonale vyhladit vhodným filtrem typu dolní propust