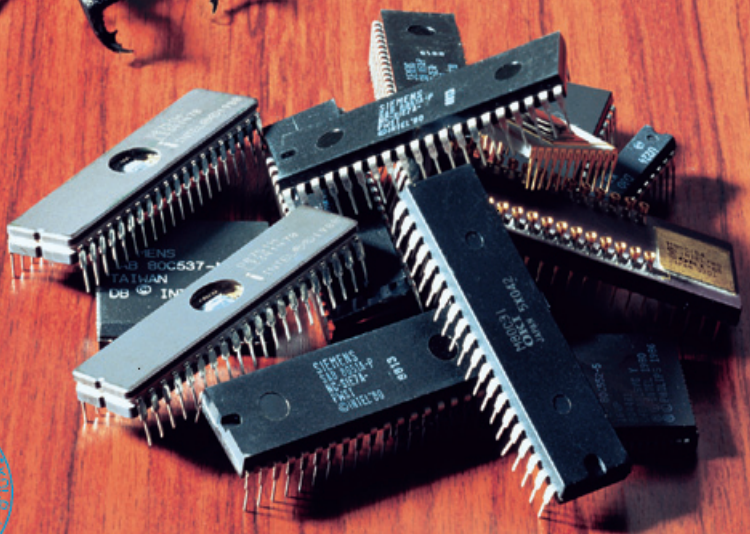
Petr Skalický

Mikroprocesory řady 8051



2. rozšířené vydání

426410

Příručka je určena především studentům a začátečníkům, kteří se rozhodli proniknout alespoň na pokraj problematiky monolitických mikropočítačů řady 8051. Pomocí této příručky se naučíte nejen programování v jazyce symbolických adres, ale zcela jistě pochopíte základy práce s těmito mikroprocesory.

Petr Skalický

MIKROPROCESORY ŘADY 8051

2. rozšířené vydání

Nakladatelství BEN - technická literatura, Věšínova 5, Praha 10

Petr Skalický: Mikroprocesory řady 8051
BEN - technická literatura, Praha 1998
2. rozšířené vydání
ISBN 80-86056-39-2 (tištěná kniha)
ISBN 978-80-7300-452-1 (elektronická kniha v PDF)

426410

OBSAH

Pár slov o autorovi

Doc. Ing. Petr Skalický, CSc.

Od ukončení svých studií v roce 1976 pracuje na katedře radioelektroniky elektrotechnické fakulty ČVUT, kde se již dlouhá léta věnuje výuce číslicové a mikroprocesorové techniky. Specializuje se na problematiku zpracování signálů, signálové procesory a další prostředky k realizaci číslicových obvodů. Ve volném čase navrhuje mikroprocesorové systémy včetně jejich programového vybavení, které jsou označeny značkou *PS-software*. Jedná se především o zakázkové řídicí systémy s jednočipovými procesory řady 8051 a systémy pro dlouhodobé střádání dat.

Tel.: 2 2435 2241

pmail: SKALICKY@felnet.feld.cvut.cz

Z vývoje firmy *PS-software* můžete v současnosti získat univerzální řídicí a vývojové systémy MIK552 a MIK537.

Bližší informace vám podá sám autor. Distribuci těchto systémů zajišťuje firma MERRET s.r.o., která mimo jiné prodává **panelové měřicí přístroje** (voltmetry, ampérmetry, teploměry, vlhkoměry, monitory procesů, wattmetry, integrátory, apod.)

MERRET s.r.o., Vodňanská 675/30, 198 00 Praha 9, www.merret.cz

Úvod

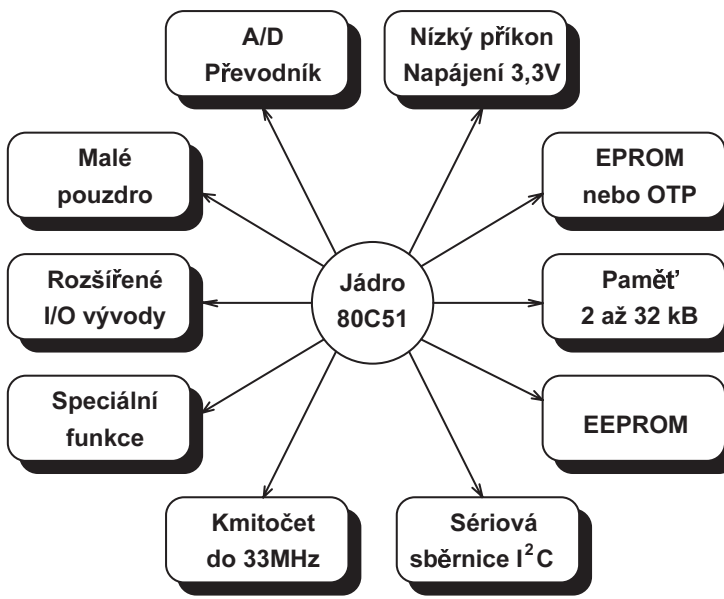
Jednočipové mikropočítače dnes najdeme v mnoha zařízeních a přístrojích, se kterými se denně běžně setkáváme. Mikropočítače řídí kancelářská zařízení jako jsou digitální telefony, faxy, telefonní ústředny, kopírky a tiskárny. V domácí a spotřební elektronice jako jsou rádia, televize, videa, CD přehrávače a zesilovače, ale i váhy, mikrovlnné trouby a regulátory topení si ani jejich existenci neuvědomujeme. Každé moderní zařízení z měřicí, automatizační a regulační techniky si lze dnes bez mikropočítače těžko představit. Proto je znalost návrhu a vývoje jednoúčelových, mikroprocesorem řízených, aplikací v současnosti velice důležitá.

V méně náročném mikropočítačovém řízení a zpracování signálů, jsou dnešním standardem jednočipové osmibitové mikropočítače od firem Motorola 68HC05 nebo 68HC11, Intel 8051 a řada jeho variant od různých výrobců (Philips, Siemens, Atmel, Dallas) nebo Zilog Z8, ale i modernizovaná řada Z80. S ohledem na historii této země se nejvíce u nás rozšířilo použití procesorů od firmy Intel, která z celosvětového hlediska nepředstavuje největšího producenta jednočipových procesorů. Procesory od firmy Motorola, které zaujímají asi třetinu světové produkce, se u nás prosazují jen pomalu.

Tento text vznikl pro potřebu výuky v předmětu „Elektronické počítače“ na střední průmyslové škole elektrotechnické s touto specializací. Ačkoliv jej nelze považovat za vyčerpávající publikaci v řadě procesorů typu 8051, přináší ucelený pohled na jádro těchto procesorů. Další část obecně popisuje typy periferií, které výrobci integrují do nových procesorů nejen z této řady. Stručný popis některých zvláště zajímavých a perspektivních typů byl nahrazen, popisem prvního procesoru z nastupující řady MCS251, které se liší architekturou, ale i výrazně vyšším výkonem. Uvedený procesor je však v jednom z módů plně kompatibilní s procesory řady 8051. Podrobný popis instrukčního souboru s řadou praktických příkladů přináší čtenáři možnost zvládnout programování v jazyce symbolických adres.

1. Mikroprocesory řady 8051

Mikroprocesor 8051 pochází z roku 1980 a je vývojově procesorem relativně starým. U návrhářů však dosáhl takové obliby, že i v současné době se řada výrobců orientuje na výrobu procesorů s jádrem procesoru 8051, které je rozšířeno o další periferie. Například firma Philips vyrábí 24 různých typů těchto procesorů mající společné jádro *obr. 1*, ke kterému jsou připojeny některé z uvedených periferií jako je: paměť programu o velikosti 2 kB až 32 kB, kterou lze programovat jenom jednou (provedení OTP) nebo několikrát (provedení EPROM), paměť EEPROM pro uchování konstant, rozšířená vnitřní paměť RAM na 256b, 8 nebo 10bitový A/D převodník obvykle s osmikanálovým analogovým multiplexerem, rozšířené vstupně/výstupní brány, komparační a záchytný systém nebo dvoudrátová přístrojová sběrnice I²C. Výrobci nabízejí procesory od základního hodinového kmitočtu 12 MHz až po 33 MHz ve standardním pouzdře DIL, přes provedení PLCC až po malá pouzdra PQFP. Jiné firmy se orientují na procesory s rozšířenými aritmetickými operacemi (Siemens 80C537) nebo s odlišným vnitřním časováním (Dallas DS80C320), který má při standardním hodinovém kmitočtu až trojnásobný výkon. Sjednocujícím základem všech těchto rozmanitých typů procesorů je vlastní jádro procesoru 8051 (80C51), které si nyní stručně popíšeme.

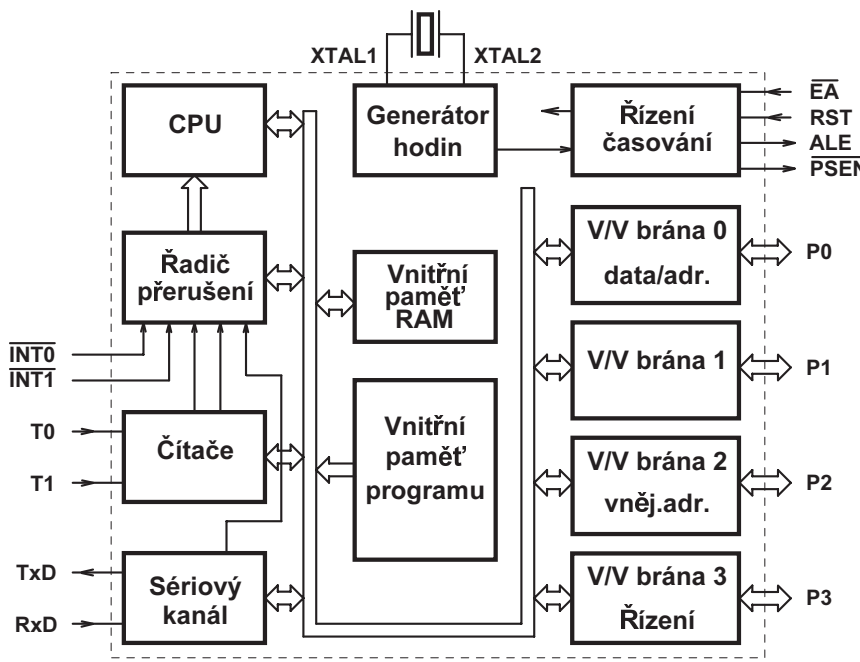


Obr. 1 Prvky rozšiřující vlastnosti základního jádra procesoru 8051

426410

1.1. Jádro procesoru 8051

Mikroprocesor 8051 je 8bitový jednočipový mikroprocesor s harwardskou strukturou, u které je oddělena programová a datová paměť. Procesor, jehož vnitřní struktura je blokově zobrazena na *obr. 2*, je schopen samostatné činnosti po připojení vnějšího piezokeramického rezonátoru („krystalu“) na vývody XTAL1 a XTAL2 a jednoho napájecího napětí 5 V. Na čipu procesoru je umístěna vlastní procesorová jednotka CPU, která je vnitřní společnou sběrnicí propojena s pamětí programu ROM nebo EPROM o kapacitě 4 kB (existují i typy bez této paměti), s pamětí RAM o kapacitě 128 bytů a se čtyřmi vstupně/výstupními branami P0 až P3, které zajišťují styk procesoru s vnějšími periferiemi. Nechceme-li využívat procesor v jednočipové konfiguraci, případně nemůžeme-li (procesor bez vnitřní ROM/ EPROM), jsou z procesoru vyvedeny řídicí signály pro správu vnější programové ($\overline{PSEN}$) nebo datové ($\overline{WR}$,$\overline{RD}$) paměti, z nichž každá může mít až 64 kB. Pro snadnější styk s periferiemi je procesor vybaven řadičem přerušení, který zpracovává 5 zdrojů přerušení (2 externí, od každého ze dvou časovačů a od sériového kanálu). Jednotlivá přerušení mají definovanou prioritu na každé ze dvou volitelných úrovní priority viz *tabulka D2*. Čítače, které usnadňují realizaci časování, jsou 16bitové s hodinovým signálem odvozeným z interního

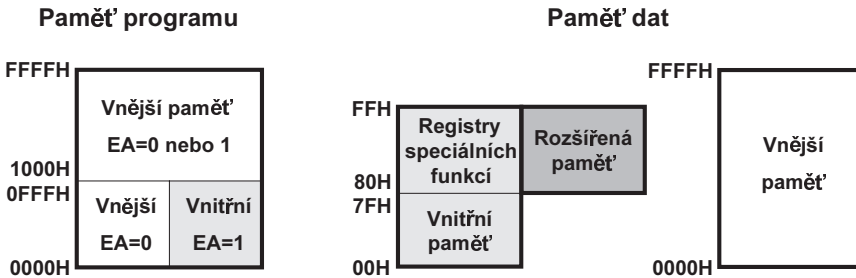


Obr. 2 Vnitřní bloková struktura procesoru 8051

generátoru hodin nebo z vnějších vstupů T0 nebo T1 *obr. 7* až *9*. Pro snazší sériový styk s nadřízenými počítači nebo jinými spolupracujícími procesory je vybaven duplexním (obousměrným) sériovým kanálem. Procesor je dále vybaven Booleovským procesorem, který umožňuje pracovat s jednotlivými bity vnitřní paměti RAM i interních periferií. Pro základní hodinový kmitočet 12 MHz trvají instrukce 1 μs nebo 2 μs s tím, že nejdelší jsou instrukce násobení a dělení, které trvají 4 μs.

1.1.1. Organizace paměti

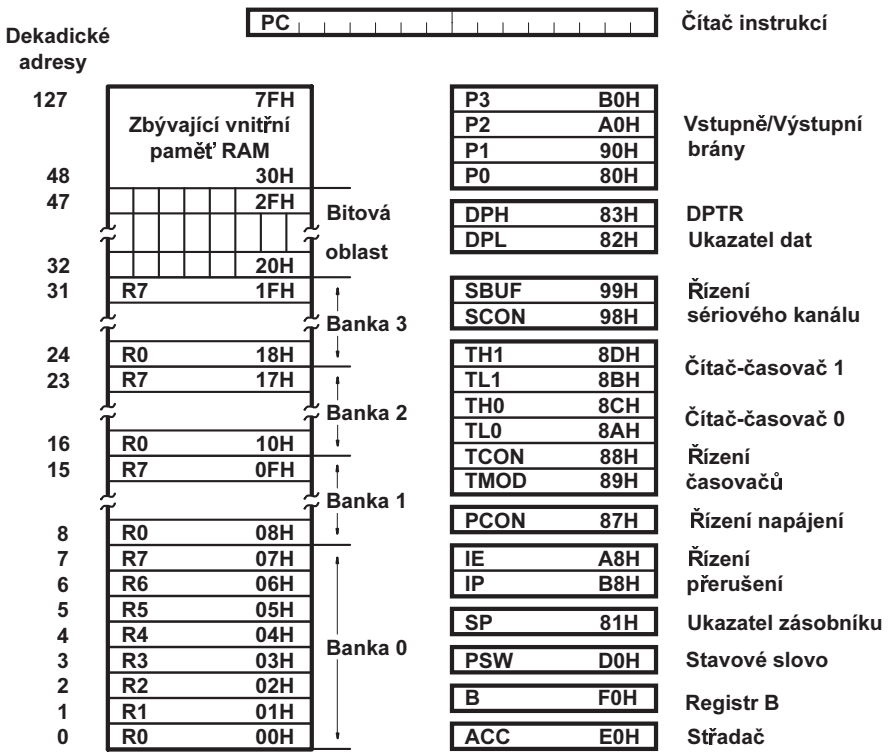
Mikroprocesor 8051 má oddělené adresové prostory programu a dat, které jsou přístupné různými instrukcemi. Paměťový prostor můžeme dále dělit na vnitřní, umístěný na čipu, a vnější, který lze v případě potřeby vytvořit s pomocí dalších součástek. Proti překrývání vnitřního (4 kB) a vnějšího (64 kB) programového adresového prostoru je procesor vybaven vstupem EA *obr. 2*. Je-li vstup EA=0, potom programová paměť je tvořena celou vnější pamětí, je-li vstup EA=1, potom instrukce v adresovém prostoru 000H až FFFH se čtou z vnitřní paměti ROM nebo EPROM a mimo tento prostor (1000H až FFFFH) ze zbývajících 60 kB vnější paměti programu *obr. 3*. Překrývání vnitřního a vnějšího datového adresového prostoru je odstraněno tím, že přístup do každého prostoru je realizován



Obr. 3 Struktura paměťového prostoru 8051 a jeho nástupců (tmavá barva)

pomocí rozdílných instrukcí. Na *obr. 4* je základní rozdělení vnitřní datové paměti včetně umístění speciálních registrů. Vnitřní datová paměť RAM 128 bytů je tvořena čtyřmi bankami (0,1,2,3) po osmi registrech R0, R1, až R7 (adresy 00H - - 1FH), za kterými je vyhrazeno 16 bytů (adresy 20H-2FH) pro tzv. **bitovou oblast**. Jednotlivé bity paměťových míst počínaje nejnižší adresou (20H) a nejnižším bitem (b0 - adresa bitu 00H) jsou vzestupně přímo adresovatelné. Přímo adresovatelných bitů v této oblasti je 128 (posledním bitem s adresou 127 (7FH) je bit b7 na adrese 2FH). Dalších 128 adres (od 80H do FFH) se využívá k adresování

některých významných bitů příslušejících speciálním registrům. Zbývající datová paměť RAM počínaje adresou 30H a konče adresou 7FH je uživateli volně přístupná pro přímé i nepřímé adresování.



Obr. 4 Rozdělení vnitřní paměti RAM procesoru 8051

Je-li procesor vybaven rozšířenou vnitřní datovou pamětí v adresovém prostoru 80H až FFH, potom je tato paměť s ohledem na speciální registry přístupná pouze pomocí nepřímého adresování. Vnější datová paměť RAM s kapacitou až 64 kB je přístupná přes 16bitový pomocný ukazatel datové paměti DPTR. Určitou nevýhodou je to, že DPTR může být pouze naplněn konkrétní adresou paměťového místa nebo inkrementován (DPTR může být zmenšován pouze odečítáním hodnoty od jeho dílčích částí tj. DPH a DPL). Vnější datová paměť může být přístupná i s pomocí registrů R0 a R1 příslušné aktivní banky, které se využívají k 8bitovému nepřímému adresování. Zapsáním 8bitové hodnoty do výstupní brány P2, která představuje horní část adresy při adresování vnější paměti, vybereme

jeden z 256 bloků paměti RAM o 256 bytech. Nepřímým adresováním pomocí registrů R0 a R1 potom můžeme zapsat nebo přečíst vybraný byte ve zvoleném bloku. Oblast speciálních funkcí (SFR) je tvořena 21 registry, které leží v adresovém prostoru 128 (80H) až 255 (FFH). To znamená za vnitřní pamětí RAM nebo ve stejném adresovém prostoru jako leží rozšířená datová paměť RAM u nástupců procesoru 8051. Z tohoto důvodu jsou speciální registry přístupné pouze pomocí přímého adresování bytů nebo případně i bitů.

Bitový (booleovský) procesor

V procesorech 8051 je integrován bitový procesor, který má vlastní soubor instrukcí a střadač tvořený příznakem přenosu, pracující nad bitově adresovatelnou částí vnitřní paměti RAM procesoru (128 bitů) a bity adresovatelných speciálních registrů nebo vstupně/výstupních registrů (128 bitů). Instrukční soubor bitového procesoru umožňuje nastavení, nulování a inverzi bitu. Přesun hodnoty bitu do jiného bitu je umožněn pouze s pomocí střadače. Výkon procesoru zvyšují instrukce větvení programu v závislosti na hodnotě daného bitu (JB bit, relativní adresa) nebo větvení podle nastaveného bitu s jeho následným vynulováním (JBC bit, relativní adresa). Střadač bitového procesoru může realizovat logický součin a logický součet s jednotlivými přímo adresovanými bity.

1.1.2. Registry speciálních funkcí

Všechny informace, důležité pro činnost mikroprocesoru a jeho periferních obvodů integrovaných na čipu procesoru, jako jsou čítače, sériový kanál, přerušovací systém a u nástupců 8051 záchytný a komparační systém, jsou soustředěny do souboru registrů tzv. registrů speciálních funkcí SFR. Nyní si postupně popíšeme jednotlivé registry a v případě registrů ovlivňujících činnost periferií i s popisem vlastností těchto obvodů.

A - **Střadač** je základní registr aritmetickologické jednotky, který vždy obsahuje jeden operand aritmetické nebo logické operace a do něhož se ukládá výsledek této operace. Protože registr dovoluje ve spolupráci s aritmetickologickou jednotkou postupné přičítání zpracovávaných čísel, nazývá se střadač nebo **akumulátor**. Zvláštností střadače u procesoru 8051 je to, že leží ve vnitřním paměťovém prostoru a je přístupný nejen běžnými instrukcemi, v kterých je označen A, ale i pomocí přímé adresy označované v mnemonice procesoru symbolickým názvem ACC.

B - **Registr** obsahuje jeden operand (druhý je umístěn ve střadači) pro instrukci násobení nebo dělení. Spolu se střadačem obsahují výsledek operací násobení a dělení. Nejsou-li tyto instrukce využívány, lze jej použít jako univerzální registr.

PSW - Stavové slovo mikroprocesoru se skládá z 8 bitů, z nichž je 7 významových. Umístění příznaků v PSW je zobrazeno na *obr. 5* a jejich význam je následující:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
C	AC	F0	RS1	RS0	OV	---	P	Adresa RAM = D0H
D7	D6	D5	D4	D3	D2	D1	D0	Bitová adresa (hex)

Obr. 5 Rozložení příznaků ve stavovém slově

♣ **C** - Přenos (Carry) je nastaven při aritmetické operaci, při které dochází k přenosu z osmého (b_7) do devátého (b_8) bitu a při některých instrukcích porovnání.

♣ **AC** - Pomocný (částečný) přenos (Auxiliary Carry) je nastaven, dojde-li při sčítání k přenosu mezi čtvrtým (b_3) a pátým (b_4) bitem střadače. Příznak využívá pouze instrukce dekadické korekce DAA, kterou aplikujeme na výsledek součtu dvou dekadických čísel vyjádřených v BCD kódu.

♣ **F0** - Uživatelský příznak F0 může být libovolně využíván programátorem k indikaci nějaké události (např. přetečení při výpočtu v aritmetice, identifikace vnější události atd.).

♣ **RS1, RS0** - Určují banku, její registry R0 až R7 budou používány (*obr. 4*) například k výpočtu atd. Jednotlivé bity RS1 a RS0 lze ovládat programově pomocí logických, přesunových nebo bitových operací. Procesor není vybaven instrukcí pro přepínání bank a po jeho vynulování je aktivní bankou banka 0 (RS1= RS0=0).

Tabulka 1

RS1	RS0	Banka	Adresy reg. R0,...,R7
0	0	0	00H až 07H
0	1	1	08H až 0FH
1	0	2	10H až 17H
1	1	3	18H až 1FH

♣ **OV** - Příznak přetečení (Overflow) indikuje přetečení při aritmetické operaci sčítání nebo odčítání, jestliže zpracovávaná čísla považujeme za čísla se znaménkem. Jedná se o případ, kdy součet dvou záporných čísel je kladný (došlo k přenosu mezi bity b_8 a b_7 a nedošlo k přenosu mezi bity b_7 a b_6) nebo součet dvou

kladných čísel je záporný (nedošlo k přenosu mezi bity b_8 a b_7 a došlo k přenosu mezi bity b_7 a b_6), kde bit b_7 představuje znaménko. Příznak je též využíván při operaci dělení k identifikaci dělení nulou a při instrukci násobení.

♣ **P** - Příznak parity (Parity Flag) indikuje lichou paritu střadače. Je-li ve střadači lichý počet jedniček, potom příznak parity je nastaven P=1. Příznak je aktualizován po každé instrukci.

SP - **Ukazatel zásobníku** (Stack Pointer) je 8bitový registr, který je na rozdíl od většiny jiných procesorů při plnění zásobníku inkrementován (hodnota ukazatele je zvětšována o jedničku). Vlastní zásobník je umístěn ve vnitřní datové paměti RAM a může být umístěn kdekoliv v této paměti tj. i v rozšířené části vnitřní datové paměti (adresy 7FH až FFH u nástupců 8051), která je přístupná pouze pomocí nepřímého adresování. Po vynulování procesoru signálem *reset* je nastaven ukazatel na hodnotu SP=07H. Ve většině aplikací musí být pomocí odpovídající instrukce přestaven na jinou hodnotu.

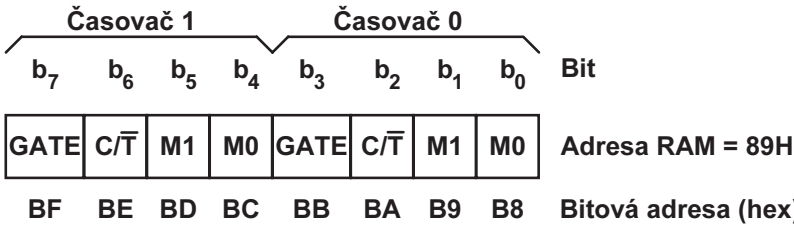
DPL, DPH - Registry DPL a DPH tvoří nižší a vyšší 8bitovou slabiku 16bitového ukazatele **DPTR**, který slouží k 16bitovému nepřímému adresování vnější datové paměti RAM nebo paměti programu. Oba registry lze využívat jako paměťová místa.

PC - Čítač instrukcí (Program Counter) je 16bitový čítač instrukcí, který není přímo programově přístupný.

1.1.3. Čítače/časovače

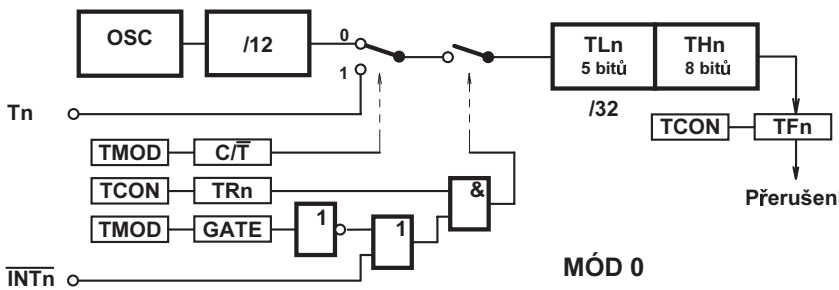
Mikroprocesor 8051 obsahuje dva 16bitové čítače, jejichž obsah je přístupný pomocí paměťově mapovaných registrů **TH0**,**TL0** (čítač 0) a **TH1**,**TL1** (čítač 1), které určují vyšší a nižší slabiku (8 bitů) příslušného čítače. Hodinový synchronizační signál čítačů může být odvozen z oscilátoru procesoru nebo z vnějšího zdroje, přivedeného na vývody procesoru T0 a T1. Je-li zdrojem signálu vnitřní oscilátor procesoru, potom čítač je ve funkci **časovače** a přičítá jedničku za každý strojový cyklus, který je tvořen 12 periodami oscilátoru. Ve funkci **čítače vnějších událostí** se obsah příslušného čítače (registru) zvýší o jedničku po přechodu signálu Tn z 1→0. Vstupy T0 a T1 se testují během stavu S5P2 každého strojového cyklu *obr. 26*. Zjistí-li se v jednom cyklu úroveň log. 1 a v příštím log. 0 přičte se k obsahu čítače jednička. Nová hodnota je v čítači nastavena v době S3P1 následujícího cyklu za cyklem, ve kterém byla zjištěna změna. Protože zjištění změny na vstupech Tn trvá 2 strojové cykly (24 period oscilátoru), je maximální čítaný kmitočet vnějšího signálu 1/24 kmitočtu oscilátoru mikropočítače. Logická úroveň čítaného signálu musí zůstat nezměněna vždy alespoň 1 celý strojový cyklus. Konfiguraci čítače/časovače 0 a 1 zajišťujeme naprogramováním registru TMOD. Vlastní čítače se programově spouští nebo zastavují nastavením nebo vynulováním bitu TRn v registru TCON.

TMOD - Registr módu časovačů / čítačů (Timer/Counter mode control) se skládá ze dvou čtveřic bitů *obr. 6* příslušejících každému ze dvou čítačů/časovačů. Význam jednotlivých bitů je následující:



Obr. 6 Rozložení bitů v registru TMOD

♣ **GATE** - Řízení hradlování. Je-li GATE=1, potom čítač/časovač n (n=0,1) je aktivován (čítá) při vstupu $\overline{INTn} = 1$ a TRn=1, kde TRn je bit z registru TCON. V tomto režimu je činnost čítače ovlivňována nejenom programově pomocí bitu TRn, ale zároveň i pomocí vnějšího signálu přivedeného na vstup $\overline{INTn}$ *obr. 7*. Je-li GATE=0, potom čítač/časovač n je aktivní pro TRn=1 (čítač je řízen pouze programově).



Obr. 7 Čítač/časovač n v módu 0 - 13bitový čítač

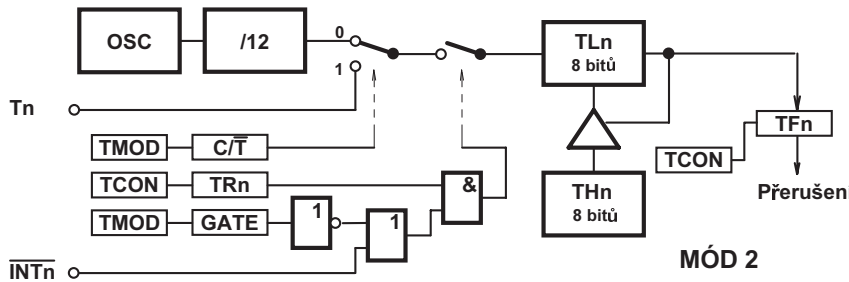
♣ **C / $\overline{T}$** - Volba čítač/časovač. Bit rozhoduje o zdroji hodinových impulzů, které bude čítač počítat. Je-li C / $\overline{T} = 0$, potom se jedná o režim časovače a hodinový signál je vytvořen z hodinového synchronizačního signálu procesoru vydělením hodnotou 12. Je-li C / $\overline{T} = 1$, potom se jedná o režim čítače vnějších událostí na vývodu Tn.

♣ **M1,M0** - Volba jednoho ze čtyř módů čítače/časovače.

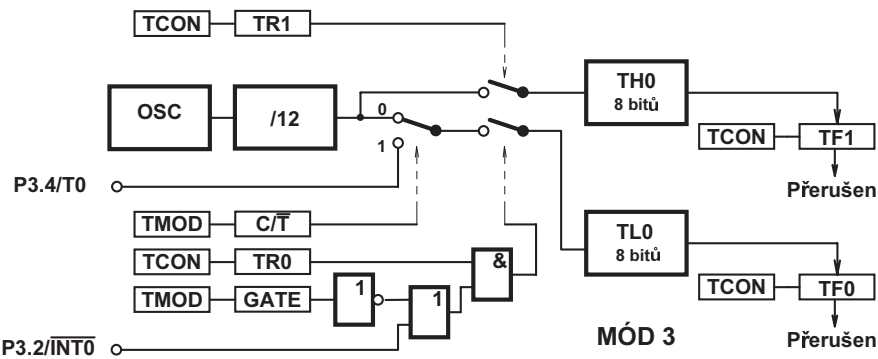
• **Mód 0** - M0=M1=0. Režim čítače/časovače je shodný s režimem čítače na procesoru 8048. Osmibitový čítač THn čítá hodinové impulzy vydělené 5bitovým předděličem tvořeným spodními bity čítače TLn *obr. 7*. Oba čítače čítají vzestupně a vytváří 13bitový čítač, který při přetečení (přechodu ze samých jedniček na samé nuly) nastaví příslušný příznakový bit TFn v registru TCON, které mohou být využívány jako zdroje přerušení procesoru. Vstup synchronizačního signálu do časovače je povolen tehdy, je-li TRn=1 (čítač je spuštěn) a současně s tím je GATE=0 nebo $\overline{INT1}=1$. Časovač 1 může být též využíván ke generování přenosové rychlosti sériového kanálu v módu 1 a 3.

• **Mód 1** - M0=1, M1=0. Mód 1 je shodný jako mód 0 s tím rozdílem, že čítače THn a TLn jsou 8bitové a vytváří tak 16bitový čítač. Dojde-li k přechodu ze samých jedniček na samé nuly, nastaví se příznak TFn.

• **Mód 2** - M0=0, M1=1. V módu 2 pracuje registr TLn jako čítač s obvodovým přednastavením *obr. 8*. Po přetečení je čítač TLn nastaven na hodnotu THn. Programové nastavení nové hodnoty v registru THn neovlivňuje současný stav čítače TLn.



Obr. 8 Čítač/časovač n v módu 2, 8bitový čítač s přednastavením



Obr. 9 Čítač/časovač 0 v módu 3, dva 8bitové čítače

• **Mód 3** - M0=M1=1. V předcházejících módech byla funkce časovače 0 a 1 shodná. V módu 3 dochází u čítače 0 k jeho dělení na dva samostatné 8bitové čítače TL0 a TH0. Čítač TL0 využívá standardní signály $C/\overline{T}$, GATE, TR0, $\overline{INT0}$ a TF0. Čítač TH0 pracuje ve funkci časovače a je ovládán pouze řídicím bitem TR1. Při přetečení nastavuje příznak TF1 *obr. 9*. Pracuje-li čítač 0 v módu 3, potom čítač 1 může pouze generovat přenosovou rychlost pro sériový kanál nebo být využit v aplikaci, která nevyužívá přerušení. Protože bit TR1 je využit pro řízení čítače 0, je zastavení nebo spuštění čítače 1 ovládáno jeho nastavením do módu 3 nebo zrušením módu 3.

TCON - Registr řízení čítače/časovače se skládá ze čtyř bitů příslušejících oběma časovačům a čtyř bitů patřících vstupům vnějšího přerušení. Na *obr. 10* je zobrazeno jejich umístění v registru TCON. Jednotlivé bity mají následující význam:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Adresa RAM = 88H
8F	8E	8D	8C	8B	8A	89	88	Bitová adresa (hex)

Obr. 10 Přiřazení bitů v registru řízení čítačů TCON

♣ **TF0, TF1** - Přetečení čítače/časovače. Bit je nastaven při přechodu časovače z maximální hodnoty do nuly a je automaticky vynulován při přechodu procesoru do odpovídajícího obslužného podprogramu přerušení.

♣ **TR0, TR1** - Spuštění čítače/časovače. Bit, který je ovládán programově, zajišťuje spuštění nebo zastavení příslušného čítače. Je-li GATE=1 v registru TMOD příslušného čítače, potom o přivedení nebo přerušení synchronizačního hodinového signálu do čítače rozhoduje vstupní signál $\overline{INRn}$.

♣ **IE0, IE1** - Přijetí vnějšího přerušení. Příslušný bit je nastaven při sestupné hraně nebo úrovni log. 0 na vstupu vnějšího přerušení $\overline{INRn}$ v závislosti na stavu konfiguračního bitu ITn. Po přechodu procesoru do obslužného podprogramu příslušného přerušení je bit automaticky vynulován.

♣ **IT0, IT1** - Konfigurace aktivace vnějšího přerušení. Je-li ITn=1, je žádost o vnější přerušení aktivována sestupnou hranou signálu na vstupu $\overline{INTn}$. Je-li ITn=0, je žádost aktivována úrovní log. 0 na vstupu $\overline{INTn}$. Je-li signál na vstupu $\overline{INTn}$ po dlouhou dobu v log. 0, může být přerušení vyvoláno i několikrát za sebou, pokud doba obslužného podprogramu přerušení je kratší než šířka impulzu v úrovni log. 0 na vstupu $\overline{INTn}$. Bity se nastavují a nulují programově.

1.1.4. Přerušení

Pro snazší komunikaci s vnějšími periferiemi je mikroprocesor 8051 vybaven přerušovacím systémem s pěti zdroji přerušení. U nástupců 8051 se počet zdrojů přerušení v závislosti na počtu vnitřních periferií zvětšuje a dosahuje až hodnoty 15. Vnější přerušení $\overline{INT0}$ a $\overline{INT1}$ mohou být vyvolána buď logickou úrovní (log. 0) nebo změnou logické úrovně (sestupnou hranou $1 \rightarrow 0$). Vznikne-li vnější přerušení, je nastaven příslušný příznak IE0, IE1, který je obvodově automaticky vynulován při vyvolání obslužného podprogramu. Přerušení od časovače 0 a 1 se vyvolávají nastavením příznaků TF0 a TF1, které indikují přetečení příslušného čítače. Vyvolá-li se přerušení od časovače, potom odpovídající příznak TFn je vynulován při přechodu do obslužného podprogramu. Přerušení od sériového kanálu se generuje logickým součtem příznaků RI a TI. Aby uživatel mohl zjistit, zda přerušení bylo generováno příznakem RI (příjem) nebo TI (vysílání), nejsou příznaky automaticky obvodově nulovány při přechodu do obslužného podprogramu. V obslužném programu je programátor „nucen“ nejprve stanovit příčinu přerušení (od RI nebo TI) a potom příslušný příznak programově vynulovat. V obslužném podprogramu pro sériový kanál tak zároveň programově rozhoduje o tom, která žádost (RI nebo TI) bude zpracována dříve a bude tak mít vyšší prioritu. Všechny příznaky, které generují přerušení, mohou být programově vynulovány nebo nastaveny. To znamená, že přerušení mohou být generována i programově nebo, vyžaduje-li to situace, mohou být nevyřízené žádosti o přerušení programově zrušeny. Každý ze zdrojů přerušení je možné individuálně povolit nebo zakázat nastavením nebo vynulováním příslušného bitu v registru speciálních funkcí IE.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
EA	---	---	ES	ET1	EX1	ET0	EX0	Adresa RAM = A8H
AF	---	---	AC	AB	AA	A9	A8	Bitová adresa (hex)

Obr. 11 Rozložení bitů povolení přerušení

IE - **Registr povolení přerušení** (Interrupt Enable) se skládá z 5 bitů příslušejících každému z pěti zdrojů přerušení a jednoho bitu zajišťujícího globální zákaz všech přerušení nebo povolení všech povolených přerušení označeného EA. Na *obr. 11* je zobrazeno umístění jednotlivých bitů v registru povolení přerušení včetně jejich přímých bitových adres, s jejichž pomocí mohou být individuálně ovlivněny bez nutné znalosti stavu ostatních bitů v registru. Význam jednotlivých bitů je následující:

♣ **EA** - Globální povolení přerušení (Enable all). Je-li EA=0, je celý přerušovací systém zablokován a nemůže být přijata jakákoliv žádost o přerušení. Je-li

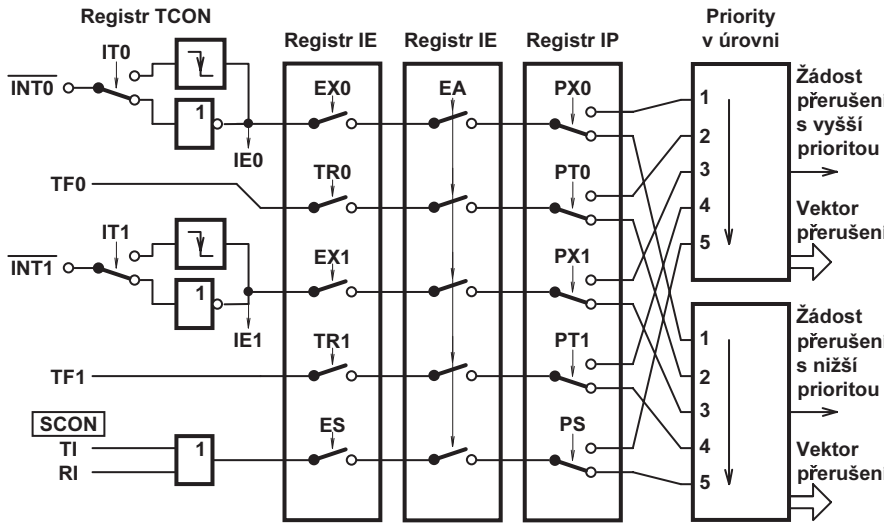
EA=1, potom mohou být přijaty žádosti těch přerušení, která mají nastavený (log. 1) svůj povolovací bit tj. ES, ET1, ... atd.

♣ **ES** - Povolení přerušení od sériového kanálu. Je-li ES=1 a zároveň EA=1, je přerušení od příjmu a vysílání sériového kanálu povoleno.

♣ **ET0, ET1** - Povolení přerušení od čítače/časovače 0,1. Je-li ETn=1 a zároveň EA=1, potom je povoleno přerušení způsobené přetečením čítače/časovače n.

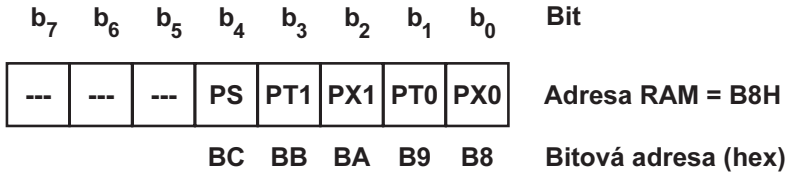
♣ **EX0, EX1** - Povolení vnějšího přerušení $\overline{INT0}$, $\overline{INT1}$. Je-li $\overline{EXn}$=1 a zároveň EA=1, potom je povoleno přijetí vnějšího přerušení ze vstupu $\overline{INTn}$. O tom, zda přerušení bude vyvoláno sestupnou hranou signálu na vstupu $\overline{INTn}$ nebo úrovní log. 0, rozhoduje bit ITn v registru TCON.

Pro využití přerušovacího systému mikroprocesoru je nezbytné znát přiřazení priorit jednotlivým přerušením. U procesoru 8051 se výrobce rozhodl pro pevně přidělené priority jednotlivým přerušením v každé ze dvou úrovní priority *obr. 12*. V rámci každé úrovně priority (nižší nebo vyšší) má nejvyšší prioritu vnější přerušení $\overline{INT0}$ (IE0) následované časovačem 0 (TF0), vnějším přerušením $\overline{INT1}$ (IE1), časovačem 1 (TF1) a sériovým kanálem (RI a TI), které má nejnižší prioritu. Potřebuje-li uživatel z nějakého důvodu přiřadit jednomu z přerušení nejvyšší prioritu, může mu přiřadit vyšší úroveň priority nastavením odpovídajícího bitu v registru **IP** - **registru priority přerušení** do log. 1. Na *obr. 13* je zobrazeno umístění bitů úrovně priority jednotlivých zdrojů přerušení včetně jejich bitových adres, na které se můžeme odkazovat uvedenými symbolickými názvy, kde PS přísluší sériovému kanálu, PT0 a PT1 časovačům 0 a 1 a PX0, PX1 vnějším vstupům přerušení



Obr. 12 Přerušovací systém procesoru 8051

$\overline{INT0}$, $\overline{INT1}$. Přiřazením vyšší úrovně priority dosáhneme toho, že i přerušení s nižší úrovní priority bude přerušeno v důsledku žádosti přerušení s vyšší úrovní. Je-li již jedno přerušení vyvoláno (probíhá obslužný podprogram) nemůže být přerušeno přerušením s vyšší prioritou v dané úrovni priority a musí počkat až na jeho dokončení. Potom budou podle vzájemné priority v dané úrovni postupně zpracovány další žádosti o přerušení. Z toho vyplývá, že přerušení s vyšší úrovní priority již nemůže být přerušeno. Otázka priority přerušení se stává důležitou v případech současného přijetí dvou a více žádostí o přerušení a u systémů využívajících velkého počtu přerušení. I když volba jednotlivých priorit přerušení v dané úrovni vychází ze zkušeností z mnoha realizací mikroprocesorových systémů, nemusí v případě využití mnoha přerušení vyhovovat pouze dvouúrovňovému členění. Proto u některých výkonných nástupců (Siemens 80C537) je přerušovací systém členěn do čtyř úrovní priority, který uživateli poskytuje přece jenom větší možnosti změn v prioritách jednotlivých přerušení.



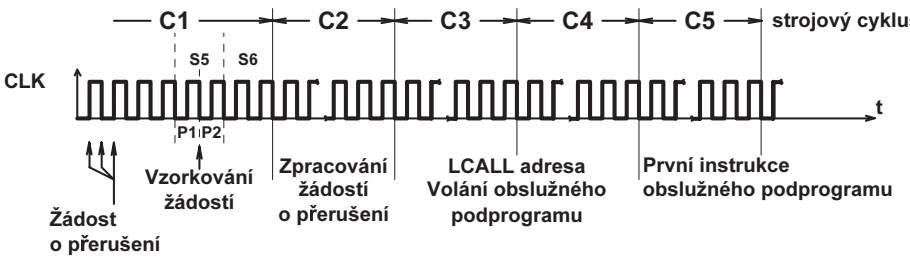
Obr. 13 Přiřazení bitů úrovně priority přerušení v IP

K využití přerušovacího systému musíme ještě znát, jakým způsobem je realizováno volání příslušného obslužného podprogramu přerušení. U procesorů řady 8051 má každý zdroj přerušení definovanou pevnou adresu, na kterou po přijetí žádosti o přerušení, dokončení rozpracované instrukce a uložení návratové adresy (adresy následující instrukce za právě dokončenou instrukcí) do zásobníku předá procesor řízení. Adresy přerušení jsou od sebe vzdáleny pouze o 8 adresových míst viz *tabulka 2*, a proto na nich obvykle bývá uložena pouze instrukce nepodmíněného skoku na vlastní obslužný podprogram zakončený obvykle instrukcí RETI (Return from interrupt).

Tabulka 2

	Zdroj přerušení	Adresa
IE0	Vnější přerušení 0	0003H
TF0	Čítač/časovač 0	000BH
IE1	Vnější přerušení 1	0013H
TF1	Čítač/časovač 1	001BH
RI +TI	Sériový kanál	0023H

Žádosti o přerušení (příznaky IE0,IE1,TF0,TF1,RI+TI) se vzorkují v době S5P2 každého strojového cyklu procesoru a vyhodnocují se v následujícím cyklu. Je-li některý z příznaků nastaven a není splněna žádná z podmínek zabraňujících vyvolání obslužného podprogramu, přerušovací systém provede instrukci LCALL (dlouhé volání podprogramu) na příslušnou adresu určenou *tabulkou 2*. Kromě individuálního nebo globálního zákazu přerušení v registru IE, může jeho vyvolání oddálit právě probíhající přerušení se stejnou nebo vyšší úrovní priority, doposud nedokončená instrukce nebo právě probíhající instrukce RETI (návrat z přerušení) nebo instrukce zapisující do registrů IE a IP. Druhá podmínka zajišťuje, že před přechodem do obslužného podprogramu se rozpracované instrukce nejprve dokončí. Poslední podmínka zajišťuje, že po instrukci RETI nebo instrukci zasahující do registrů IE a IP je vykonána ještě jedna instrukce a teprve potom je provedeno směrování na příslušný vektor přerušení. Cyklus vyhodnocení se opakuje v každém strojovém cyklu a vyhodnocují se v něm platné (vzorkované) hodnoty v periodě S5P2 předchozího strojového cyklu. Není-li žádost o přerušení obsloužena, protože byla platná některá z blokovacích podmínek, pak v době, kdy už není aktivní, se neobslouží, i když podmínky blokování zanikly. Žádost neobslouženého příznaku přerušení se nikde neuchovává (vyjma příznaku) a v každém cyklu vyhodnocení se pracuje s novými hodnotami získanými v předcházejícím strojovém cyklu. Aby vzorkování vnějších přerušení $\overline{INT0}$, $\overline{INT1}$ bylo správné, musí aktivaci přerušení sestupnou hranou ($1 \rightarrow 0$) předcházet hodnota log. 1 i log. 0 na vstupech $\overline{INT0}$, $\overline{INT1}$ alespoň 12 period oscilátoru (jeden strojový cyklus). V případě aktivace nízkou úrovní signálu, musí žádost trvat tak dlouho, dokud nedojde k přechodu do obslužného podprogramu. Nebude-li žádost v průběhu obslužného podprogramu zrušena, potom dojde k jeho opětovnému vyvolání.



Obr. 14 Nejrychlejší odezva na přerušení

Na *obr. 14* je zobrazena situace při přijetí žádosti o přerušení v případě, kdy procesor zpracovává jednocyklové instrukce nepracující s registry IE a IP. V cyklu C1 před periodou S5P2 přichází žádost o přerušení, která se vyhodnocuje v cyklu C2. Jsou-li splněny všechny podmínky pro vyvolání přerušení, potom v cyklech C3 a C4 je generována instrukce LCALL na příslušnou adresu obslužného

podprogramu. V cyklu C5 potom může být zpracována první instrukce obslužného podprogramu. Vznikne-li žádost o přerušení s vyšší prioritou před periodou S5P2 strojového cyklu C3, potom bude přerušení obslouženo podle výše uvedených pravidel během strojových cyklů C5 a C6, aniž se vykoná jediná instrukce obslužného podprogramu s nižší prioritou. Mezi vznikem požadavku na přerušení a první instrukcí obslužného programu proběhnou nejméně tři strojové cykly. Odezva na přerušení se prodlouží při platné podmínce blokující přechod do obslužného podprogramu. V případě probíhajícího přerušení se stejnou nebo vyšší úrovní priority, je doba čekání závislá na délce a vlastnostech obslužného podprogramu. Není-li vyhodnocení žádostí prováděno v posledním cyklu instrukce, potom prodloužení odezvy nebude větší než 3 strojové cykly (instrukce MUL a DIV trvají 4 cykly). Probíhá-li právě instrukce RETI nebo instrukce operující s IE nebo IP, prodlouží se doba odezvy nejvíce o 5 strojových cyklů (1 cyklus pro dobíhající instrukci a 4 cykly za dokončení nejdelší následující instrukce MUL nebo DIV). Využíváme-li v systému pouze jedno přerušení, potom časová odezva bude vždy delší než 3 a kratší než 9 strojových cyklů.

Krokování programu

Struktura přerušení dovoluje provádět činnost zkoumaného programu po jednotlivých krocích s velmi malými softwarovými a hardwarovými náklady za cenu ztráty jednoho vstupu externího přerušení např. INT0 nebo INT1. Z předcházejícího textu je zřejmé, že požadavek na přerušení se nebude přijat, bude-li se již zpracovávat přerušení se stejnou úrovní priority. Odmítání požadavku na přerušení bude trvat do té doby, dokud se za instrukcí RETI neprovede další jedna instrukce. Využijeme-li vnější přerušení INT0 v režimu, kdy reaguje na úroveň, můžeme zakončit obslužný program přerušení následovně:

```
INT0:       .............
.............
ČEKEJ1:JNB P3.2, ČEKEJ1;čekej až na vývodu INT0 bude log. 1
ČEKEJ0:   JB P3.2, ČEKEJ0;čekej až na vývodu INT0 bude log. 0
RETI
```

Přechod signálu na vývodu INT0 do log. 0 způsobí přechod do výše uvedeného obslužného programu přerušení. Přechod signálu z 0→1 a z 1→0 způsobí opuštění obslužného programu přerušení. Protože je zároveň vytvořena další žádost o přerušení, je vykonána právě jedna instrukce zkoumaného (krokovaného) programu.

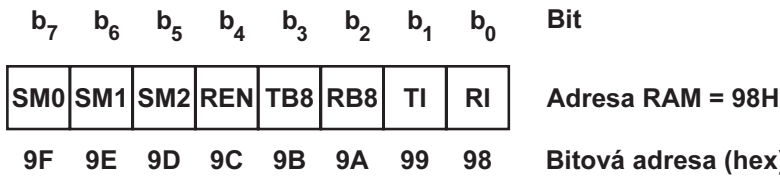
1.1.5. Sériový kanál

Jednou z velkých výhod procesoru 8051, zvláště v době jeho vzniku, byl a je plně duplexní sériový kanál integrovaný na čipu procesoru, který umožňuje komuni-

kaci ve standardním 8 a 9bitovém asynchronním režimu nebo 8bitovém synchronním režimu s pevnou přenosovou rychlostí. Tím byla výrazně usnadněna komunikace s nadřízeným počítačem např. typu PC, k jejíž realizaci je v současné době zapotřebí jeden integrovaný obvod (MAX232, MAX233) zajišťující převod úrovní TTL sériového kanálu na úrovně V24 (PC). Plně duplexní sériový kanál umožňuje současně vysílat i přijímat hodnoty po tomto kanálu, který tvoří minimálně 3 vodiče (RxD,TxD a zem). Přijímací kanál je vybaven vyrovnávacím registrem, do kterého je uložena právě přijatá hodnota, čímž je umožněn okamžitý příjem další hodnoty. Přijatá hodnota však musí být převzata dříve, než je dokončen příjem následující hodnoty, který by způsobil přepsání původní hodnoty. Procesor není vybaven příznaky, které indikují ztrátu přijaté hodnoty (chybu přeplnění), chybu rámce a parity nebo indikaci přerušení, které jsou obvyklé u specializovaných obvodů.

Přijímací i vysílací registr je dostupný na stejné adrese se symbolickým označením **SBUF** - **registr sériového kanálu** v prostoru speciálních funkcí. Zápisem se naplňuje vysílací registr, čtením SBUF je přečtena hodnota z vyrovnávacího registru, do kterého byla přepsána z přijímacího registru po přijetí celého bytu. Sériový kanál může pracovat ve čtyřech módech v závislosti na naprogramováním registru SCON a nejvyššího bitu v registru PCON.

SCON - Registr módu a řízení sériového kanálu se skládá z 8 bitů, jejichž umístění je zobrazeno na *obr. 15*. Význam jednotlivých bitů je následující:

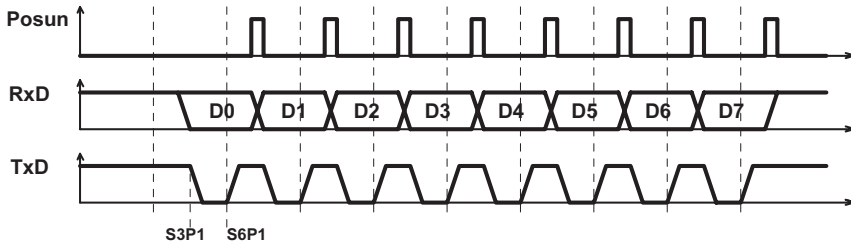


Obr. 15 Rozložení bitů v registru SCON

♣ **SM0, SM1** - Konfigurační bity určují jeden ze čtyř módů sériového kanálu, které jsou popsány *tabulkou 3*.

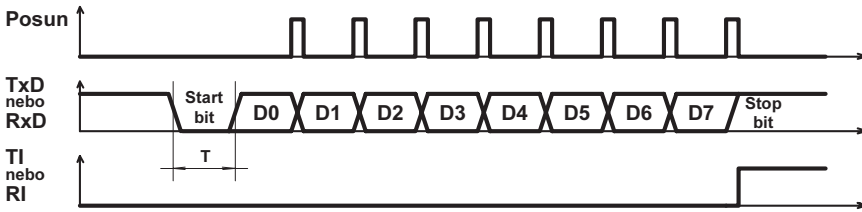
SM0, SM1	Mód	Typ přenosu	Bitová rychlost
0 0	0	Synchronní 8bit. bez rám. synchr.	$f_{osc}/12$
0 1	1	8 bitový UART	časovač 1
1 0	2	9 bitový UART	$f_{osc}/32$, $f_{osc}/64$
1 1	3	9 bitový UART	časovač 1

Tabulka 3



Obr. 16 Časování sériového kanálu v módu 0

• **Mód 0** - Sériová data se vysílají nebo přijímají vstupem P3.0 označeném RxD, synchronně s hodinovým (posouvacím) signálem, vysílaným na výstupu TxD=P3.1. Přenášená 8bitová informace se vysílá, počínaje bitem s nejnižší váhou *obr. 16*. Přenosová rychlost je pevná a rovna 1/12 kmitočtu oscilátoru.

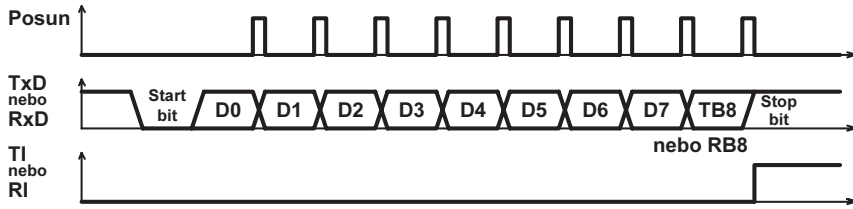


Obr. 17 Časování sériového kanálu v módu 1

• **Mód 1** - 8bitový UART *obr. 17*. Hodnoty se vysílají výstupem TxD a přijímají vstupem RxD a skládají se z deseti intervalů, určených převrácenou hodnotou přenosové rychlosti v baudech pro přenos jednotlivých bitů. První bit je vždy nulový (log. 0) a představuje tzv. **start bit**, po němž následuje 8 přenášených bitů počínaje bitem s nejnižší váhou a posledním, který je vždy v log. 1 a představuje tzv. **stop bit** (přesně jeden stop bit). Při příjmu se stop bit ukládá do bitu RB8 v registru SCON. Přenosová rychlost je volitelná a je určena periodou přetečení čítače/časovače 1 a hodnotou nejvyššího bitu (SMOD) v registru PCON. Pro přenosovou rychlost můžeme za předpokladu, že čítač 1 pracuje v módu 2, snadno odvodit tento vztah

$$\text{přenosová rychlost} = \frac{1}{7} = \frac{2^{SMOD}}{32} \cdot \frac{f_{osc}}{12 * [256 - (TH1)]}$$

kde (TH1) je obsah registru TH1 a f_{osc} je kmitočet oscilátoru. Pro dosažení nízkých přenosových rychlostí se využívá čítač v módu 1, kde po vzniku přerušení provedeme programové přednastavení potřebnou 16bitovou hodnotou.



Obr. 18 Časování sériového kanálu v módu 2 a 3

• **Mód 2** - 9bitový UART. Při vysílání je na vývodu TxD generováno 11 bitů v odpovídajících intervalech, daných přenosovou rychlostí, uvedených start bitem (log. 0) a ukončených jedním stop bitem (log. 1) jako v předcházejícím módu. Přijímaný znak přichází na vstup RxD. Devátým vysílaným bitem je hodnota bitu TB8 z registru SCON. Přijatý devátý bit se ukládá do bitu RB8 v registru SCON a stop bit se ignoruje. Devátý bit může být využit k přenosu hodnoty (9 bitů) nebo k přenosu zabezpečovacího bitu, například parity P (ochrany proti chybě v jednom bitu při přenosu). Hodnotu parity získáme z registru příznaků PSW, pokud do střadače uložíme přenášenou hodnotu. Přenosová rychlost je dána 1/32 nebo 1/64 kmitočtu oscilátoru v závislosti na hodnotě bitu SMOD v registru PCON.

• **Mód 3** - 9bitový UART s programovatelnou přenosovou rychlostí. Příjem i vysílání hodnot probíhá stejně jako v módu 2 s tím, že přenosová rychlost je určena periodou přetečení čítače/časovače 1 a nastavením bitu SMOD.

Ve všech čtyřech režimech se vysílání spouští instrukcí, která využívá SBUF jako cílový registr (zápisem do SBUF). Příjem v módu 0 se spouští podmínkami RI=0 a REN=1. V ostatních režimech se příjem spouští příchodem start bitu při REN=1.

♣ **SM2** - Bit povolující vytvoření víceprocesorové sériové sběrnice v módu 2 a 3. Je-li v módu 2 a 3 nastaven příznak SM2=1, pak RI se nenastaví (nemůže být vyvoláno přerušení), jestliže přijatý devátý bit (RB8) má hodnotu log. 0. V módu 1 může být SM2 využit ke kontrole platnosti stop bitu a příjmu dat jenom s platným stop bitem. V módu 0 se bit SM2 nevyužívá.

♣ **REN** - Povolení příjmu (REN=1). Bit se nastavuje i nuluje programově.

♣ **TB8** - Devátý datový bit při vysílání. Vysílá se v módech 2 nebo 3. Nastavuje se a nuluje se programově.

♣ **RB8** - Devátý datový bit při příjmu. Přijímá se v módech 2 nebo 3. V módu 1 při SM2=0, obsahuje RB8 přijatý stop bit. V módu 0 se RB8 nevyužívá.

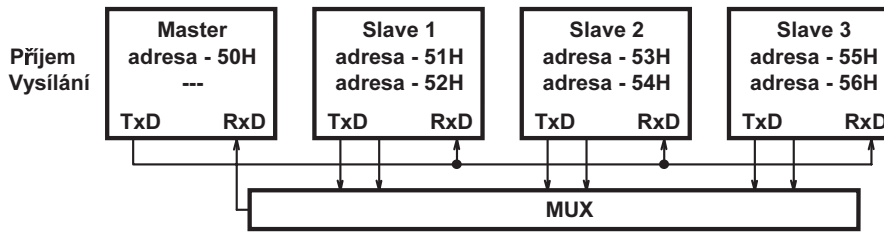
♣ **TI** - Příznak prázdného vysílacího posuvného registru se nastavuje obvodově v okamžiku vysílání osmého přenášeného bitu v módu 0 nebo na začátku

vysílání stop bitu v ostatních módech. Příznak TI je společně s příznakem RI zdrojem žádosti o přerušení sériového kanálu, a proto nemůže být z důvodu identifikace zdroje (od TI nebo RI) přerušení nulován obvodově. Proto uživatel sám musí po přijetí žádosti rozlišit, zda se jedná o žádost od příjmu (RI) nebo vysílání (TI), a teprve potom příslušný příznak programově vynulovat.

♣ **RI** - Příznak přijatých platných dat se nastaví na konci příjmu osmého bitu v módu 0 nebo uprostřed přijímaného stop bitu v ostatních módech. Příznak se stejně jako TI nuluje programově, aby bylo možné rozlišit příčinu přerušení.

1.1.6. Multiprocesorová komunikace

Režim 2 a režim 3 je vybaven prostředky, které umožňují realizovat multiprocesorovou komunikaci. V těchto režimech se přijímá devět datových bitů, z nichž devátý bit se ukládá do bitu RB8 v řídicím registru sériového kanálu SCON. Sériový kanál lze naprogramovat pomocí bitu SM2 z registru SCON tak, aby se při příjmu stop bitu aktivovalo přerušení od sériového kanálu jedině tehdy, je-li logická hodnota RB8=1. Této možnosti lze využít v klasické multiprocesorové



Obr. 19 Jedna z možných multiprocesorových konfigurací s procesory 8051

konfiguraci (Master-Slave) následujícím způsobem *obr. 19*. Chce-li nadřízený procesor (master) přenášet do jednoho z podřízených procesorů (slave) blok dat, vyšle nejdříve adresový byte, který identifikuje cílový procesor. Adresový byte se liší od datového bytu logickou hodnotou právě devátého bitu. U adresového bytu je v devátém bitu log. 1, zatímco datový byte obsahuje v devátém bitu log. 0. Je-li logická hodnota SM2=1, pak nebude žádný procesor přerušován datovým bytem, avšak adresový byte přeruší všechny podřízené procesory. Testováním přijatého bytu může každý procesor zjistit, zda je adresován a následující datové údaje jsou určeny pro jeho činnost. Adresovaný podřízený procesor vynuluje bit SM2 a připraví se k příjmu datových bytů. Ten podřízený procesor, který nebyl adresován, ponechá svůj bit SM2 nastaven a pokračuje v původní činnosti před přerušením a ignoruje předcházející datové byty. Formát takové zprávy, která se

přenáší mezi procesory, je tvořen adresou následovanou vlastní informací s délkou n bytů takto

⟨adresa⟩, ⟨1 byte dat⟩, ⟨2 byte dat⟩,…, ⟨n byte dat⟩

Například přenos hodnot 55H a AAH z procesoru master do slave 1 bude vypadat následovně (devátý bit je zvýrazněn)

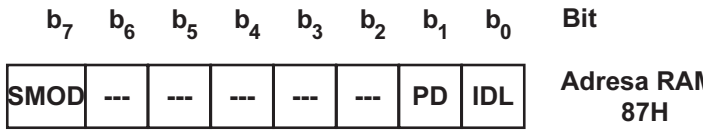
10001010 **1**	10101010 **0**	01010101 **0**
DGUHVD	55+	AA+

Komunikace z podřízeného do řídicího mikroprocesoru bude probíhat obdobným způsobem s tím, že například jinou adresou vyzve nadřízený procesor jeden z podřízených procesorů k přenosu dat. Ten zajistí připojení výstupu TxD na vstup RxD nadřízeného procesoru a realizuje přenos. V konfiguraci z *obr. 19* by takový přenos nemusel být uveden adresou. Vzhledem k tomu, že výstupy brány 3 (kde je vývod TxD) jsou realizovány s otevřeným kolektorem, mohou být jednotlivé výstupy TxD paralelně spojeny a připojeny na vstup RxD nadřízeného procesoru. Podmínkou takového propojení je, že vysílání bude realizováno pouze jedním podřízeným procesorem.

1.1.7. Režimy se sníženou spotřebou

U aplikací, kde není vyžadována soustavná činnost a jsou kladeny přísné požadavky na spotřebu energie, je obvykle nutné u procesorů využívat režimů se sníženou spotřebou. Procesory řady 8051 umožňují v závislosti na tom, v jaké technologii jsou vyrobeny, režimy se sníženým příkonem a se sníženým napájením. U starých procesorů vyrobených technologií HMOS je snížení příkonu možné pouze v režimu s vypnutým napájením, při kterém chceme uchovat obsah paměti RAM na čipu. Pro tento účel byl procesor vybaven vstupem pro pomocný (záložní) zdroj napětí U_{PD} sdruženým se vstupem nulování procesoru RST. Zjistí-li uživatelský systém, že dochází k poklesu napájecího napětí, musí přerušit činnost procesoru a přesunout důležité údaje do paměti RAM na čipu procesoru. Potom připojí zdroj záložního napětí na vstup RST a vygeneruje tak nulování procesoru (RESET). Tato činnost musí být ukončena dříve, než napětí U_{CC} poklesne pod spodní úroveň provozního napětí určeného výrobcem. Po opětovném naběhnutí napájení musí zůstat záložní napětí připojeno tak dlouho, dokud nedojde k spolehlivé činnosti oscilátoru procesoru (10 ms) a U_{CC} nedosáhne alespoň spodní úroveň provozního napětí. Po splnění všech podmínek může být záložní napětí odpojeno a procesor může zahájit normální činnost. Moderní procesory vyrobené CMOS technologií jsou vybaveny dvěma režimy se sníženým příkonem, do kterých lze procesor uvést nastavením příslušných bitů v registru PCON. Vstupem, na který se přivádí záložní napětí, je vývod pro vlastní napájení U_{CC}.

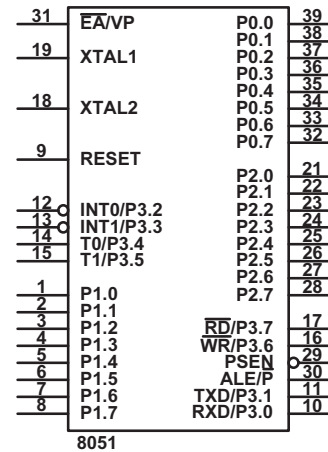
PCON - Registr řízení napájení (Power Control) obsahuje u jádra procesoru 80(C)51 jeden nebo tři bity *obr. 2*0. První odlišností tohoto registru od všech předcházejících je to, že jeho bity nejsou bitově adresovatelné a jeho obsah musí být měněn zápisem celého nového bytu. Nejvyšší bit označený SMOD nemá s řízením napájení nic společného a je pouze doplňujícím bitem pro řízení sériového kanálu SMOD sloužícím k zdvojnásobení jeho přenosové rychlosti. Jedná-li se



Obr. 20 Rozložení bitů v registru PCON

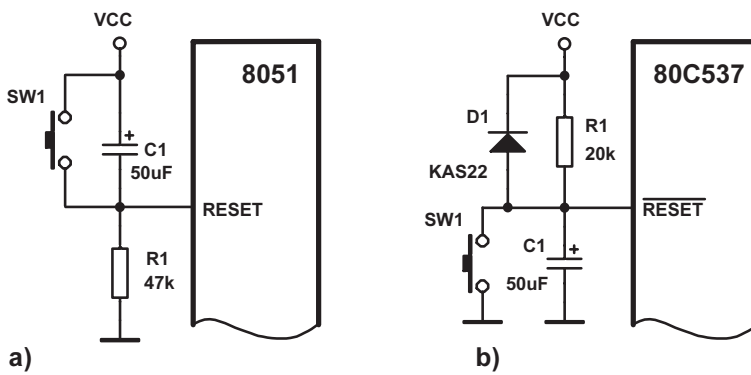
o procesor, který je vyroben HMOS technologií, potom je tento bit jediným bitem využívaným z registru PCON. U procesorů vyrobených technologií CMOS jsou využívány další dva bity označené PD a IDL, které slouží k uvedení procesoru do módů se sníženou spotřebou. Instrukce, která nastaví bit **IDL** do log. 1 je poslední instrukcí provedenou před přechodem do tzv. **Idle** módu, u kterého jsou všechny aktivity jádra procesoru zastaveny. Oscilátor i periferní obvody jako je sériový kanál, časovače, atd. pokračují v práci dál. Zůstává zachován stav procesoru jako je ukazatel zásobníku, programový čítač, stavové slovo, střadač i zbývající speciální registry a vnitřní stav datové paměti RAM. Stavy vstupně/výstupních bran drží svůj logický stav, který měly v době přechodu do Idle módu. Signály ALE a $\overline{\text{PSEN}}$ se stávají neaktivní. Uvedený mód, při kterém klesne spotřeba obvodu na cca 15 mA, lze opustit přijetím libovolného nezamaskovaného přerušení některé z periferií nebo vynulováním procesoru. Přijetí přerušení obvodově vynuluje bit IDL a procesor obslouží příslušné přerušení a přejde do normální činnosti. Po provedení instrukce RETI a ještě jediné instrukce může být znovu uveden do módu Idle. Nastavením bitu **PD** do log. 1 uvedeme procesor do tzv. **Power down** módu, u kterého je zastavena činnost celého procesoru včetně periferií i oscilátoru a díky tomu se sníží odběr ze zdroje až na desítky mA. Stejně jako v předcházejícím módu je instrukce, která nastaví bit PD do log. 1 poslední instrukcí provedenou před přechodem do power down módu. Zůstává zachován stav registrů speciálních funkcí a vnitřní paměti RAM. Oproti předcházejícímu módu nemohou být generována přerušení od vnitřních periferií a procesor může uvedený mód opustit pouze vynulováním. Tato funkce však znovu definuje obsah všech registrů speciálních funkcí, obsah paměti RAM se nezmění. V tomto režimu může být dále snížen odběr snížením záložního napájení až na hodnotu 3,3 V. S výhodou tak lze využít obvodů MAX690 až MAX696 sdružujících watchdog, komparátor napětí a přepínač záložního napájení.

1.2. Zapojení vývodů mikroprocesoru 8051



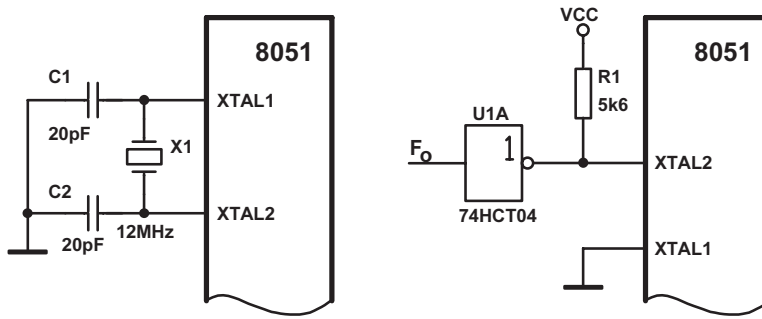
Obr. 21 Procesor 8051

Klasický mikroprocesor 8051 *obr. 21* se vyráběl v plastikovém nebo keramickém pouzdře DIL se 40 vývody, nyní se většinou využívá plastikového pouzdra PLCC se 44 vývody. Ke své činnosti vyžaduje jedno napájecí napětí U_{CC} **= 5 V** a U_{SS} **= 0 V** a připojení piezokeramického rezonátoru („krystalu“) k vývodům XTAL1 a XTAL2 obvodu vnitřního oscilátoru. Procesor je vybaven 4 vstupně/výstupními branami P0 až P3, z nichž P0, P2 a P3 zajišťují další funkce, které jsou závislé na tom, zda využíváme vnější paměť programu nebo dat, vnější přerušení, vnější vstupy časovačů nebo sériový kanál. Pro řízení vnější paměti je procesor vybaven řídicími signály ALE (pro zápis spodní poloviny platné adresy A0 až A7), $\overline{\text{PSEN}}$ (pro čtení z vnější paměti programu), $\overline{\text{RD}}$ a $\overline{\text{WR}}$ (pro čtení nebo zápis do vnější datové paměti). Kromě uvedených signálů je procesor vybaven již zmíněným vstupem EA, který určuje přístup k vnější programové paměti a nulovacím vstupem RESET s aktivní úrovní v log. 1. Na tomto místě je třeba upozornit na to, že řada nástupců má aktivní úroveň nulovacího impulzu opačnou (log. 0), jak bývá u většiny procesorů obvyklé. Na *obr. 22a* je zobrazen nejjednodušší nulovací obvod pro aktivní signál RESET=1, na *obr. 22b* je obdobný obvod pro opačnou polaritu nulovacího signálu (RESET=0). Nulo-



Obr. 22 Jednoduché nulovací obvody

vací signál musí být aktivní nejméně po dva strojové cykly (tj. 24 period oscilátoru) v době plné činnosti oscilátoru. Ačkoliv se úloha zdá být jednoduchá, je třeba pamatovat na to, že napájecí zdroj může mít relativně pomalý náběh napětí a vnitřní oscilátor procesoru je obvykle nastartován po 5 ms až 10 ms od dosažení plného napájecího napětí. Uvedené jednoduché nulovací obvody nejsou vhodné pro systémy se zálohovanou pamětí dat RAM nebo pamětí EEPROM, protože v okamžiku odpojení nebo krátkodobého poklesu napájecího napětí nevytvářejí nulovací impulz, který by vyřadil procesor z činnosti po dobu napájecího napětí mimo tolerance stanovené výrobcem (tj. 4,75 až 5,25 V nebo 4,5 až 5,5 V). Necháme-li procesor mimo tento rozsah napájecího napětí pracovat, potom má „dostatek času“ k přepsání zálohovaných dat. Proto v systémech, kde je třeba zálohovat data, je nutné použít nulovací obvod (TL7705) nebo nulovací generátor s obvodem watchdog (MAX690 až MAX699). Nulováním procesoru se obsah vnitřní paměti RAM nemění. Po nulovacím impulzu se samozřejmě vynuluje čítač instrukcí PC, zakáže se přijetí jakéhokoliv přerušení IE=0X000000B a V/V brány se nastaví do vstupního režimu P0 až P3 (FFH). Kromě nastavení registrů PC, IE a P0 až P3, které je pro správnou činnost procesoru nezbytné, je vynulována většina ostatních speciálních registrů jako jsou ACC-střadač, B, PSW, DPTR, IP, TMOD, TCON, PCON (0XXX0000B), SCON, TH0, TL0, TH1 a TL1. Pouze ukazatel zásobníku SP je nastaven na hodnotu 07H.

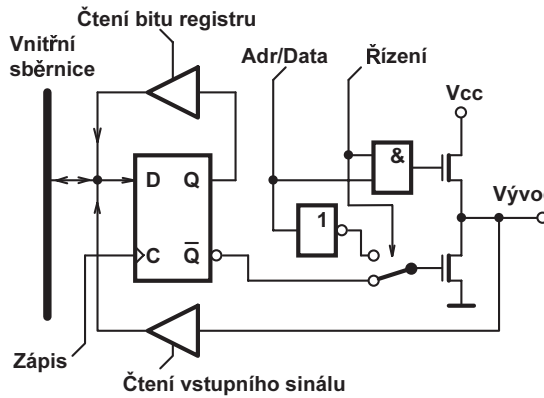


Obr. 23 Způsoby synchronizace mikroprocesoru 8051

Synchronizační signál je obvykle tvořen vnitřním oscilátorem, ke kterému připojujeme krystal dle *obr. 23a*. Kmitočet oscilátoru u standardního mikroprocesoru 8051 se může pohybovat v rozsahu 1,2 MHz až 12 MHz. U některých nástupců lze spodní kmitočet snížit až k nule, nejvyšší kmitočet může dosáhnout až 33 MHz. Budeme-li využívat synchronizační kmitočty nad 20 MHz, pak většinou využijeme integrované oscilátory, které připojíme k mikroprocesoru dle *obr. 23b*.

1.2.1. Struktura a činnost vstupně/výstupních bran

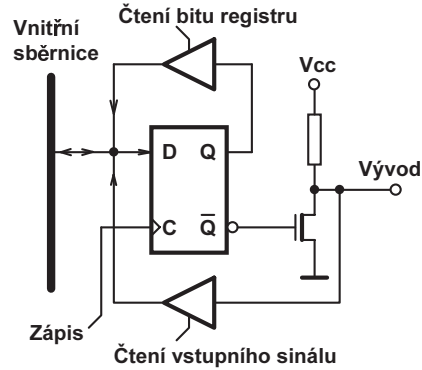
Mikroprocesor 8051 je vybaven čtyřmi obousměrnými branami *obr. 2*, kde každou bránu tvoří záchytné klopné obvody, výstupní budiče a vstupní vyrovnávací paměť. Výstupní budiče brány 0 a 2 a vstupní vyrovnávací paměť brány 0 se využívají pro styk s vnější pamětí. V takovém případě je na bráně 0 časově přepínán výstup nižší části adresy pro vnější paměť s hodnotou dat, která jsou zapisována nebo čtena z paměti. Z brány 2 pak vystupuje vyšší byte 16bitové adresy pro vnější paměť. V ostatních případech brána 2 přenáší obsah speciálního vstupně/výstupního registru P2. Brány P1 a P3 jsou uživateli volně k dispozici s tím, že na bráně 3 a u nástupců i na dalších branách jsou zavedeny další alternativní funkce. I když zapojení vstupně/výstupního obvodu každé brány je poněkud odlišné, je pouze brána P0 typickou obousměrnou branou *obr. 24*. Brány 1, 2 a 3 jsou tvořeny tzv. pseudoobousměrným obvodem *obr. 25*, jehož výstup je tvořen obvodem s otevřeným kolektorem s integrovaným odporem cca 50 kΩ. Pro zajištění rychlého náběhu čela impulzu (nabití parazitních kapacit) je hodnota tohoto odporu po zápisu do V/V registru na dva hodinové cykly výrazně snížena (cca 500 Ω). Výstupní tranzistor může být typicky zatížen proudem 1,6 mA, což odpovídá počtu vstupů TTL obvodů LS=6, ALS=12 a F=2. Počet obvodů HCT bude omezen jejich vstupní kapacitou.



Obr. 24 Konfigurace bitu brány 0

Chceme-li používat brány P1, P2 a P3 nebo jejich bity jako vstupní, potom v jejich registrech nebo příslušných bitech, které ovládají výstupní tranzistory, musí být zapsány logické jedničky. Teprve potom budeme moci z vodiče číst správnou logickou hodnotu. Hlavní výhodou V/V bran je to, že jsou bitově adresovatelné, a není proto při testování stavu jednoho bitu nutné provádět čtení celé brány a její maskování. Oproti procesoru 8048 jsou ve V/V bráně 8051 umístěny oddělovače, které umožňují přenést na vnitřní sběrnici procesoru buď obsah klopného obvodu (registru) nebo logickou hodnotu z vývodů brány. Tím je umožněno zjistit

hodnotu zapsanou do výstupní brány nezávisle na obvodech připojených k vývodům této brány (např. přechodu tranzistoru báze-emitor). Z popisu instrukcí zjistíme, že instrukce modifikující hodnotu zapsanou do V/V brány, čtou obsah výstupního registru. Všechny instrukce, jejichž cílovým operandem je V/V brána



Obr. 25 Konfigurace bitu brány 1

nebo jeden její bit, patří k instrukcím pracujícím s registrem brány a nikoliv s vývody brány. Instrukce typu „čtení-modifikace-zápis“ jsou zobrazeny v *tabulce 4*. Protože při zapnutí mikroprocesoru není známo, zda pseudoobousměrný vývod bude využíván jako vstupní nebo výstupní (určeno až programem), musí být všechny vstupně//výstupní brány nastaveny do vstupního režimu (nedojde ke spojení výstup-výstup) tím, že do registrů bran P0 až P3 jsou zapsány jedničky (FFH).

ANL - logický součin, např. ANL P3,A	ORL - logický součet, např. ORL P1,A
XRL - operace EX-OR, např. XRL P2,A	CPL - inverze bitu, např. CPL P1.2
INC - inkrementace (+1), např. INC P1	DEC - dekrementace (-1), např. DEC P1
CLR Pn.m - vynulování bitu m brány n	SET Pn.m - nastavení bitu m brány n
MOV Pn.m,C přesun přenosu do bitu m brány n	
JBC - skok, je-li bit=1 (bit je poté vynulován), např. JBC P1.1, návěští	
DJNZ - dekrementace a skok je-li obsah nenulový, např. DJNZ P2, návěští	

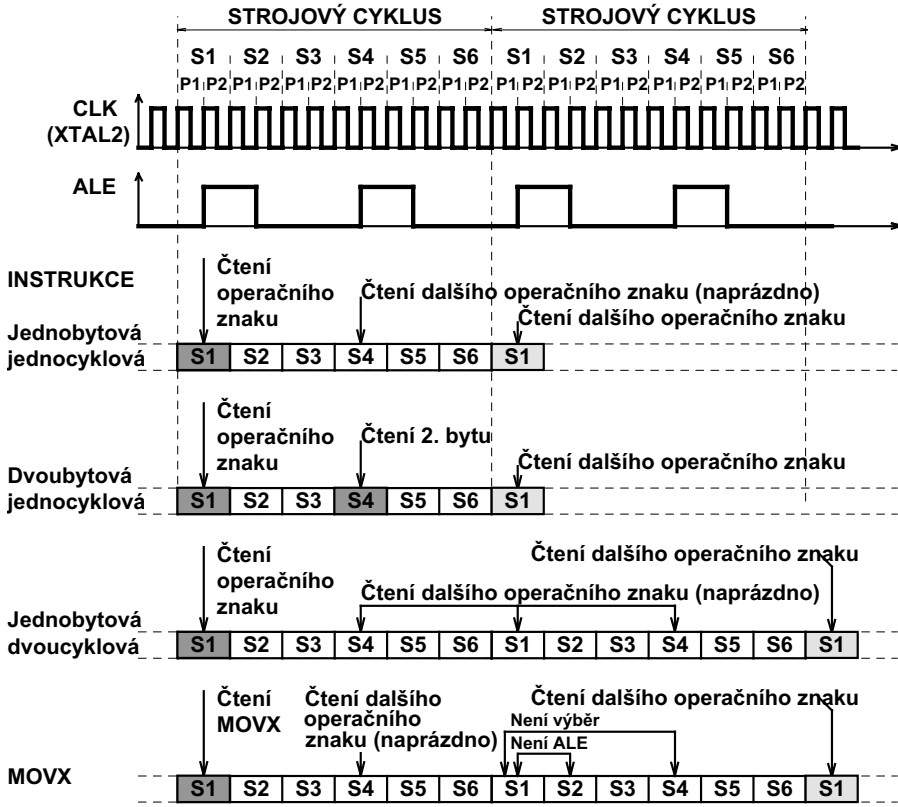
Tabulka 4 Instrukce „čtení-modifikace-zápis“

1.2.2. Časování centrální procesorové jednotky

Strojový cyklus procesoru se skládá ze šesti stavů označených S1,S2, ... ,S6, z nichž každý je dále rozdělen na dvě fáze P1 a P2. Každý strojový cyklus je tak tvořen 12 fázemi označovanými S1P1, S1P2, S2P1, ... , S6P2 shodnými s periodami synchronizačního oscilátoru. Na *obr. 26* je zobrazeno časování čtení a vy-

konání čtyř možných typů instrukcí procesoru 8051 vzhledem k jeho vnitřnímu časování, které je uživateli nedostupné. Pro lepší orientaci jsou na *obr. 26* zobrazeny průběhy signálů ALE a XTAL2. Signál ALE se normálně aktivuje dvakrát během jednoho strojového cyklu. Poprvé ve fázi S1P2 až S2P1 a podruhé během S4P2 až S5P1 s výjimkou přístupu do vnější paměti dat (MOVX), kdy v druhém cyklu je vynechán první cyklus ALE. Z tohoto důvodu není vhodné využívat signál ALE k časování v navrhovaném systému.

Realizace jednocyklové instrukce začíná ve fázi S1P2 uložením přečteného operačního znaku do registru instrukcí. Ve stavu S4 ještě téhož strojového cyklu se provádí čtení ještě jednoho bytu z následujícího paměťového místa. Je-li přečtený byte využit v instrukci (dvoubytová a jednocyklová instrukce nebo dvoubytová a dvoucyklová instrukce), potom čítač instrukcí je inkrementován. Je-li zpracovávaná instrukce jednobytová a jednocyklová, potom přečtený byte ve sta-

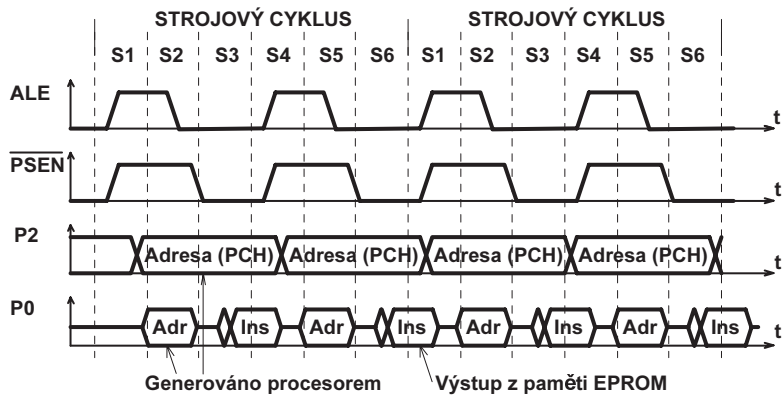


Obr. 26 Výběrové sekvence na procesoru 8051

vu S4 je ignorován a čítač instrukcí se nemění. Instrukce je vykonána ve fázi S6P2. V případě jednobytové dvoucyklové instrukce je situace stejná jako u předcházejícího případu s tím, že čtení dalšího bytu je ignorováno třikrát. Jedinou výjimkou je instrukce přístupu do vnější paměti MOVX, která jako jednobytová dvoucyklová instrukce negeneruje ve fázi S1P2 až S2P1 druhého cyklu signál ALE a nerealizuje oba výběry dalšího bytu *obr. 2*6. Většina instrukcí se realizuje v jednom nebo dvou strojových cyklech. Jedinými instrukcemi, které potřebují více než dva strojové cykly, jsou instrukce MUL (násobení) a DIV (dělení). Obě vyžadují čtyři strojové cykly.

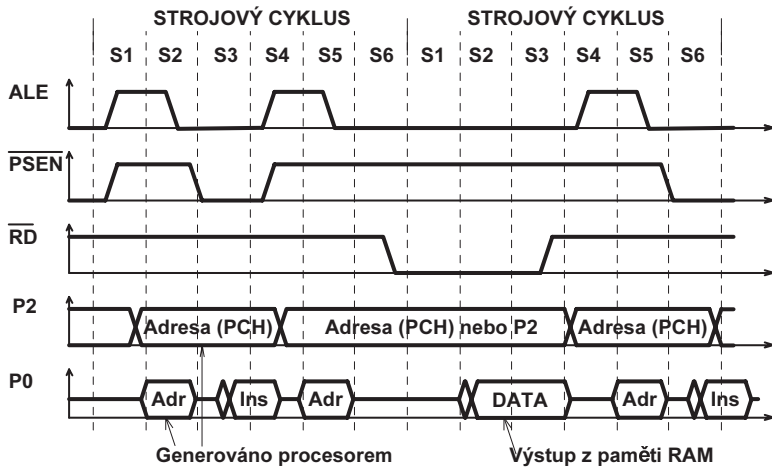
1.2.3. Přístup do vnější paměti

Vzhledem k harwardské struktuře, u níž je mimo jiné oddělena programová a datová paměť, existuje u procesoru 8051 dvojí přístup do vnějšího paměťového prostoru. Při spolupráci s vnější pamětí programu se využívá aktivační (řídicí) signál PSEN, pro spolupráci s vnější pamětí dat se používají řídicí signály RD (čtení) a WR (zápis). Při čtení operačního znaku instrukce z vnější paměti programu se využívá vždy 16bitová adresa. Při styku s vnější pamětí dat se může využívat buď 16bitová adresa (MOVX @DPTR,A) nebo 8bitová adresa (MOVX A,@Ri). Při využití celé adresové sběrnice se vyšší byte adresy automaticky vysílá na bránu P2, kde setrvá po dobu operace. Nižší byte adresy je vysílán na bránu P0 současně se signálem ALE, který slouží k jeho zápisu do vyrovnávacího registru. Adresový byte je platný při sestupné hraně signálu ALE *obr. 27* a proto je vhodné zapisovat nižší byte adresy do registru řízeného logickou úrovní (např. 74HCT373, 74HCT573, apod.). Je-li k přístupu využívána instrukce pro 8bitové adresování, potom na bráně P2 i v době cyklu s vnější pamětí zůstává zachována posledně zapsaná informace. Brána P2 pak může být využívána jako ukazatel datové stránky nebo zcela obecně.



Obr. 27 Časování při čtení z paměti programu (vyjma instrukce MOVX)

Na *obr. 27* je zobrazeno časování při přístupu procesoru do vnější paměti programu. Cyklus čtení jednoho bytu z paměti (obvykle EPROM) je zahájen přivedením spodního bytu adresy z čítače instrukcí (PCL) na bránu P0 a horního bytu adresy (PCH) na bránu P2, při signálech ALE, $\overline{\text{PSEN}}$ v log. 1. Sestupná hrana signálu ALE zajistí zápis spodního bytu adresy do vyrovnávacího registru, jehož výstup společně s bránou P2 vytváří celou 16bitovou platnou adresu. S určitým odstupem je aktivován signál $\overline{\text{PSEN}}$=0, který určuje vlastní délku cyklu čteného bytu. Signál $\overline{\text{PSEN}}$ se proto obvykle používá jako aktivační signál pro paměť programu EPROM (tj. pro vstupy CS a OE). Před vzestupnou hranou signálu PSEN, která ukončuje cyklus čtení, je procesorem přečtena hodnota vybavená z paměti. V jednom strojovém cyklu realizuje procesor dva cykly čtení z paměti programu a díky tomu umožňuje realizovat i dvoubytovou instrukci v jednom strojovém cyklu. V případě jednobytové jednocyklové nebo dvoucyklové instrukce není po přečtení druhého bytu inkrementován čítač instrukcí PC a přečtená hodnota je ignorována. Na *obr. 28* jsou zobrazeny časové průběhy při operaci čtení z vnější datové paměti. Cyklus je stejně jako v případě čtení programové paměti zahájen generováním adresy při signálech ALE, $\overline{\text{PSEN}}$, RD a WR v log. 1. Byla-li použita instrukce s 16bitovou adresou (MOVX A,@DPTR), pak spodní část adresy je generována branou P0 a horní část branou P2. Při 8bitovém adresování je pouze na bráně P0 generována spodní část adresy a na bráně P2 zůstává původní hodnota. S odstupem alespoň jedné periody hodinového signálu je aktivován signál RD=0 (WR=0 pro zápis). Protože instrukce přístupu do vnější datové paměti je dvoucyklová, je aktivní signál RD nebo WR dvakrát delší než aktivní signál PSEN. S určitým odstupem před náběžnou hranou signálu RD je pomocí brány



Obr. 28 Časování při čtení z vnější datové paměti RAM (instrukce MOVX)

P0 procesorem přečtena vybavená hodnota z paměti RAM. S definovaným odstupem po náběžné hraně RD přichází i náběžná hrana signálu ALE, která ukončuje celý cyklus s vnější pamětí RAM. Aby během celého cyklu mohla být adresa adresovaného místa stabilní, není v první periodě druhého cyklu generován signál ALE. Z tohoto důvodu, pokud je využívána instrukce MOVX, nelze využívat signál ALE k časování vnějších obvodů.

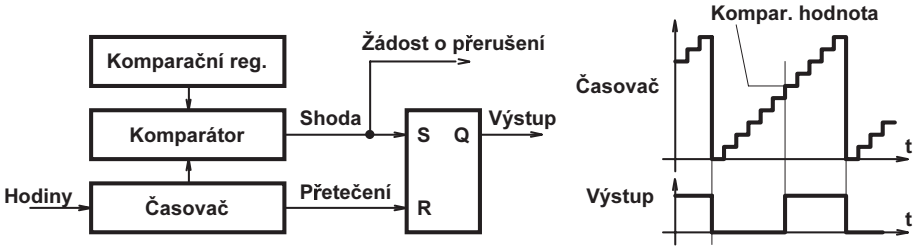
V některých vývojových systémech a aplikacích může být žádoucí, ukládat program a data do stejné paměti RAM. Tato koncepce je u procesoru 8051 možná za předpokladu, že vytvoříme signál pro čtení vnější paměti jako logický součin signálu PSEN a RD. Protože signál PSEN je kratší než signál RD, musí rychlost přístupu do vnější paměti vyhovovat časování signálu PSEN.

1.3. Procesory s jádrem 8051

Vývoj procesorů s jádrem mikroprocesoru 8051 se ubíral dvěma směry. Jeden směr vedl k vývoji malých, obvykle 20 až 28vývodových procesorů (kontrolérů), které jsou vybaveny vnitřní pamětí programu EPROM nebo EEPROM o kapacitě 1 až 4 kB, s menším počtem vstupně/výstupních bran (obvykle P1 a P3), někdy i vnitřní pamětí RAM a případně rozšířené o jednou nebo více specializovaných periferií jako je 8bitový A/D převodník, pulzně šířková modulace (PŠM, anglicky PWM) nebo sběrnice I^2C. Tyto obvody jsou určeny pro jednodušší řídicí a kontrolní aplikace. Druhý a častěji využívaný směr vedl k vývoji procesorů s jádrem 8051, které jsou rozšířeny o další vstupně/výstupní brány, 8bitový nebo 10bitový A/D převodník, další čítače, časovače pro realizaci obvodu watchdog, komparační a záchytný systém, sběrnici I^2C a případně pomocné jádro urychlující některé matematické operace. Tyto procesory se vyrábějí v závislosti na počtu periferií v pouzdrech se 40 (DIL), 44 (PLCC,QFP), 68 (PLCC), 80 (QFP) a 84 (PLCC) vývody.

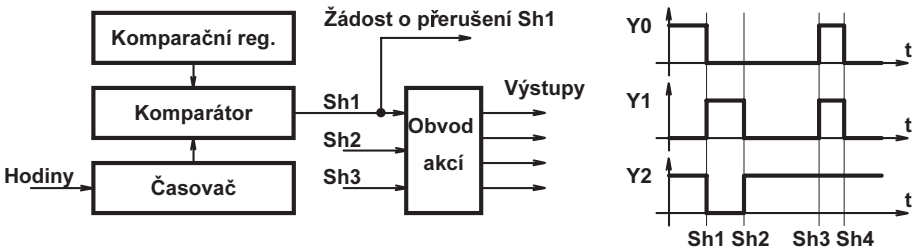
1.3.1. Periferie rozšířených procesorů

Jednou z nových periferií, integrovaných do řady univerzálních i signálových procesorů, je **komparační systém**. Jedná se o velmi výkonnou periferii určenou převážně pro průmyslové aplikace, ve kterých je třeba: generovat pulzy s různou šířkou, realizovat měření šířky impulzů, řídit krokové motory, generovat kmitočty, D/A převody a řídit procesy (obecně). Tento systém umožňuje **pulzně šířkovou modulaci** (dále **PŠM**) nebo generování řídicích signálů s minimální programovou obsluhou. Komparační systém se skládá z jednoho nebo několika komparačních registrů, časovače a komparátorů, které indikují shodnost obsahů registrů s časovačem. Systém může pracovat v konfiguraci PŠM nebo v konfiguraci generování řídicích signálů. Při konfiguraci pro PŠM *obr. 29* je systém doplněn paměťovým členem, který je nastavován při dosažení shody obsahu komparačního registru s obsahem čítače a nulován při přetečení časovače. Pro včasné uložení nové hodnoty do registru je možné od signálu shody odvodit žádost o pře-



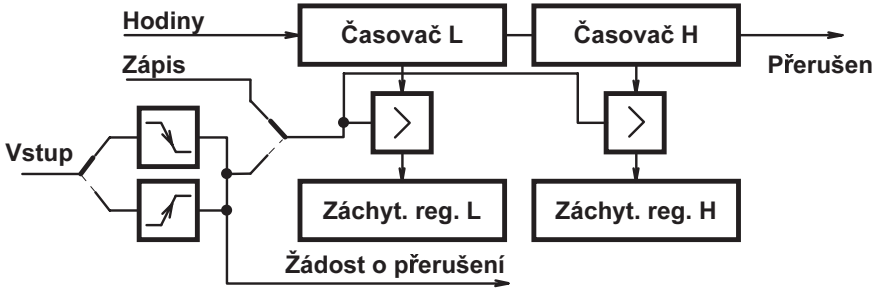
Obr. 29 Komparační systém pro PŠM

rušení, které zajistí její uložení. Kmitočet PŠM lze regulovat vstupním hodinovým kmitočtem nebo modulem časovače. V konfiguraci **generující řídicí signály** *obr. 30* je systém vybaven několika komparačními registry a komparátory. Jejich výstupní signály jsou přivedeny do výběrového obvodu, který má pro každou shodu přiřazenu odpovídající reakci na všech výstupních vodičích systému. K určení akcí, kterými mohou být přechody 0→1, 1→0 nebo impulzy na výstupech komparačního systému, existují tzv. akční registry. Z výstupu komparátorů může být stejně jako v předcházejícím případě odvozeno přerušení.



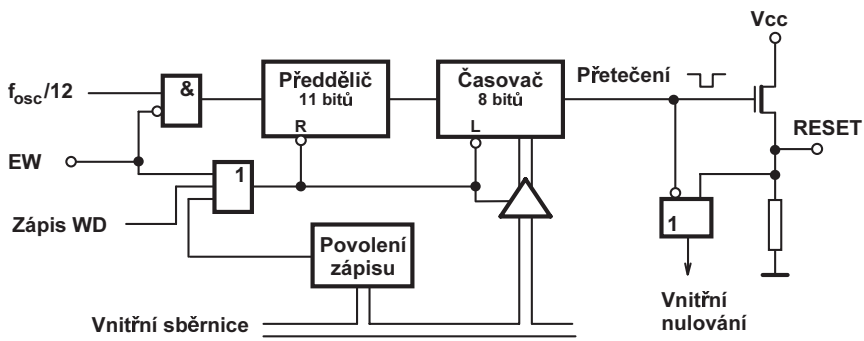
Obr. 30 Komparační systém pro generování signálů

Další výkonnou jednotkou moderních procesorů je **záchytný systém (capture)**, který slouží k pulzně šířkovému měření. Systém je tvořen obvykle 16bitovým čítačem, obvodem výběru události (vzestupná nebo sestupná hrana) a záchytnými registry. Jedná-li se o 8bitový mikroprocesor, potom záchytné registry jsou dva *obr. 31*, z nichž jeden zachycuje spodní a druhý horní část obsahu čítače. U systémů se záchytnými registry může k zachycení další události dojít až po vyvolání přerušení způsobeného událostí a po převzetí hodnot z registrů k dalšímu zpracování. Nebude-li tato podmínka splněna, dojde ke čtení nesprávného časového údaje z registrů. U rychlých procesorů bývají záchytné registry nahrazeny několikaúrovňovým zásobníkem FIFO, který umožňuje získat časovou informaci o událostech rychle po sobě jdoucích. Není-li záchytný systém využíván, lze jeho vstupy použít jako další externí vstupy žádostí o přerušení.



Obr. 31 Zjednodušené zapojení záchytného systému 8bitového procesoru

Jedním z problémů mikroprocesorových systémů bez obsluhy s nepřetržitým provozem je jejich selhání v důsledku vnějšího rušení nebo poklesu napájecího napětí. Pro takovéto systémy byl do mikroprocesorů integrován obvod tzv. hlídací pes (**watchdog**). Obvod se skládá z volně běžícího, obvykle 8bitového čítače, který může být systémem po zapsání předepsané posloupnosti vynulován. Nulovací posloupnost je volena tak, aby pravděpodobnost jejího vzniku u systému nevykonávajícího požadovaný program byla minimální. Jestliže čítač nebude včas vynulován, dojde k jeho přetečení, které způsobí vznik nulování mikroprocesoru (RESET). Nulování může být pouze interní nebo je vyvedeno na jeden vývod procesoru a umožňuje tak vynulování i vnějších obvodů mikroprocesorového systému. U některých procesorů (80C537) jsou dokonce dva obvody watchdog, z niž jeden hlídá činnost procesoru a druhý činnost jeho oscilátoru. Činnost těchto obvodů přináší problémy u procesorů, které lze uvést do stavu se sníženou spotřebou (**Idle Mode** nebo **Power Down Mode**). U módu Idle, při kterém zůstává oscilátor v činnosti, by došlo k vynulování procesoru a jeho opětného uvedení do normální činnosti. U power down módu, kdy je vyřazen z činnosti i oscilátor, je činnost obvodu watchdog vyloučena. Obvod watchdog proto bývá využíván u systémů, kde nepředpokládáme využití stavů se sníženou spotřebou.



Obr. 32 Typická konfigurace časovače ve funkci „watchdog“

1.3.2. Procesory 8052

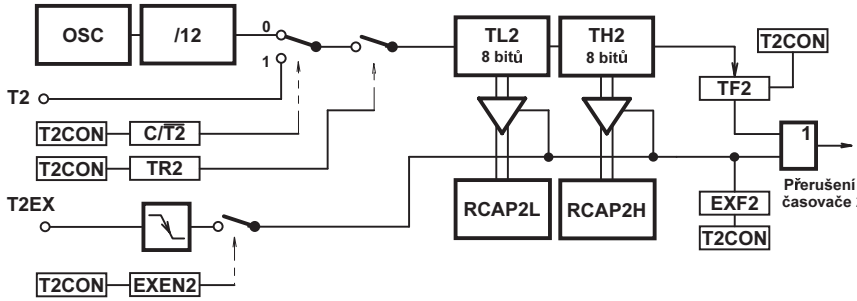
Procesory řady 8052 a některé další typy se odlišují od popsaného procesoru tím, že mají na čipu integrovánu rozšířenou vnitřní datovou paměť na kapacitu 256 bytů. Tím dochází k tomu, že specializované registry leží ve stejném adresovém prostoru jako rozšířená vnitřní paměť RAM. Aby nedocházelo při adresování ke kolizím, je rozšířená část paměti RAM od adres 80H do 7FH přístupná pouze přes instrukce nepřímého adresování (tj. MOV A,@R1, apod.) a specializované registry pouze přes přímé adresování. Velkou výhodou rozšířené paměti je též fakt, že do ní může být umístěn zásobník (ukazatel zásobníku SP od 80H do 7FH), čímž se rozšíří část přímo adresovatelné paměti RAM.

Druhou odlišností je existence třetího 16bitového čítače/časovače, který se označuje jako **časovač 2**. Tento časovač může sloužit ke generování přenosové rychlosti sériového kanálu, k zachycení časového údaje události (záchytný systém) nebo jako volně použitelný čítač s možností obvodového 16bitového přednastavení. Podobně jako časovače 0 a 1 může pracovat v režimu časovače nebo čítače vnějších událostí. Jednotlivé pracovní režimy se volí v registru T2CON *obr. 36* podle *tabulky 5*.

RCLK + TCLK	CP / $\overline{RL2}$	TR2	Funkce
0	0	1	16bitové přednastavení
0	1	1	16bitový záchytný
1	X	1	generátor přenosové rychlosti
X	X	0	časovač 2 zastaven

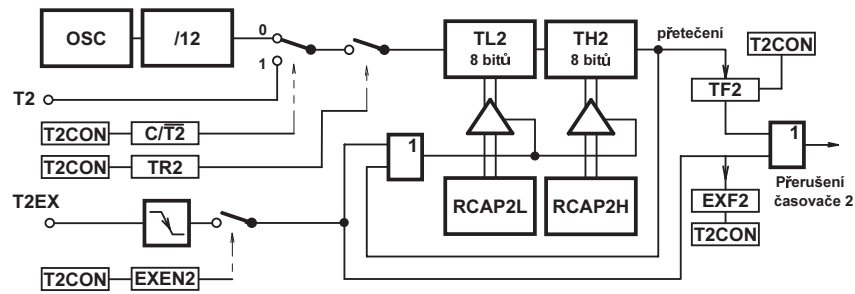
Tabulka 5

Možnosti záchytného systému u procesorů 8052 jsou zobrazeny na *obr. 33*. Existují dva režimy činnosti časovače 2 v závislosti na bitu EXEN2 v registru T2CON. Je-li EXEN2 =0, pak časovač 2 pracuje jako obyčejný 16bitový časovač nebo čítač (C / $\overline{T2}$), který při přetečení nastavuje příznakový bit přetečení (TF2) časovače 2. Od tohoto příznaku může být generováno přerušení jako v případě časovačů 0 a 1. Jediný rozdíl je v tom, že příznak musí být v obslužném podprogramu vyvolaného přerušení vynulován programově (není nulován automaticky jako u časovačů 0 a 1). Je-li EXEN2=1, pak se časovač 2 chová stejně jako v předchozím případě, ale při sestupné hraně na vstupu T2EX dojde k uložení současné hodnoty časovače 2 (TH2 a TL2) do RCAP2H a RCAP2L (speciální registry procesorů 8052). Současně s tím je nastaven i bit EXF2 v řídicím registru T2CON, od kterého může být, stejně jako od TF2, vyvoláno přerušení. Z popisu je zřejmé, že vstup T2EX lze využít i jako další vstup vnějšího přerušení.



Obr. 33 Záchytný režim časovače 2

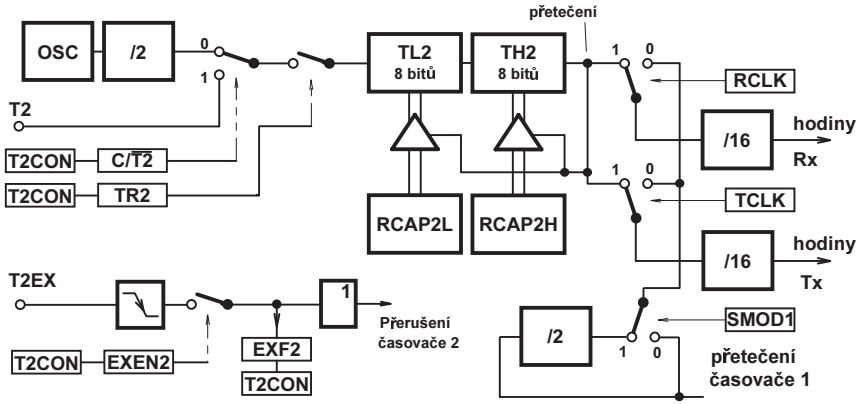
V pracovním režimu s přednastavením existují v závislosti na hodnotě bitu EXEN2 opět dvě možnosti *obr. 34*. Je-li EXEN2=0, pak se v případě přeplnění časovače 2 nastaví příznak TF2 a současně se registry časovače TH2 a TL2 naplní 16bitovou hodnotou, uloženou v registrech RCAP2H a RCAP2L. Je-li EXEN2=1, potom časovač 2 pracuje stejně jako v předcházejícím případě s tím, že dojde-li na vstupu T2EX k přechodu z log. 1 do log. 0, potom dojde k novému přednastavení časovače 2 (TH2=RCAP2H a TL2=RCAP2L).



Obr. 34 Režim časovače 2 s přednastavením

Režim, při kterém pracuje časovač 2 jako generátor přenosové rychlosti, se volí nastavením RCLK=1 nebo TCLK=1 a týká se pouze módu 1 a 3 sériového kanálu. Toto řešení umožňuje realizovat vysílání a příjem po sériovém kanálu na různých přenosových rychlostech, stejně jako uvolnění časovačů 0 a 1 pro jiné účely. Na rozdíl od časovače 1 je časovač 2 v tomto režimu inkrementován poloviční kmitočtem oscilátoru (nikoliv 1/12), což umožňuje nastavení přenosové rychlosti pro větší počet krystalů *obr. 35*. Přenosová rychlost je určena vztahem

$$\text{přenosová rychlost} = \frac{\text{kmitočet oscilátoru}}{32 * (65536 - \{\text{RCAP2H,RCAP2L}\})}$$



Obr. 35 Časovač 2 jako generátor přenosové rychlosti

Jednotlivé funkce časovače nebo čítače 2 se volí nastavením nebo vynulováním bitů v registru **T2CON** *obr. 36*. Jeho jednotlivé bity, které nejsou bitově přístupné, mají následující význam.

♣ **TF2** - Příznak přetečení časovače 2. Nastavuje se při přetečení a lze jej vynulovat pouze programově (i po vyvolání přerušení). TR2 nemůže být nastaven, je-li RCLK=1 nebo TCLK=1.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	Adresa RAM = D0H

Obr. 36 Rozložení bitů ovlivňujících časovač 2 v registru T2CON

♣ **EXF2** - Vnější příznak časovače 2. Příznak se nastavuje pro EXEN2=1 při změně signálu na vstupním vodiči T2EX z log. 1 na log. 0. Je-li povoleno přerušení od časovače 2, pak EXF2=1 způsobí jeho vyvolání. Příznak EXF2 se musí v obslužném programu nulovat programově.

♣ **RCLK** - Příznak využití časovače 2 jako generátoru hodin pro příjem sériového kanálu. Je-li nastaven, pak se k synchronizaci příjmu sériového kanálu v režimu 1 a 3 využívají impulzy z přetečení časovače 2. Je-li RCLK=0, pak k časování sériového kanálu je využíván časovač 1.

♣ **TCLK** - Příznak využití časovače 2 jako generátoru hodin pro vysílání sériového kanálu. Je-li nastaven, pak se k synchronizaci vysílání sériového kanálu

v režimu 1 a 3 se využívají impulzy z přetečení časovače 2. Je-li TCLK=0, pak k časování sériového kanálu je využíván časovač 1.

♣ **EXEN2** - Příznak povolení vnějšího ovlivnění časovače 2. Je-li EXEN2=1 a není časovač 2 využit jako generátor přenosové rychlosti, potom zachycení nové hodnoty (záchytný režim) nebo přednastavení časovače nastane při sestupné hraně na vývodu T2EX. Při EXEN2=0 časovač 2 ignoruje události na vývodu T2EX.

♣ **TR2** - Spuštění čítače/časovače. Programovým nastavením bitu (TR2=1) je zajištěno spuštění časovače 2.

♣ **C / $\overline{\text{T2}}$** - Volba režimu časovače (log. 0) nebo čítače vnějších událostí (log. 1). Čítač vnějších událostí reaguje na sestupnou hranu na vývodu T2.

♣ **CP / $\overline{\text{RL2}}$** - Příznak výběru režimu zachycení (Capture) nebo přednastavení (Reload flag). Nastavení příznaku dovoluje, aby se zachycení provedlo na sestupnou hranu signálu T2EX, pokud EXEN2=1. Je-li CP / $\overline{\text{RL2}}$ = 0 a EXEN2=1, potom na sestupnou hranu signálu T2EX dojde k přednastavení časovače 2. Je-li využit časovač pro sériový kanál (RCLK=1 nebo TCLK=1), potom bit CP / $\overline{\text{RL2}}$ je ignorován a časovač je automaticky při přetečení přednastaven na 16bitovou hodnotu, uloženou v registrech RCAP2H a RCAP2L.

Na novějších typech procesorů 8052 je přiřazen ještě jeden registr **T2MOD**, který umožňuje realizovat pomocí bitu **DCEN** dva režimy s přednastavením. V prvém, kdy DCEN=0, čítá čítač 2 nahoru *obr. 34*, v druhém umožňuje čítat nahoru i dolů. Směr čítání je řízen vstupním vývodem T2EX (nahoru=1, dolů=0). Pro směr čítání je přednastavitelná hodnota dána obsahem registrů RCAP2H a RCAP2L, pro směr čítání dolů je přednastavitelná hodnota FFFFH. Při přetečení nebo podtečení je nastaven příznak přetečení časovače 2 (bit TF2), od kterého může být vyvoláno přerušení. Zároveň je změněna i hodnota bitu EXF2, kterou lze využívat jako sedmnáctý bit obousměrného čítače. V módu obousměrného čítání časovače 2 nelze bit EXF2 využívat k žádosti o přerušení.

1.3.3. Klony procesoru 8051

V současné době je u nás na trhu obrovské množství procesorů založených na jádře procesoru 8051, které vyrábí firma Intel, Philips, Siemens a Atmel. Procesory můžeme rozdělit na procesory (kontroléry) v pouzdrech s 24 a 28 vývody s omezeným počtem bran, a na procesory rozšířené o další periferie a brány v pouzdrech s 44 až 84 vývody. Protože je obtížné najít další společné vlastnosti těchto obvodů uvedeme si výčet jejich vlastností. Prostor, na kterém měly být některé z těchto obvodů popsány, byl věnován nejnovějšímu nástupci procesoru 8051. Jedná se o obvod z nové generace MCS251 procesorů Intel, které již mají odlišnou architelturu. V jednom z módů je programově i obvodově slučitelný s procesorem 8052. V *tabulce 6* je přehled některých základních typů podle výrobců a barevně jsou označeny nejzajímavější typy.

Tabulka 6

Typ	Hod. MHz	ROM	RAM bytů	V/V 8 bitů	A/D bitů	Čas/ čítač	Čas. WD	Přer. úrov.	Serial	Za/Ko PWM	Pouzdro
INTEL											
80C31	33	0	128	4	–	2	–	5/2	UART	–	DIP40
80C51	33	4kB	128	4	–	2	–	5/2	UART	–	nebo
87C51	33	E4kB	128	4	–	2	–	5/2	UART	–	PLCC44
8xC51FA	12/16	8kB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
8xC51FB	12/16	16kB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
8xC51FC	12/16	32kB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
80C251SB	12/16	16kB	1kB	4	Cout	4	0+1	7/4	UART	5/5/5	PLCC44
PHILIPS											
Většina uvedených verzi je v provedení bez ROM (80), v provedení OTP (83), v provedení EPROM (87)											
87C52	24	E8kB	256	4	–	3	–		UART	1/0/0	PLCC44
87C54	24	E16kB	256	4	–	3	–		UART	1/0/0	nebo
87C58	24	E32kB	256	4	–	3	–		UART	1/0/0	PQFP44
87C550	16	E4kB	128	4	8/8	2	1	6/2	UART	–	PLCC44
87C524	12/20	E16kB	512	4	–	3	1	6/2	UART, I2C	–	PLCC44
83C552	16/24	8kB	256	5+1	8/10	3	0+1	15/2	UART, I2C	4/8/2	PLCC68
89C558	16	F32kB	1kB	5+1	8/10	3	0+1	15/2	UART, I2C	4/8/2	PQFP80
87C575	16	E8kB	256	4	4 kom.	3	1	7/2	UART	5-PCA	PLCC44
87C576	16	E8kB	256	4	8/10	3	1		UART, I2C	PCA	PLCC44
87C654	16/24	E16kB	256	4	–	2	–	6/2	UART, I2C	–	PLCC44
87C751	16	E2kB	64	2+3b	–	1	–	5/2	I2C	–	PLCC28
87C752	16	E2kB	64	2+5b	5/8	1	–	7/2	I2C	0/0/1	PLCC28
83C852	16	6kB EE2kB	256	4	–	2	–	5/2	UART	–	PLCC44

Poznámka: U procesorů firmy **Dallas** se strojový cyklus skládá ze čtyř hodinových cyklů. V důsledku toho jsou při stejném krystalu prakticky třikrát rychlejší, než procesory od firem Intel (vyjma 80C251), Atmel, Philips a Siemens. Díky tomu jsou procesory firmy Dallas svým výkonem srovnatelné s procesorem 80C251 v módu procesoru 8051.

Tabulka 6

Typ	Hod. MHz	ROM	RAM bytů	V/V 8 bitů	A/D bitů	Čas/ čítač	Čas. WD	Přer. úrov.	Serial	Za/Ko PWM	Pouzdro
SIEMENS											
SAB80C515 SAB80C535	12/16	8 –	256	6+1	8/8	3	1+1	12/4	UART	4/4/0	PLCC68
SAB80C517 SAB80C537	12/16	8 –	256	7+1,5	12/8	4	2+2	14/4	2xUART	5/21/5	PLCC84 PQFP100
ATMEL											
AT89C51	0–24	F4kB	256	4	–	2	–	5/2	UART	–	DIL40 nebo PLCC44
AT89C52	0–24	F8kB	256	4	Cout	3	–	8/2	UART	1/0/0	
AT89C53	0–24	F12kB	256	4	SPI Cout	3	1	9/2	UART	1/0/0	
AT89C55	0–24	F20kB	256	4	Cout	3	–	8/2	UART	1/0/0	
AT89C8252	0–24	F8kB EE1kB	256	4	SPI Cout	3	1	9/2	UART	1/0/0	
AT89C2051	0–24	F2kB	128	2	1xkom	2	–	5/2	UART	–	SOIC20
DALLAS											
DS80C320	25	–	256	4	–	3	1	13/2	2xUART	1/0/0	PLCC44
DS80C320	25	E16kB	256 +1kB	4	–	3	1	13/2	2xUART	1/0/0	PLCC44 TQFP44
DS80C320	25	E16kB	256 +1kB	4	RTC	3	1	14/2	2xUART	1/0/0	PLCC44 TQFP44

Vysvětlivky:

- ♣ ROM - programová paměť
  - xxkB programová paměť OTP pouze jednou programovatelná
  - ExxkB programová paměť EPROM
  - FxxkB programová paměť EEPROM nebo Flash
- ♣ x kom v kolonce A/D značí počet analogových komparátorů
- ♣ PCA - čítačem podporované programovatelné pole viz. 80C251
- ♣ většina obvodů má uveden modernější typ pouzdra (obvody do 44 se většinou vyrábí i pouzdře DIL)

Za/Ko/PWM - počet vstupů záchytného, výstupů komparačního a pulzně šířkových modulátorů

RTC - hodiny reálného času

426410

2. Instrukční soubor CPU 51

Instrukční soubor procesoru 8051 lze rozdělit do těchto sedmi skupin:

- **Přesunové instrukce** umožňují přesun 8bitové hodnoty z registru Rr a přímo nebo nepřímo adresované vnitřní datové paměti do střadače a obráceně **MOV**, přesun hodnoty z nebo do střadače z nepřímo adresované vnější datové paměti **MOVX**, přesun hodnoty do střadače z nepřímo adresované programové paměti **MOVC**, záměnu obsahu střadače s obsahem registru Rr a přímo nebo nepřímo adresované vnitřní datové paměti **XCH** a **XCHD**. Posledními instrukcemi, které lze zařadit do této skupiny jsou instrukce pro práci se zásobníkem **PUSH** pro uložení a **POP** pro vyjmutí hodnoty ze zásobníku z přímo adresovaného místa vnitřní datové paměti.
- **Aritmetické instrukce** umožňují zvětšení (inkrementace) **INC** a zmenšení (dekrementace) **DEC** střadače, registru a přímo nebo nepřímo adresovaného místa ve vnitřní datové paměti, 8bitový aritmetický součet střadače s registrem, přímo nebo nepřímo adresovaným místem ve vnitřní datové paměti nebo přímo uvedenou hodnotou bez příznaku přenosu **ADD** nebo s příznakem přenosu **ADDC**, 8bitový rozdíl střadače s příznakem přenosu s registrem, přímo nebo nepřímo adresovaným místem ve vnitřní datové paměti nebo přímo uvedenou hodnotou a dekadické korekce po binárním součtu dvou čísel v BCD kódu **DA**. Aritmetické instrukce obsahují i instrukce pro 8bitové násobení **MUL** a dělení **DIV** dvou hodnot bez znaménka uložených ve střadači a registru B a instrukci pro realizaci jednotkového doplňku střadače **CPL**.
- **Logické operace** umožňují logický součin **ANL**, součet **ORL** a operaci výhradně-nebo (EX-OR = neekvivalence) **XRL** střadače s registrem, přímo nebo nepřímo adresovaným místem ve vnitřní datové paměti nebo přímo uvedenou hodnotou nebo přímo adresovaným paměťovým místem a střadačem nebo přímo uvedenou hodnotou. Protože vstupně/výstupní brány jsou paměťově adresované, jsou v logických operacích obsaženy i operace s bránami.
- **Posunové instrukce** umožňují 8bitové a 9bitové logické rotace střadače doleva **RL** a **RLC** nebo doprava **RR** a **RRC** případně s příznakem přenosu (9 bitů). Mezi posunové instrukce můžeme zařadit i instrukci výměny horního a dolního půlbytu střadače **SWAP**.
- **Bitové operace** umožňují nastavit **SETB** nebo nulovat **CLR** kterýkoliv z 256 přímo adresovatelných bitů procesoru 8051. Bitové instrukce umožňují přesun **MOV**, logický součin **ANL** a logický součet **ORL** mezi přímo adresovaným bitem a příznakem přenosu C, který přejímá pro bitové operace funkci střadače.

- **Skokové instrukce** umožňují nepodmíněný dlouhý **LJMP** (16bitový) a krátký **AJMP** (11bitový) skok, podmíněné relativní skoky (7bitové) závislé na nulovosti a nenulovosti střadače, příznaku přenosu C nebo přímo adresovaném bitu ve vnitřní datové paměti. Umožňují pouze nepodmíněná dlouhá **LCALL** (16bitové) a krátké **ACALL** (11bitové) volání podprogramů a nepodmíněné návraty z podprogramů **RET** a obslužných podprogramů přerušení **RETI**.
- **Sdružené instrukce**, které v sobě obsahují dvě samostatné operace a urychlují tak realizaci programových smyček, umožňují dekrementovat registr nebo přímo adresované paměťové místo a realizovat relativní skok (7 bitů) při jeho nenulovosti (instrukce **DJNZ**) nebo porovnat střadač s přímo adresovaným paměťovým místem, střadač, registr nebo nepřímo adresované paměťové místo s přímo uvedenými daty a realizovat relativní skok (7 bitů) při jejich neshodě (instrukce **CJNE**).

SYNTAXE INSTRUKCE	POPIS INSTRUKCE		
ACALL adr11	PC(*10 ÷ 0*) ← adr 11		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Volání podprogramu uvnitř 2kB adresového prostoru (adresa je 11bitová). Instrukce uloží návratovou adresu do zásobníku s tím, že nejprve uloží nižší a potom vyšší slabiku. Volaný podprogram musí ležet uvnitř 2kB stránky, v které leží instrukce následující po instrukci ACALL.			
ADD A,<zdrojová slabika> Sčítání střadače se zdrojovou slabikou			
ADD A, Rr	(A) ← (A) + (Rr)		
ADD A, @Rr	(A) ← (A) + ((Rr)), kde r = 0,1		
	Ovlivňuje: C,AC,OV,P	Bytů: 1	Cyklů: 1
ADD A, adresa	(A) ← (A) + (adresa)		
ADD A, #data	(A) ← (A) + data		
	Ovlivňuje: C,AC,OV,P	Bytů: 2	Cyklů: 1
Instrukce přičte obsah adresované slabiky ke střadači a výsledek v něm ponechá. Adresovanou slabikou může být registr aktivní banky **Rr**, přímo uvedená **adresa** (8 bitů), nepřímo adresované paměťové místo registrem R0 nebo R1 (obsah registru určuje adresu místa jehož obsah se bude přičítat) nebo přímo uvedená hodnota **data**.			

ADDC A,<zdrojová slabika> Sčítání střadače se slabikou a příznakem přenosu			
ADDC A, Rr	$(A) \leftarrow (A) + (Rr) + (C)$		
ADDC A, @Rr	$(A) \leftarrow (A) + ((Rr)) + (C)$, kde $r = 0,1$		
	Ovlivňuje: C,AC,OV,P	Bytů: 1	Cyklů: 1
ADDC A, adresa	$(A) \leftarrow (A) + (\text{adresa}) + (C)$		
ADDC A, #data	$(A) \leftarrow (A) + \text{data} + (C)$		
	Ovlivňuje: C,AC,OV,P	Bytů: 2	Cyklů: 1
Instrukce přičte obsah adresované slabiky a příznakový bit přenosu C ke střadači a výsledek v něm ponechá.			
AJMP adr11	$PC(10 \div 0) \leftarrow \text{adr}\,11$		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Krátký nepodmíněný skok na adresu uvnitř 2kB stránky.			
ANL <cílová slabika>, <zdrojová slabika> Operace logického součinu mezi cílovou a zdrojovou slabikou.			
ANL A, Rr	$(A) \leftarrow (A)$ **AND** (Rr), kde $r = 0,1,\ldots,7$		
ANL A, @Rr	$(A) \leftarrow (A)$ **AND** $((Rr))$, kde $r = 0,1$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
ANL A, adresa	$(A) \leftarrow (A)$ **AND** (adresa)		
ANL A, #data	$(A) \leftarrow (A)$ **AND** data		
ANL adresa, A	$(\text{adresa}) \leftarrow (\text{adresa})$ **AND** (A)		
	Ovlivňuje: P - pro zápis do střadače	Bytů: 2	Cyklů: 1
ANL adresa, #data	$(\text{adresa}) \leftarrow (\text{adresa})$ **AND** data		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce ANL provede logický součin mezi odpovídajícími bity cílového a zdrojového bytu a výsledek uloží do cílové slabiky. Je-li cílovým bytem výstupní brána (přímá adresa), pak se operace provede mezi výstupním registrem a zdrojovým bytem (nikoliv vstupními signály).			

ANL C, <zdrojový bit>	Logický součin bitů		
ANL C, bit	$(C) \leftarrow (C)$ **AND** (bit)		
ANL C, /bit	$(C) \leftarrow (C)$ **AND** $\overline{(bit)}$		
	Ovlivňuje: C	Bytů: 2	Cyklů: 2
Logický součin příznaku přenosu C s přímo adresovaným bitem. Výsledek operace se uloží do příznaku C. Je-li před adresou bitu lomítko, potom hodnota bitu bude před operací negována. Adresovaný bit lze adresovat jenom **přímou adresou**.			
CJNE <cílová slabika>, <zdrojová slabika>,relativní adresa Porovnej a skoč, když se operandy nerovnají.			
CJNE A, adresa, relativní adresa	$(PC) \leftarrow (PC) + 3$ Je-li $(A) \neq$ (adresa) pak $(PC) \leftarrow (PC) +$ relativní adresa jinak $(PC) \leftarrow (PC) + 2$		
CJNE A, #data, relativní adresa	$(PC) \leftarrow (PC) + 3$ Je-li $(A) \neq$ data pak $(PC) \leftarrow (PC) +$ relativní adresa jinak $(PC) \leftarrow (PC) + 2$		
CJNE Rr, #data, relativní adresa	$(PC) \leftarrow (PC) + 3$ Je-li $(Rr) \neq$ data, kde $r = 0,1,\ldots,7$ pak $(PC) \leftarrow (PC) +$ relativní adresa jinak $(PC) \leftarrow (PC) + 2$		
CJNE @Rr, #data, relativní adresa	$(PC) \leftarrow (PC) + 3$ Je-li $((Rr)) \neq$ data, kde $r = 0,1$ pak $(PC) \leftarrow (PC) +$ relativní adresa jinak $(PC) \leftarrow (PC) + 2$		
	Ovlivňuje: C	Bytů: 3	Cyklů: 2
Instrukce porovná střadač nebo registr **Rr** nebo slabiku adresovanou registry R0,R1 se zdrojovou slabikou (adresa, data). V případě jejich nerovnosti provede relativní skok (maximálně o +127 nebo -128 bytů) na požadovanou adresu. Adresa skoku se vypočte přičtením relativního posunu k čítači instrukcí, který byl nejprve třikrát inkrementován v důsledku čtení prováděné instrukce. Je-li cílová slabika menší než zdrojová je zároveň nastaven příznak přenosu C=1, jinak ho vynuluje C=0.			
CLR A	$(A) \leftarrow 0$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
Instrukce vynuluje obsah střadače.			

CLR C	$(C) \leftarrow 0$		
	Ovlivňuje: C	Bytů: 1	Cyklů: 1
CLR bit	$(bit) \leftarrow 0$		
	Ovlivňuje: AC, F0, RS1, RS0, OV, P jen je-li adresován	Bytů: 2	Cyklů: 1
Instrukce vynuluje adresovaný bit.			
CPL A	$(A) \leftarrow \overline{(A)} = {}^{1}(A)$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
Instrukce neguje každý bit střadače a vytváří tak jeho jednotkový doplněk.			
CPL C	$(C) \leftarrow \textbf{NOT}\,(C) = \overline{(C)}$		
	Ovlivňuje: C	Bytů: 1	Cyklů: 1
CPL bit	$(bit) \leftarrow \textbf{NOT}\,(bit) = \overline{(bit)}$		
	Ovlivňuje: C, AC, F0, RS1, RS0, OV, P jen je-li adresován	Bytů: 2	Cyklů: 1
Instrukce neguje (invertuje) adresovaný bit.			
DA A	Dekadická korekce střadače		
	Ovlivňuje: C, AC, P	Bytů: 1	Cyklů: 1
Instrukce koriguje obsah střadače po binárním sčítání dvou dekadických čísel vyjádřených v BCD kódu tak, aby výsledek opět tvořil dvě čtyřbitová BDC čísla. Je-li hodnota na nižších čtyřech bitech >9 nebo AC=1, potom se ke střadači přičte hodnota 6. Je-li hodnota na vyšších čtyřech bitech >9 nebo C=1, potom se ke střadači přičte hodnota 60H.			
DEC slabika	Dekrementace registru nebo paměťového místa		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
DEC A	$(A) \leftarrow (A) - 1$		
DEC Rr	$(Rr) \leftarrow (Rr) - 1$ kde $Rr = 0,1,\ldots,7$		
DEC adresa	$(adresa) \leftarrow (adresa) - 1$		Bytů: 2
DEC @Rr	$((Rr)) \leftarrow ((Rr)) - 1$ kde $Rr = 0,1$		
Instrukce odečte od obsahu adresovaného paměťového místa hodnotu jedna. Po zmenšení hodnoty 00H dojde k podtečení na hodnotu FFH. Dekrementace obsahu výstupní brány zmenšuje obsah přečtený z registru brány a nikoliv ze vstupně/výstupních vodičů.			

DIV AB	$(A) \leftarrow$ celá čá st$(A)/(B)$ $(B) \leftarrow$ zbytek $(A)/(B)$		
	Ovlivňuje: OV,P a C=0	Bytů: 1	Cyklů: 4
Instrukce provádí celočíselné dělení obsahu střadače s obsahem registru B. Celá část podílu zůstává ve střadači, zbytek (nikoliv desetinná část) zůstává v registru B. Při dělení nulou se nastaví příznak přetečení OV=1.			
DJNZ Rr, relativní adresa	$(Rr) \leftarrow (Rr) - 1$, kde $r = 0,1,\ldots,7$ Je – li $(Rr) \neq 0$, pak $(PC) \leftarrow (PC) +$ relativní adresa, jinak $(PC) \leftarrow (PC) + 2$		
	Ovlivňuje: OV,P a C=0	Bytů: 2	Cyklů: 2
Instrukce odečte od adresového registru jedničku a zjistí zda výsledek je nulový. Je-li výsledek nenulový provede skok na definovanou adresu.			
DJNZ adresa, relativní adresa			
(adresa) $\leftarrow$ (adresa) – 1, Je – li (adresa) $\neq 0$ pak $(PC) \leftarrow (PC) +$ relativní adresa jinak $(PC) \leftarrow (PC) + 2$			
	Ovlivňuje: OV,P a C=0	Bytů: 3	Cyklů: 2
Instrukce odečte od adresového paměťového místa jedničku a zjistí zda výsledek je nulový. Je-li výsledek nenulový provede skok na definovanou adresu.			
INC slabika	Inkrementace registru nebo paměťového místa		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
INC A	$(A) \leftarrow (A) + 1$		
INC Rr	$(Rr) \leftarrow (Rr) + 1$, kde $r = 0,1,\ldots,7$		
INC adresa	(adresa) $\leftarrow$ (adresa) $+ 1$		Bytů: 2
INC @Rr	$((Rr)) \leftarrow ((Rr)) + 1$, kde $r = 0,1$		
Instrukce přičte od obsahu adresovaného paměťového místa hodnotu jedna. Po zvětšení hodnoty FFH dojde k přetečení na hodnotu 00H. Inkrementace obsahu výstupní brány zvětšuje obsah přečtený z registru brány a nikoliv ze vstupně/výstupních vodičů.			
INC DPTR	$(DPTR) \leftarrow (DPTR) + 1$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 2
Instrukce přičte jedničku k registrovému páru DPH a DPL, které vytváří 16-bitový ukazatel datové paměti DPTR. Dojde-li při přičítání k přetečení u registru DPL (FFH $\rightarrow$00H) potom je přičtena jednička k registru DPH. Přičítání jedničku k DPTR probíhá modulo 2^{16} (FFFFH +1$\rightarrow$0000H). Instrukce INC DPTR je jedinou 16-bitovou instrukcí v instrukčním souboru procesoru 8051.			

JB bit, relativní adresa	Je-li bit=1, potom skoč		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce testuje adresovaný bit a v případě jeho nastavení (log.1) provede skok na adresu kterou vypočte jako součet čítače instrukcí a relativní adresy.			
JBC bit, relativní adresa	Je-li bit=1, potom skoč a nuluj bit (bit←0)		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce testuje adresovaný bit a v případě jeho nastavení (log.1) provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy a vynuluje testovaný bit.			
JC relativní adresa	Je-li C=1, potom skoč		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce testuje příznak přenosu a v případě jeho nastavení (log.1) provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy.			
JMP @A+DPTR	$(PC) \leftarrow (A) + (DPTR)$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 2
Instrukce nepřímého nepodmíněného skoku na adresu určenou 16-bitovým součtem obsahu střadače (8 bitů bez znaménka) s obsahem ukazatele datové paměti DPTR.			
JNB bit, relativní adresa	Je-li bit=0, potom $(PC) \leftarrow (PC) +$ relativní adresa		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce testuje adresovaný bit a v případě jeho nulovosti (log.0) provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy.			
JNC relativní adresa	Je-li C=0, potom $(PC) \leftarrow (PC) +$ relativní adresa		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce testuje příznak přenosu a v případě jeho nulovosti (log.0) provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy.			
JNZ relativní adresa	Je-li $(A) \neq 0$, potom $(PC) \leftarrow (PC) +$ relativní adresa		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce testuje obsah střadače a v případě jeho nenulovosti provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy.			
JZ relativní adresa	Je-li $(A) = 0$, potom $(PC) \leftarrow (PC) +$ relativní adresa		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce testuje obsah střadače a v případě jeho nulovosti provede skok na adresu, kterou vypočte jako součet čítače instrukcí a relativní adresy.			

LCALL adr16	PC($15 \div 0$) ← adr 16		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce vykoná nepodmíněné volání podprogramu z přímo uvedené adresy ve svém druhém (vyšší byte) a třetím (nižší byte) bytu. Před uložením přečtené adresy do čítače instrukcí (PC) se uloží návratová adresa (současný stav (PC) = adresa následující instrukce) do zásobníku.			
LJMP adr16	PC($15 \div 0$) ← adr 16		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce vykoná nepodmíněný skok na adresu přímo uvedenou ve svém druhém (vyšší byte) a třetím (nižší byte) bytu. Adresa může ležet kdekoliv v 64kB adresovém prostoru.			
MOV <cílová slabika>, <zdrojová slabika> Přesun bytu z paměťového místa na jiné paměťové místo ve vnitřní datové paměti.			
MOV A, Rr	(A) ← (Rr), kde r = 0,1,…,7		
MOV A, @Rr	(A) ← ((Rr)), kde r = 0,1		
MOV Rr, A	(Rr) ← (A), kde r = 0,1,…,7		
MOV @Rr, A	((Rr)) ← (A), kde r = 0,1		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
MOV A, adresa	(A) ← (adresa)		
MOV A, #data	(A) ← data		
MOV Rr, #data	(Rr) ← data, kde r = 0,1,…,7		
MOV @Rr, #data	((Rr)) ← data, kde r = 0,1		
MOV adresa, A	(adresa) ← (A)		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 1
MOV Rr, adresa	(Rr) ← (adresa), kde r = 0,1,…,7		
MOV adresa, Rr	(adresa) ← (Rr), kde r = 0,1,…,7		
MOV @Rr, adresa	((Rr)) ← (adresa), kde r = 0,1		
MOV adresa, @Rr	(adresa) ← ((Rr)), kde r = 0,1		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
MOV adresa1, adresa2	(adresa1) ← (adresa2)		
MOV adresa, #data	(adresa) ← data		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2

Instrukce MOV přesune obsah zdrojové slabiky do cílové slabiky bez ovlivnění jakýchkoliv příznaků.

MOV <cílový bit>, <zdrojový bit>

Přesun hodnoty mezi přenosem C a daným bitem.		Ovlivňuje: ---	
MOV C, bit	(C) ← (bit)	Bytů: 2	Cyklů: 2
MOV bit, C	(bit) ← (C)	Bytů: 2	Cyklů: 2
MOV DPTR, #data16	(DPTR) ← data16 tj. (DPH) = data(*15 ÷ 8*) (DPL) = data(*7 ÷ 0*)		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2

Instrukce přesune obsah druhého a třetího bytu instrukce do ukazatele dat (DPTR). Druhý byte do DPH a třetí byte do DPL.

MOVC A, @A+<bázový registr>	Přesuň byte z paměti programu		
MOVC A, @A+DPTR	(A) ← ((A) + (DPTR))		
MOVC A, @A+PC	(A) ← ((A) + (PC))		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 2

Instrukce přesune byte z programové paměti (operační kód nebo konstantu) do střadače. Adresa místa, jehož obsah se přesouvá, získáme jako 16-bitový součet obsahu střadače (8 bitů) a ukazatele dat DPTR nebo čítače instrukcí PC. Pro případ čítače instrukcí je jeho obsah před provedením instrukce inkrementován (ukazuje na následující instrukci).

MOVX <cílová slabika>, <zdrojová slabika>

Přesun byte z/do vnější datové paměti

MOVX A, @DPTR	(A) ← ((DPTR))		
MOVX @DPTR, A	((DPTR)) ← (A)		
MOVX A, @Rr	(A) ← ((Rr)), kde r = 0,1		
MOVX @Rr, A	((Rr)) ← (A), kde r = 0,1		
	Ovlivňuje: P - pro zápis do střadače	Bytů: 1	Cyklů: 2

Instrukce přesune byte ze/do střadače do/z vnější paměti dat. Instrukce mohou využívat 16-bitovou nebo 8-bitovou nepřímou adresu. V prvém případě se vysílá adresa uložená v DPTR na bránu P2 (DPH) a bránu P0 (DPL). V případě druhém se vysílá na bránu P0 adresa uložená v registru R0 nebo R1 a na bráně P2 zůstává hodnota naposledy zapsaná.

MUL AB	(A)←nižší byte (A)*(B), (B)←vyšší byte (A)*(B)		
	Ovlivňuje: OV,P a C=0	Bytů: 1	Cyklů: 4
Instrukce vynásobí dvě osmibitová čísla bez znaménka uložená ve střadači a registru B. Je-li součin větší než hodnota 255 (FFH), nastaví se příznakový bit přetečení OV=1. V opačném případě jej vynuluje.			
NOP	Prázdná operace		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
Kromě čítače instrukcí neovlivňuje instrukce žádné registry a příznaky.			
ORL <cílová slabika>, <zdrojová slabika> Operace logického součtu mezi cílovou a zdrojovou slabikou.			
ORL A, Rr	$(A) \leftarrow (A)$ **OR** (Rr), kde $r = 0,1,\ldots,7$		
ORL A, @Rr	$(A) \leftarrow (A)$ **OR** $((Rr))$, kde $r = 0,1$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
ORL A, adresa	$(A) \leftarrow (A)$ **OR** (adresa)		
ORL A, #data	$(A) \leftarrow (A)$ **OR** data		
ORL adresa, A	(adresa) ← (adresa) **OR** (A)		
	Ovlivňuje: P - pro zápis do střadače	Bytů: 2	Cyklů: 1
ORL adresa, #data	(adresa) ← (adresa) **OR** data		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce ORL provede logický součet mezi odpovídajícími bity cílového a zdrojového bytu a výsledek uloží do cílové slabiky. Pro operaci s výstupní bránou (přímá adresa) se operace provede mezi výstupním registrem a zdrojovým bytem a nikoliv vstupními signály.			
ORL C, <zdrojový bit>	Logický součet bitů		
ORL C, bit	$(C) \leftarrow (C)$ **OR** (bit)		
ORL C, /bit	$(C) \leftarrow (C)$ **OR** $\overline{(bit)}$		
	Ovlivňuje: C	Bytů: 2	Cyklů: 2
Logický součet příznaku přenosu C s přímo adresovaným bitem. Výsledek operace se uloží do příznaku C. Je-li před adresou bitu lomítko, potom hodnota bitu bude před operací negována. Adresovaný bit lze adresovat jenom **přímou adresou**.			
POP adresa	(adresa) ← ((SP)), (SP) ← (SP) – 1		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2

Instrukce vyzvedne obsah vrcholu zásobníku a uloží jej na adresované paměťové místo. Pak odečte od ukazatele zásobníku jedničku.			
PUSH adresa	$(SP) \leftarrow (SP)+1, \quad ((SP)) \leftarrow (\text{adresa})$		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce přičte k ukazateli zásobníku jedničku a potom uloží obsah adresovaného místa do vrcholu zásobníku (zásobník je vždy ve vnitřní datové paměti).			
RET	$(PC(15 \div 8)) \leftarrow ((SP)), \quad (SP) \leftarrow (SP)-1$ $(PC(7 \div 0)) \leftarrow ((SP)), \quad (SP) \leftarrow (SP)-1$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 2
Instrukce návratu z podprogramu vyzvedne ze zásobníku dva byty a uloží je do čítače instrukcí (PC). Nejprve vyjme vyšší byte a potom nižší byte a odečte od ukazatele zásobníku hodnotu dvě.			
RETI	$(PC(15 \div 8)) \leftarrow ((SP)), \quad (SP) \leftarrow (SP)-1$ $(PC(7 \div 0)) \leftarrow ((SP)), \quad (SP) \leftarrow (SP)-1$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 2
Instrukce návratu z obslužného podprogramu přerušení vyzvedne ze zásobníku dva byty a uloží je do čítače instrukcí (PC). Nejprve vyjme vyšší byte a potom nižší byte a odečte od ukazatele zásobníku hodnotu dvě. Nakonec se povolí přijetí žádostí o přerušení se stejnou nebo nižší úrovní priority. Stavové slovo (jako 8048) se automaticky neobnovuje.			
RL A	$(A(n+1)) \leftarrow (A(n)), \quad \text{pro } n = 0,1,\ldots,6$ $(A(0)) \leftarrow (A(7))$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
Instrukce osmibitové logické rotace střadače o jednu pozici vlevo.			
RLC A	$(A(n+1)) \leftarrow (A(n)), \quad \text{pro } n = 0,1,\ldots,6$ $(A(0)) \leftarrow (C), \quad (C) \leftarrow (A(7))$		
	Ovlivňuje: C, P	Bytů: 1	Cyklů: 1
Instrukce devítibitové logické rotace střadače a příznaku přenosu o jednu pozici vlevo.			
RR A	$(A(n)) \leftarrow (A(n+1)), \quad \text{pro } n = 0,1,\ldots,6$ $(A(7)) \leftarrow (A(0))$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
Instrukce osmibitové logické rotace střadače o jednu pozici vpravo.			
RRC A	$(A(n)) \leftarrow (A(n+1)), \quad \text{pro } n = 0,1,\ldots,6$ $(A(7)) \leftarrow (C), \quad (C) \leftarrow (A(0))$		
	Ovlivňuje: C, P	Bytů: 1	Cyklů: 1

Instrukce devítibitové logické rotace střadače a příznaku přenosu o jednu pozici vpravo.			
SETB <bit>	Nastav bit		
SETB C	$(C) \leftarrow 1$		
	Ovlivňuje: C	Bytů: 1	Cyklů: 1
SETB bit	$(bit) \leftarrow 1$		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 1
Instrukce nastaví přímo adresovaný bit na log.1.			
SJMP relativní adresa	$(PC) \leftarrow (PC) + 2$, $(PC) \leftarrow (PC) + \text{relativní adresa}$		
	Ovlivňuje: ---	Bytů: 2	Cyklů: 2
Instrukce realizuje krátký nepodmíněný skok na definovanou adresu, která se vypočítá jako součet čítače instrukcí ukazujícího na adresu následující instrukce a posunu (-128;127) určeného relativní adresou.			
SUBB A,<zdrojová slabika> Odečítání slabiky a příznaku přenosu od střadače.			
SUBB A, Rr	$(A) \leftarrow (A) - (Rr) - (C)$, kde $r = 0,1,\ldots,7$		
SUBB A, @Rr	$(A) \leftarrow (A) - ((Rr)) - (C)$, kde $r = 0,1$		
	Ovlivňuje: C,AC,OV,P	Bytů: 1	Cyklů: 1
SUBB A, adresa	$(A) \leftarrow (A) - (\text{adresa}) - (C)$		
SUBB A, #data	$(A) \leftarrow (A) - \text{data} - (C)$		
	Ovlivňuje: C,AC,OV,P	Bytů: 2	Cyklů: 1
Instrukce odečte obsah adresované slabiky od střadače včetně příznaku přenosu (výpůjčky) a výsledek v něm ponechá. Adresovanou slabikou může být registr aktivní banky **Rr**, přímo uvedená **adresa,** nepřímo adresované paměťové místo registrem R0 nebo R1 (obsah registru určuje adresu místa jehož obsah budu odečítat) nebo přímo uvedená hodnota **data**. Je-li při výpočtu vyžadována v bitu b_7 výpůjčka, je příznakový bit C nastaven, v opačném případě je vynulován.			
SWAP A	$(A) \leftarrow (A(3 \div 0) * 16 - A(7 \div 4))$		
	Ovlivňuje: ---	Bytů: 1	Cyklů: 1
Instrukce prohodí navzájem obsah nižšího a vyššího "půlbytu" střadače. Instrukce je shodná s osmibitovou rotací o čtyři bity vpravo nebo vlevo.			

XCH A,<slabika>	Zaměň obsah střadače a adresované slabiky		
XCH A, Rr	$(A) \leftrightarrow (Rr)$		
XCH A, @Rr	$(A) \leftrightarrow ((Rr)), \quad r = 0,1$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
XCH A, adresa	$(A) \leftrightarrow (\text{adresa})$		
	Ovlivňuje: P	Bytů: 2	Cyklů: 1
Instrukce vymění obsah střadače a určeného registru nebo adresovaného paměťového místa.			
XCHD A,@Rr	$A(3 \div 0) \leftrightarrow (Rr(3 \div 0)), \quad kde\ r = 0,1$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
Instrukce prohodí navzájem obsah nižší "půlbyte" střadače s nepřímo adresovaným paměťovým místem vnitřní paměti RAM.			
XRL <cílová slabika>, <zdrojová slabika> Operace neekvivalence (EX-OR) mezi cílovou a zdrojovou slabikou.			
XRL A, Rr	$(A) \leftarrow (A) \oplus (Rr), \quad kde\ r = 0,1,\ldots,7$		
XRL A, @Rr	$(A) \leftarrow (A) \oplus ((Rr)), \quad kde\ r = 0,1$		
	Ovlivňuje: P	Bytů: 1	Cyklů: 1
XRL A, adresa	$(A) \leftarrow (A) \oplus (\text{adresa})$		
XRL A, #data	$(A) \leftarrow (A) \oplus \text{data}$		
XRL adresa, A	$(\text{adresa}) \leftarrow (\text{adresa}) \oplus (A)$		
	Ovlivňuje: P - pro zápis do střadače	Bytů: 2	Cyklů: 1
XRL adresa, #data	$(\text{adresa}) \leftarrow (\text{adresa}) \oplus \text{data}$		
	Ovlivňuje: ---	Bytů: 3	Cyklů: 2
Instrukce XRL provede operaci EX-OR odpovídajícími hodnotami cílového a zdrojového bytu a výsledek uloží do cílové slabiky. Je-li cílovým bytem výstupní brána (přímá adresa), potom se operace provede mezi výstupním registrem a zdrojovým bytem (nikoliv vstupními signály).			

3. Příklady použití CPU 51

Příklad 1

Navrhněte podprogram pro převod binárního čísla, uloženého ve střadači, na dekadické číslo v BCD kódu. Výsledné číslo uložte na paměťová místa se symbolickými názvy STOVKY a DESJED.

Protože je procesor 8051 vybaven instrukcí dělení, můžeme převod realizovat dělením hodnoty uložené ve střadači hodnotou 100. Celočíselný zbytek z prvního dělení vydělíme hodnotou 10 a celočíselný zbytek po druhém dělení představuje jednotky hledaného čísla.

```
        ;NIČÍ:  A, B, Příznaky      Doba:  18 strojových cyklů

BINBCD: MOV     B,#100D     ; A=BIN  B=100
        DIV     AB          ; A-urči počet stovek B-zbytek celočísel. dělení
        MOV     STOVKY,A    ; ulož stovky
        MOV     A,#10D
        XCH     A,B         ; A=zbytek, B=10
        DIV     AB
        SWAP    A           ; přesuň desítky do vyšší půlslabiky
        ADD     B           ; přičti jednotky
        MOV     DESJED,A
        RET

        DSEG                ; Rezervace datového prostoru

        ORG 30H
DESJED: DS      1H          ; paměť desítek (bity 7÷4) a jednotek (3÷0)
STOVKY: DS      1H          ; paměť pro uložení stovek (3÷0)
```

Příklad 2

Navrhněte stejný program jako v příkladě 1 bez použití instrukce dělení a metody postupného odečítání mocnin základu 10.

Není-li procesor vybaven instrukcí dělení nebo převáděné číslo je dlouhé, pak lze k zajištění stejného převodu využít algoritmu vycházejícího z tzv. Hornerova schématu. Každé číslo ve dvojkové soustavě můžeme vyjádřit výrazem

$$N = ((a_n.2 + a_{n-1}).2 + a_{n-2}).2 + ... + a_0$$

který vyjadřuje hledané číslo jako postupný součet dvou stejných čísel (.2) s hodnotou 0 nebo 1 ($a_{n-i} \in (0,1)$). Jsou-li dvě čísla v BCD formátu (např.0), potom na jejich součet s přičtením hodnoty 0 nebo 1 může být aplikována instrukce deka-

dické korekce, která koriguje rozdíl mezi součtem dekadických čísel v binární sčítačce (v procesoru) a dekadické sčítačce (přenos do dalšího řádu je generován po dosažení modulu 10). Navržený algoritmus má proti metodě postupného odečítání mocnin základu konstantní dobu trvání nezávislou na převáděném čísle. Při převodu využijeme registry R2 a R1 jako 16bitový posuvný registr obsahující na konci algoritmu výsledek, registr R3 jako dočasnou paměť převáděného čísla a registr R4 jako počitadlo cyklů (posunů).

```
                ;NIČÍ:  A, R1, R2, R3, R4, Příznaky
                ;Doba:  24 + 8*13 strojových cyklů
BINBCD:   MOV     R2,#OH      ; nuluj výsledek
          MOV     R1,#OH
          MOV     R3,A        ; paměť převáděného čísla
          MOV     R4,#8D      ; počet posunů
CYKL:     MOV     A,R3
          ADD     A           ; CY = bit a7
          MOV     R3,A        ; uchovej mezivýsledek
          MOV     A,R1
          ADDC    A,ACC       ; součet čísel R2,R1 + a_n-i ∈(0,1)
          DAA                 ; dekadická korekce
          MOV     R1,A
          MOV     A,R2
          ADDC    A,ACC
          DA      A
          MOV     R2,A
          DJNZ    R4,CYKL
          MOV     DESJED,R1
          MOV      STOVKY,R2
          RET
```

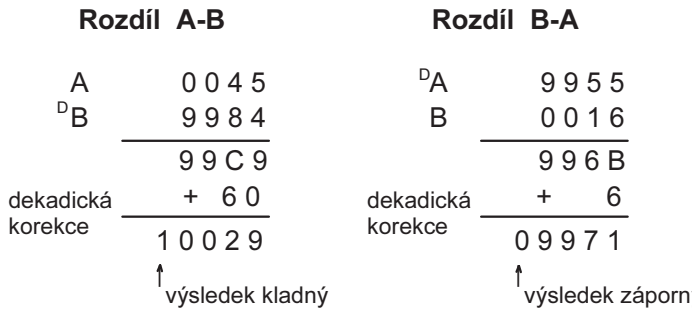
Příklad 3

Navrhněte podprogram realizující rozdíl dvou BCD čísel z intervalu 0÷999.

Protože většina mikroprocesorů neumožňuje aplikovat dekadickou korekci po rozdílu čísel, můžeme k řešení problému použít tyto metody:

a) konverze BCD->BIN->**odečtení**->konverze BIN->BCD - *rozsáhlý program*
b) využitím jednotkového nebo dvojkového dekadického doplňku *obr. 37*

Analogicky s definicí dvojkového doplňku můžeme vytvořit dvojkový dekadický doplněk $^{2D}B = 10^n - B$. Rozdíl čísel A-B pak můžeme realizovat jako součet čísel $A + {}^{2D}B$. Je-li A>B, potom součet $A + {}^{2D}B \geq 10^n$ ($A - B = A + {}^{2D}B - 10^n$). Pro získání správné hodnoty rozdílu A–B potřebujeme odečíst hodnotu 10^n. Protože se jedná



Obr. 37

o jedničku na bitu vlevo od nejvyššího bitu v dalším zpracování ji neuvažujeme. Je-li $A \leq B$, potom pro rozdíl můžeme psát $A - B = 10^n + A - B = 10^n - R = {}^{2D}R$. Je-li rozdíl záporný, pak získáváme přímo jeho dekadický doplněk. Jako příklad si ukážeme oba případy při rozdílu dvou BCD čísel A=45 a B=16 viz *obr. 37*. Protože rozdíl BCD čísel je takto realizován jako součet čísel, může být na něj uplatněna dekadická korekce.

Předpokládejme, že číslo A je uloženo v registrech R1 (vyšší), R2 (nižší), číslo B je uloženo v registrech R3 a R4 a výsledek má být uložen do registrů R6 (vyšší), R7 (nižší).

```
                ;NIČÍ:  A, R1,R2,R3,R4,R6,R7, příznaky
                ;Doba:  22 strojových cyklů
BCDROZD: MOV    A,#99H      ; Vytvoření dvojkového dekadického doplňku
         SUBB   A,R3        ; čísla B, 2DB=9999–B+1
         MOV    R3,A
         MOV    A,#99H
         SUBB   A,R4
         ADD    A,#1        ; nebo INC A
         DA     A           ; A-dvojkový dekadický doplněk
         MOV    R4,A
         CLR    A           ; neovlivňuje CY
         ADDC   A,R3
         DA     A
         MOV    R3,A        ; R3,R4 dvojkový dekadický doplněk čísla B
         MOV    A,R2        ; A +2DB
         ADD    A,R4
         DA     A
         MOV    R7,A
         MOV    A,R3
         ADD    A,R1
         DA     A
         MOV    R6,A        ; R6,R7 = A-B v BCD formátu. Je-li výsledek
         RET                ; záporný, pak je v dekadickém doplňku
                            ; a bity (7÷4) registru R6 obsahují hodnotu 9.
```

Příklad 4

Navrhněte program realizující číslicový filtr prvního řádu popsaný diferenční rovnicí $y_n = 0{,}73.x_n + 0{,}16.y_{n-1}$. Vstupní hodnota x_n i výstupní hodnota y_n je vyjádřena 7bitovým číslem bez znaménka s řádovou čárkou mezi 8 a 7 bitem. Součet dílčích součinů realizujte 16bitově a teprve potom proveďte omezení výsledné hodnoty.

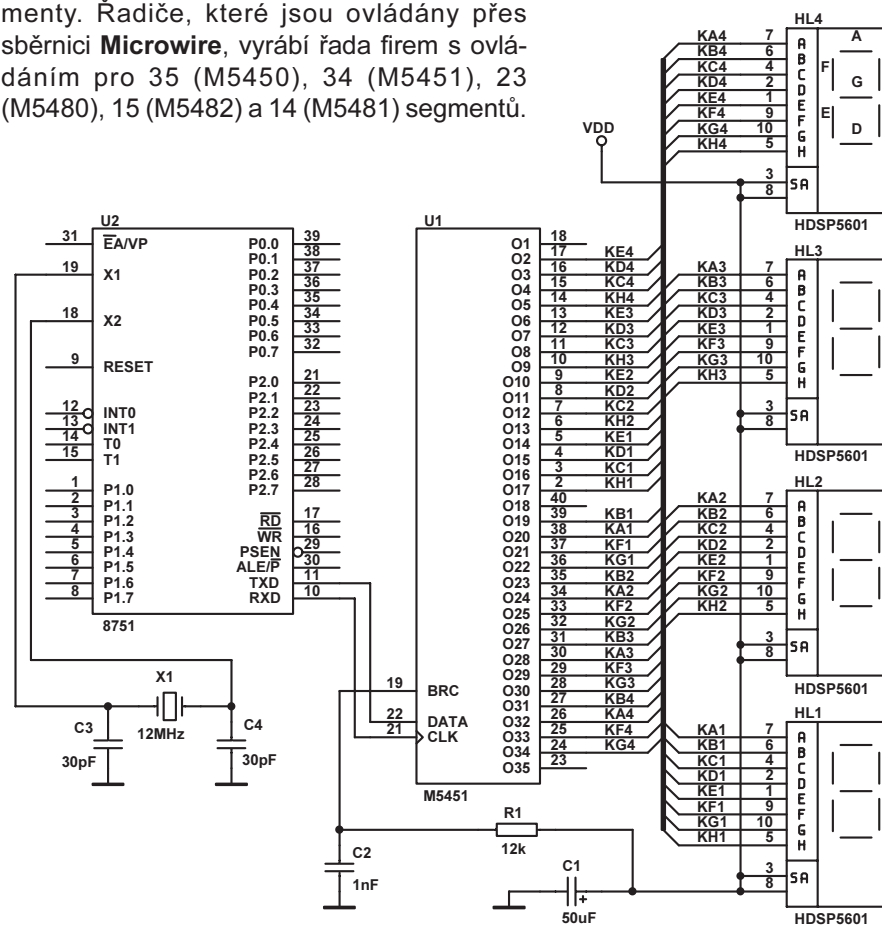
Nejprve musíme zvážit v jakém formátu vyjádříme koeficienty filtru. Předpokládejme, že násobicí konstanty budou též vyjádřeny 7bitovým číslem bez znaménka (řádová čárka převáděných konstant bude mezi 8 a 7 bitem). Převedeme konstanty do binární soustavy 0.73 = 0.1011101 = 5DH, 0.16 = 0.0010100 = 14H. Při řešení budeme předpokládat , že výstup z A/D převodníku je přiveden na bránu P1, vývod procesoru P3.7 slouží ke spouštění dalšího převodu (náběžnou hranou) a k bráně P0 je připojen vstup D/A převodníku.

```
        ;NIČÍ:  A, B, R1,R2, příznaky
        ;Doba:  34 + ZPOZD  strojových cyklů
        MOV     R1,#0H      ; počáteční podmínka y(n-1) = 0
FILTR:  MOV     A,P1        ; čti novou hodnotu x(n)
        ANL     P3,#7FH     ; generuj STB pro další převod
        ORL     P3,#80H
        MOV     B,#05DH     ; B = 0.73
        MUL     AB          ; B,A = 0,73 * x(n)
        MOV     R2,B
        XCH     A,R1        ; A = y(n-1)
        MOV     B,#14H      ; B = 0.16
        MUL     AB          ; B,A = 0,16 * y(n-1)
        ADD     A,R1
        MOV     R1,A
        MOV     A,B
        ADDC    A,R2        ; Součet  0,73 * x(n) + 0,16 * y(n-1)
        XCH     A,R1
        ADD     A,ACC       ; Korekce posunu řádové čárky  = 2*(B,A)
        XCH     A,R1
        ADDC    A,ACC
        MOV     R1,A        ; ulož y(n-1) s řádovou čárkou mezi 8 a 7 bitem
        MOV     P0,A        ; výstup na D/A převodník
        CALL    ZPOZD       ; Zpoždění zajišťující požadovaný vzorkovací
        AJMP    FILTR       ; kmitočet
```

Příklad 5

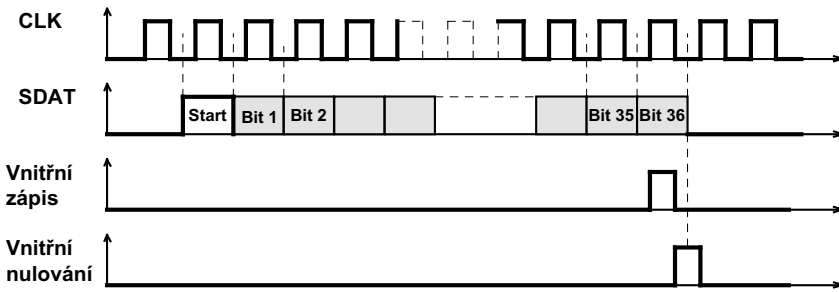
Připojte k procesoru 8751 čtyřmístný sedmisegmentový displej a navrhněte program pro jeho obsluhu.

Nejprve musíme rozhodnout jakým způsobem budeme sedmisegmentový displej ovládat. Jedním z řešení je dynamický zobrazovač, u kterého v daném okamžiku svítí jenom segmenty jednoho displeje. Jednotlivé displeje jsou periodicky (2,5 ms) aktivovány sepnutím anodových spínačů. Hlavní nevýhoda tohoto řešení není ani v nutnosti častého přepínání, které vytváří vyšší úrovně rušení, ale v potřebě zvyšovat špičkový proud jednotlivými segmenty pro zvětšující se počet displejů. Druhým, a v současné době stále častěji používaným řešením, je využití sériově ovládaného inteligentního řadiče sedmisegmentových displejů. Předností řadiče je statické řízení jednotlivých segmentů s možností analogového řízení jejich proudu. Řadič obsahuje posuvný registr, který je po naplnění přepsán do registru ovládajícího jednotlivé segmenty. Řadiče, které jsou ovládány přes sběrnici **Microwire**, vyrábí řada firem s ovládáním pro 35 (M5450), 34 (M5451), 23 (M5480), 15 (M5482) a 14 (M5481) segmentů.



Obr. 38 Připojení sériového řadiče displeje k procesoru 8751

Na *obr. 38* je zobrazeno jedno z možných zapojení čtyř sedmisegmentových displejů se společnou anodou k řadiči M5451 a jeho připojení k bráně P3 procesoru 8751. Připojení jednotlivých katod displejů k řadiči je ovlivněno buď dosažením jednoduchého ovládacího programu nebo realizací jednoduchého plošného spoje (případ z *obr. 38*). Program pro obsluhu bude koncipován tak, že zobrazovaný znak bude mít přiřazen kód. Program přijatý kód znaku daného displeje převede na číslo, které určuje svítící (log. 1) a nesvítící (log. 0) segmenty. Informace o jednotlivých segmentech jsou dekódovány na posloupnost odpovídající propojení katod segmentů k řadiči displeje a postupně sériově vysílány do řadiče viz *obr. 39.*



Obr. 39 Časování při zápisu do řadiče M5451 (sběrnice Microwire)

```
POCZNAK  EQU     36              ;Maximální počet zobrazovaných znaků
                                 ;Připojení displeje
SDAT     EQU     0B0H            ;P3.1
CLK      EQU     0B1H            ;P3.0
```

;PODPROGRAM DISPLEJ

;Podprogram pro obsluhu 4místného displeje V2.0. Zobrazované znaky jsou
;uloženy v paměťových místech **dis4,dis3,dis2,dis1**, polohu desetinné tečky
;určuje paměťové místo **tečka** (0 až 4). Nula = není desetinná tečka.

```
         ;NICI: A, R0, DPTR, příznaky, dis4, dis3, dis2, dis1
DISPLEJ: MOV     A,dis4          ; Ochrana displeje proti velkým kódům
         CLR     C
         SUBB    A,#POCZNAK; Je-li disX>=POCZNAK, pak zobrazená
         JC      O1              ; hodnota bude prázdná (kód 0AH)
         MOV     dis4,#0AH       ; Špatný kód nezobrazuj
O1:      MOV     A,dis3
         CLR     C
         SUBB    A,#POCZNAK
         JC      O2
         MOV     dis3,#0AH
O2:      MOV     A,dis2
         CLR     C
```

```
        SUBB    A,#POCZNAK
        JC      O3
        MOV     dis2,#0AH
O3:     MOV     A,dis1
        CLR     C
        SUBB    A,#POCZNAK
        JC      O4
        MOV     dis1,#0AH
O4:     MOV     A,dis4          ; Dekódování zobrazovaných znaků
        MOV     DPTR,#TABZNAK   ; Nastav ukazatel tabulky znaků
        MOVC    A,@A+DPTR
        MOV     dis4,A          ; Ulož dekódovaný 4 znak
        MOV     A,dis3          ; Dekóduj třetí znak
        MOVC    A,@A+DPTR
        MOV     dis3,A
        MOV     A,dis2          ; Dekóduj druhý znak
        MOVC    A,@A+DPTR
        MOV     dis2,A
        MOV     A,dis1          ; Dekóduj první znak
        MOVC    A,@A+DPTR
        MOV     dis1,A
        CLR     SDAT            ; Smazání displeje
        MOV     R0,#02DH
V0:     SETB    CLK
        CLR     CLK             ; Generuj hodinový impulz
        DJNZ    R0,V0           ; Opakuj R0-krát
                                ; Připojení segmentů displejů k řadiči
        CLR     A               ; Nuluj paměť generované sériové
        MOV     tisk,A          ; posloupnosti
        MOV     A,dis4          ; Vytvoření prvních dílčích 8 bitů
        JNB     ACC.4,S1        ; Má příslušný segment svítit
        XRL     tisk,#080H      ; Je-li ACC.4=1, pak 8 bit = 1
S1:     JNB     ACC.3,S2
        XRL     tisk,#040H
S2:     JNB     ACC.2,S3
        XRL     tisk,#020H
S3:     MOV     A,tecka
        CJNE    A,#04H,S4
        XRL     tisk,#010H
S4:     MOV     A,dis3
        JNB     ACC.4,S5
        XRL     tisk,#08H
S5:     JNB     ACC.3,S6
        XRL     tisk,#04H
S6:     JNB     ACC.2,S7
        XRL     tisk,#02H
S7:     MOV     A,tecka
        CJNE    A,#03H,S8
        XRL     tisk,#01H       ; Prvních 8 bitů připraveno
```

```
S8:     SETB    SDAT        ; Začátek posloupnosti =1
        SETB    CLK
        CLR     CLK         ; Hodinový impulz
        SETB    CLK
        CLR     CLK         ; Nevyužitý vývod
        MOV     A,tisk      ; Vysílání prvních 8 bitů
        MOV     R0,#08H
V1:     ADD     A,ACC       ; Vysílaný bit v příznaku přenosu C
        MOV     SDAT,C
        SETB    CLK
        CLR     CLK         ; Hodinový impulz
        DJNZ    R0,V1
        MOV     tisk,#0H    ; Nuluj dílčí paměť
        MOV     A,dis2      ; Vytvoření druhých dílčích 8 bitů
        JNB     ACC.4,S21
        XRL     tisk,#080H
S21:    JNB     ACC.3,S22
        XRL     tisk,#040H
S22:    JNB     ACC.2,S23
        XRL     tisk,#020H
S23:    MOV     A,tecka
        CJNE    A,#02H,S24
        XRL     tisk,#010H
S24:    MOV     A,dis1
        JNB     ACC.4,S25
        XRL     tisk,#08H
S25:    JNB     ACC.3,S26
        XRL     tisk,#04H
S26:    JNB     ACC.2,S27
        XRL     tisk,#02H
S27:    MOV     A,tecka
        CJNE    A,#01H,S28
        XRL     tisk,#01H
S28:    MOV     A,tisk
        MOV     R0,#08H     ; Vysílání druhých 8 bitů
V2:     ADD     A,ACC       ; Vysílaný bit v příznaku přenosu C
        MOV     SDAT,C
        SETB    CLK
        CLR     CLK         ; Hodinový impulz
        DJNZ    R0,V2
        SETB    CLK
        CLR     CLK         ; Nevyužitý vývod
        MOV     tisk,#0H    ; Nuluj dílčí paměť
        MOV     A,dis1      ; Vytvoření třetích dílčích 8 bitů
        JNB     ACC.1,S31
        XRL     tisk,#080H
S31:    JNB     ACC.0,S32
        XRL     tisk,#040H
S32:    JNB     ACC.5,S33
```

```
        XRL     tisk,#020H
S33:    JNB     ACC.6,S34
        XRL     tisk,#010H
S34:    MOV     A,dis2
        JNB     ACC.1,S35
        XRL     tisk,#08H
S35:    JNB     ACC.0,S36
        XRL     tisk,#04H
S36:    JNB     ACC.5,S37
        XRL     tisk,#02H
S37:    JNB     ACC.6,S38
        XRL     tisk,#01H
S38:    MOV     A,tisk          ; Vysílání třetích 8 bitů
        MOV     R0,#08H
V3:     ADD     A,ACC           ; Vysílaný bit v příznaku přenosu C
        MOV     SDAT,C
        SETB    CLK
        CLR     CLK             ; Hodinový impulz
        DJNZ    R0,V3
        MOV     tisk,#0H        ; Nuluj dílčí paměť
        MOV     A,dis3          ; Vytvoření čtvrtých dílčích 8 bitů
        JNB     ACC.1,S41
        XRL     tisk,#080H
S41:    JNB     ACC.0,S42
        XRL     tisk,#040H
S42:    JNB     ACC.5,S43
        XRL     tisk,#020H
S43:    JNB     ACC.6,S44
        XRL     tisk,#010H
S44:    MOV     A,dis4
        JNB     ACC.1,S45
        XRL     tisk,#08H
S45:    JNB     ACC.0,S46
        XRL     tisk,#04H
S46:    JNB     ACC.5,S47
        XRL     tisk,#02H
S47:    JNB     ACC.6,S48
        XRL     tisk,#01H
S48:    MOV     A,tisk          ; Vysílání čtvrtých 8 bitů
        MOV     R0,#08H
V4:     ADD     A,ACC           ; Vysílaný bit v příznaku přenosu C
        MOV     SDAT,C
        SETB    CLK
        CLR     CLK             ; Hodinový impulz
        DJNZ    R0,V4
        SETB    CLK
        CLR     CLK             ; Nevyužitý vývod
        RET

        ;TABULKA ZNAKŮ
        ;0,1,2,3,4,5,6,7,8,9, ,E,r,",C,t,I,M,b,-,L,i,S,P,d,A,c,H, ,F,U,u
```

```
TABZNAK:  DB 3fH,06H,0dbH,4fH,66H,6dH,7dH,07H,7fH,6fH,00H,79H
          DB 50H,063H,39H,78H,30H,37H,7cH,40H,38H,04H,6dH,73H
          DB 5eH,77H,58H,76H,00H,71H,3eH,1cH,00H,00H,00H,00H

          DSEG                ; Rezervace ve vnitřní datové paměti
          ORG     40H         ; Počátek rezervace
dis4:     DS      1H          ; Paměť 4 znaku
dis3:     DS      1H          ; Paměť 3 znaku
dis2:     DS      1H          ; Paměť 2 znaku
dis1:     DS      1H          ; Paměť 1 znaku
tisk:     DS      1H          ; Pomocná paměť
tecka:    DS      1H          ; Paměť desetinné tečky (1 až 4)
STACK:    DS      6H          ; Prostor pro zásobník
          END
```

Příklad 6

Navrhněte program pro realizaci hodin do 60 minut s BCD výstupem a se vstupem vnějšího nulování (aktivní úroveň je log. 0). Program doplňte podprogramem umožňujícím zobrazení na displeji z příkladu 5.

Chceme-li realizovat hodiny bez využití specializovaného hodinového obvodu, potom využíváme jeden z čítačů obvykle ve funkci časovače (čítače vnitřního oscilátoru). Takové řešení nám umožňuje, že funkce hodin bude nezávislá na hlavním programu, který bude občas přerušen. Protože přístup do přerušovacího podprogramu netrvá vždy stejnou dobu, je třeba využívat režimu časovače s obvodovým přednastavením (tj. mód 2 časovače 0 nebo 1 (8bitové přednastavení) nebo u časovače 2 procesorů 8052 (16bitové přednastavení)). Použijeme-li časovač 0, potom k jeho přetečení v módu 2 a krystalu 12 MHz dojde maximálně do 256 μs. Proto časovač budeme nastavovat tak, aby přerušovací program byl volán každých 250 μs. Čítáním jednotlivých volání vytvoříme informaci o 1 s (tj. 4000 volání) a z ní odvodíme zbývající údaje. Protože v módu 2 dochází k obvodovému přednastavení současně s přetečením časovače, je nutnou podmínkou správné činnosti takto koncipovaného řešení, aby každá žádost časovače o přerušení byla obsloužena. Z toho vyplývá, že v hlavním programu nesmí být znemožněno přerušení na dobu blížící se 250 μs. V programu budeme definovat paměťová místa pro ČÍTAČ (16 bitů) čítající jednotlivá přerušení, VTEŘINY (8 bitů) čítající v BCD kódu počet vteřin, MINUTY (8bitů) čítající v BCD kódu počet minut a HODBOD (bit) vytvářející 0,5 s intervaly pro blikání tečky na displeji.

```
HODBOD    BIT     00H         ; Bit indikující blikání tečky na displeji
          ORG     0000H       ; Začátek programu
          JMP     START       ; Obskoč vektory přerušení
          ORG     000BH       ; Adresa obsluhy časovače 0
          JMP     HODINY      ; Skok na obslužný program přerušeni
START:    MOV     TMOD,#022H  ; Nastav mód 2 časovačů 0 a 1
```

```
        MOV     TH0,#NOT(250)+1 ; Nastav přednastavení časovače 0
        MOV     TL0,TH0
        SETB    TR0             ; Spusť časovač 0
        MOV     IE,#092H        ; Povol přerušení časovače 0 i globálně
        MOV     SP,#STACK       ; Nastav ukazatel zásobníku
        CLR     A
        MOV     VTERINY,A       ; Nastav počáteční stav
        MOV     MINUTY,A
        MOV     CITAC,A
        MOV     CITAC+1,A
        ;.....
        ;Hlavní program
        ;.....
        CALL    ZOBRAZ          ; Zobrazte údaj na displeji
        ;.....
        ; Podprogramy
        ; NICI: ————
HODINY: PUSH    ACC             ; Ulož střadač před ztrátou
        PUSH    PSW             ; Ulož příznaky před jejich ztrátou
        MOV     A,CITAC+01H     ; Je obsah CITAC > 0,5s
        CJNE    A,#LOW(2000),NENITE
        MOV     A,CITAC
        CJNE    A,#HIGH(2000),NENITE
        CPL     HODBOD          ; Invertuj blikající tečku s intervalem 0,5s
NENITE: INC     CITAC+01H       ; Zvětši CITAC o jedničku
        MOV     A,CITAC+01H
        JNZ     NOHIGH
        INC     CITAC
NOHIGH: XRL     A,#LOW(4000)    ; Je CITAC = 4000
        JNZ     VEN             ; Není shoda
        MOV     A,CITAC
        XRL     A,#HIGH(4000)
NESHODA: JNZ    VEN             ; Není shoda
        CPL     HODBOD          ; Inverze blikající tečky
        MOV     CITAC,A         ; Dosažena 1s - nuluj čítač CITAC
        MOV     CITAC+01H,A
        MOV     A,VTERINY       ; VTERINY+1
        ADD     A,#01H          ; Dekadická korekce jen po sčítaní
        DA      A
        MOV     VTERINY,A
        XRL     A,#60H          ; Dosaženo minuty
        JNZ     VEN
        MOV     VTERINY,A       ; Ano, nuluj VTERINY
        MOV     A,MINUTY        ; MINUTY+1
        ADD     A,#01H
        DA      A
        MOV     MINUTY,A
        CJNE    A,#60H,VEN      ; Dosaženo 60 minut
        CLR     A               ; Ano, nuluj MINUTY
```

```
        MOV     MINUTY,A
VEN:    P0P     PSW             ; Obnov původní příznaky
        POP     ACC             ; Obnov obsah akumulátoru
        RETI                    ; Návrat z obsluhy přerušení
        ; NIČI: A,Příznaky
ZOBRAZ: MOV     A,VTERINY
        ANL     A,#0FH          ; Maskuj jednotky vteřin
        MOV     dis1,A          ; Ulož je do paměti prvního displeje
        MOV     A,VTERINY
        ANL     A,#0F0H         ; Maskuj desítky vteřin
        SWAP    A
        MOV     dis2,A          ; Ulož je do paměti druhého displeje
        MOV     A,MINUTY
        ANL     A,#0FH          ; Maskuj jednotky minut
        MOV     dis3,A          ; Ulož je do paměti třetího displeje
        MOV     A,MINUTY
        ANL     A,#0F0H         ; Maskuj desítky minut
        SWAP    A
        MOV     dis4,A          ; Ulož je do paměti čtvrtého displeje
        CLR     A
        JNB     HODBOD,NESVIT
        MOV     A,#3            ; Tečka na třetím displeji svítí
NESVIT: MOV     TECKA,A
        CALL    DISPLEJ
        RET

        ;PODPROGRAM DISPLEJ z příkladu 5

        DSEG
        ORG 30h
CITAC:  DS      2               ; počitadlo 250us
VTERINY: DS     1               ; Počitadlo vteřin
MINUTY: DS      1               ; Počitadlo minut
                                ; Proměnné podprogramu DISPLEJ
STACK:  DS      6H
        END
```

Příklad 7

Navrhněte podprogram pro vysílání řetězce znaků sériovým kanálem zakončeným znakem 00H.

Při řešení se musíme rozhodnout zda vysílání bude obsluhováno po celou dobu vysílání programově nebo přes přerušovací systém. V prvém případě vyčkává vysílací program ve smyčce dokud není znak sériovým kanálem vyslán. Potom předá další znak do vysílacího registru a přechází opět do čekací smyčky. V druhém případě je předán do vysílacího registru a řízení je vráceno do hlavního programu. Po odvysílání znaku bude nastaven bit TI (vysílací buffer prázdný),

který umožní vyvolání přerušení od sériového kanálu. Protože přerušení od sériového kanálu může být vyvoláno od příjmu i od vysílání, je třeba na začátku obslužného programu rozhodnout o jakou žádost se jedná. V následujícím textu jsou uvedeny oba případy.

```
        ;Vysílání sériovým kanálem bez využití přerušovacího systému
        ORG     0000h
        MOV     SCON,#52h       ; Inicializace, Režim 1 - 8bitový UART
        MOV     TMOD,#20h       ; Časovač 1 - mód 2
        MOV     TH1,#0F4h       ; Přenosová rychlost 2400 baudů pro
                                ; krystal 11,059 MHz
        SETB    TR1             ; Spusť časovač 1
        ; ......
        ; Hlavní program
        ; ......
        MOV     DPTR,#TEXT
        CALL    VYSPTX          ; Podprogram bude opuštěn až po odvysílání
                                ; celé zprávy
        ; ......
        ; Podprogramy a obslužné programy

        ;NICI:  A, DPTR, Příznaky
VYSPTXT: CLR    A               ; Nuluj 8bitový ukazatel
        MOVC    A,@A+DPTR       ; Vysílaný znak do střadače
        JZ      VY1             ; Je-li nulový pak konči
        MOV     SBUF,A          ; Zápis znaku do vysílacího registru
        JNB     TI,$            ; Čekej na jeho odvysílání
        CLR     TI              ; Nuluj návěští prázdného vys. registru
        INC     DPTR            ; Zvyš 16bitový ukazatel
        JMP     VYSPTXT         ; Opakuj
VY1:    RET

TEXT:   DB      'Vysílaný text ',0DH,00H
        END
        ;Vysílání sériovým kanálem s využitím přerušovacího systému
VPRENOS BIT     00H             ; Bit indikující vysílání sériovým kanálem
        ORG     0000h
        LJMP    START
        ORG     0023H
        LJMP    SERIAL          ; Skok na obslužný podprogram

START:  MOV     SCON,#52h       ; Inicializace, Režim 1 - 8bitový UART
        MOV     TMOD,#20h       ; Časovač 1 - mód 2
        MOV     TH1,#0F4h       ; Přenosová rychlost 2400 baudů pro
                                ; krystal 11,059 MHz
        SETB    TR1             ; Spusť časovač 1
        MOV     IE,#90H         ; Povol přerušení sériového kanálu a
                                ; globální masku
        CLR     VPRENOS         ; Sériový přenos neprobíhá
```

```
        MOV     SP,#STACK       ; Nastav zásobník
        ; .....
        ; Hlavní program
        ; .....
        MOV     DPTR,#TEXT      ; Nastav ukazatel zprávy
        CLR     A
        JB      VPRENOS,$       ; Počkej na odvysílání předcházející zprávy
        CLR     TI              ; Nuluj indikátor prázdného buffru
        MOVC    A,@A+DPTR       ; Přečti první znak
        INC     DPTR            ; Inkrementuj ukazatel
        MOV     SPOINT,DPH      ; Ulož ukazatel do paměti
        MOV     SPOINT+1,DPL    ; DPTR může být volně využíván hl.program.
        MOV     SBUF,A          ; Vyšli první znak zprávy
        SETB    VPRENOS         ; Sériový přenos zahájen
        ; .....
        ; Podprogramy a obslužné programy
        ;NICI: ——
SERIAL: JB      RI, PRIJEM      ; Je-li přerušení od příjmu, skoč na obsluhu
        PUSH    ACC             ; Ulož střadač před zničením
        PUSH    PSW             ; Ulož příznaky před zničením
        PUSH    DPH             ; Ulož DPTR před zničením
        PUSH    DPL
        MOV     DPL,SPOINT+1    ; Nastav ukazatel řetězce
        MOV     DPH,SPOINT
        CLR     A
        MOVC    A,@A+DPTR       ; Vyber další znak
        JNZ     VY2
        CLR     VPRENOS         ; Je-li znak 0, ukonči vysílání
        JMP     VY1
VY2:    MOV     SBUF,A          ; Vyšli další znak
        INC     DPTR            ; Inkrementuj ukazatel
        MOV     SPOINT,DPH      ; Ulož ukazatel do paměti
        MOV     SPOINT+1,DPL
VY1:    CLR     TI              ; Smaž indikátor prázdného vysílacího reg.
        POP     DPL             ; Obnov původní DPTP
        POP     DPH
        POP     PSW             ; Obnov příznaky
        POP     ACC             ; Obnov střadač
        RETI                    ; Vrať se k původní práci v hlavním programu
PRIJEM: CLR     RI              ; Obsluha příjmu se nevyužívá
        RETI

TEXT:   DB 'Vysílaný text ',0DH,00H

        DSEG                    ; Rezervace datové paměti
        ORG     30h
SPOINT: DS      2H              ; Paměť ukazatele vysílaného řetězce
STACK:  DS      12H             ; Hloubka zásobníku pro návratové adresy
        END                     ; a instrukce PUSH
```

K uvedenému příkladu je třeba dodat, že každý obslužný program musí uložit ty registry procesoru, které využívá. Jinak dojde k jejich změně a neočekávaným chybám v hlavním programu. Pokud obslužný program využívá velkého počtu registrů, je výhodnější využívat možnosti výměny banky registrů R0 až R7. V hlavním programu využíváme banku 0, v přerušeních s nižší úrovní priority banku 1 a v přerušeních s vyšší úrovní priority banku 2.

Příklad 8

Navrhněte program pro měření doby periody logického signálu s periodou v rozsahu (100 ms ÷ 50 ms) s přesností alespoň 5 μs.

Příklad je koncipován tak, aby bylo možné si ukázat, jak časovače a specializované periferie usnadňují tuto úlohu. Analýzou problému zjistíme, že k měření doby periody postačí 16bitový čítač. Problém měření periody můžeme řešit zcela programově, s využitím vnitřního časovače a přerušovacího systému nebo využitím specializované periferie - záchytného systému. Při čistě programovém řešení budeme programově čekat např. na vzestupnou hranu měřeného signálu. Po jejím dosažení bude program testovat logickou úroveň vstupního signálu (log. 1) a zároveň za provedený počet instrukcí definované délky (INC DPTR = 2 cykly, NOP = 1 cyklus a JB bit, adresa = 2 cykly, dohromady 5 μs pro 12 MHz) započte do registru DPTR jedničku. Analogicky program testuje i dobu log. 0 a s novou náběžnou hranou signálu měření ukončí. Pro správnou činnost je třeba, aby vstupní signál setrval v každé úrovni alespoň dva strojové cykly, tj. 2 μs. Hlavní nevýhoda tohoto řešení spočívá ve 100% využití výkonu procesoru v době měření periody signálu.

```
VSTUP   EQU     P1.2            ; Vývod, na který je přiveden měřený signál

        ORG     0000h
        MOV     DPTR,#0         ; Nuluj počitadlo 5 μs
NOVYME: JB      VSTUP,$         ; Čekej na konec log.1
        JNB     VSTUP,$         ; Čekej na konec log.0
LOG1:   INC     DPTR            ; Za každých 5 μs přičti jedničku
        NOP                     ; Korekce zpoždění smyčky
        JB      VSTUP,LOG1      ; Čítej dokud je VSTUP v log.1
LOG0:   INC     DPTR            ; Za každých 5 μs přičti jedničku
        NOP                     ; Korekce zpoždění smyčky
        JNB     VSTUP,LOG0      ; Čítej dokud je VSTUP v log.0
        ; .....
        CALL    ZOBRAZ          ; Doba periody = DPTR*5 μs
        ; .....
        JMP     NOVYME
        END
```

Druhé řešení, které si ukážeme bude využívat přerušovací systém a vnitřní časovače. To nám umožní provádět měření doby periody nezávisle na hlavním programu, protože programová obsluha (přerušení) bude pouze zajišťovat spuštění, zastavení a uložení hodnot. Po dobu vlastního měření pak může procesor vykonávat hlavní program. Předpokládejme, že využijeme časovač 0 v módu 1 (16 bitů) čítající vnitřní oscilátor o kmitočtu 12 MHz /12. Měřený signál přivedeme na vstup vnějšího přerušení, např. INT0, a aktivitu přerušovacího systému nastavíme na sestupnou hranu tohoto signálu. Před prvním vyvoláním přerušení vynulujeme časovač. S první sestupnou hranou měřeného signálu na vstupu INT0 bude vyvoláno přerušení, které spustí činnost časovače. Druhé přerušení čítání časovače zastaví, zajistí jeho uložení a vynulování. Třetí přerušení zahájí další měření doby periody. Díky tomu bude program, stejně jako v předcházejícím případě, provádět měření každé druhé periody signálu.

```
DRUHY    BIT     00H

         ORG     0000h
         LJMP    START
         ORG     0BH          ; Adresa obsluhy přerušení INT0
         JMP     MPERIOD
         ORG     30H
START:   MOV     TMOD,#21h    ; Časovač 0 - mód 1, časovač 1 - mód 2
         MOV     TCON,#41h    ; Spusť časovač 1, INT0 - sestupná hrana
         MOV     IE,#81h      ; Povol přerušení INT0 a globální masku
         MOV     SP,#STACK    ; Nastav ukazatel zásobníku
         CLR     DRUHY        ; Nastav indikátor přerušení na první cyklus
         ; ......
         ; Hlavní program
         ; ......
         ; Obslužné programy
         ;NICI: ——
MPERIOD: JB      DRUHY,STOP   ; Jedná se o druhou sestupnou hranu
         SETB    TR0          ; První hrana, spusť časovač
         SETB    DRUHY        ; Nastav indikátor druhé sestupné hrany
         RETI                 ; Návrat z přerušení
STOP:    CLR     TR0          ; Druhá hrana, zastav časovač
         CLR     DRUHY        ; Smaž indikátor druhé hrany
         MOV     PERIOD,TH0   ; Ulož naměřenou periodu
         MOV     PERIOD+1,TL0
         MOV     TH0,#0h      ; Nuluj časovač
         MOV     TH0,#0h      ;
         RETI                 ; Návrat z přerušení
         DSEG                 ; Rezervace datové paměti
         ORG 30h
PERIOD:  DS      2H           ; Paměť periody signálu
STACK:   DS      1H
         END
```

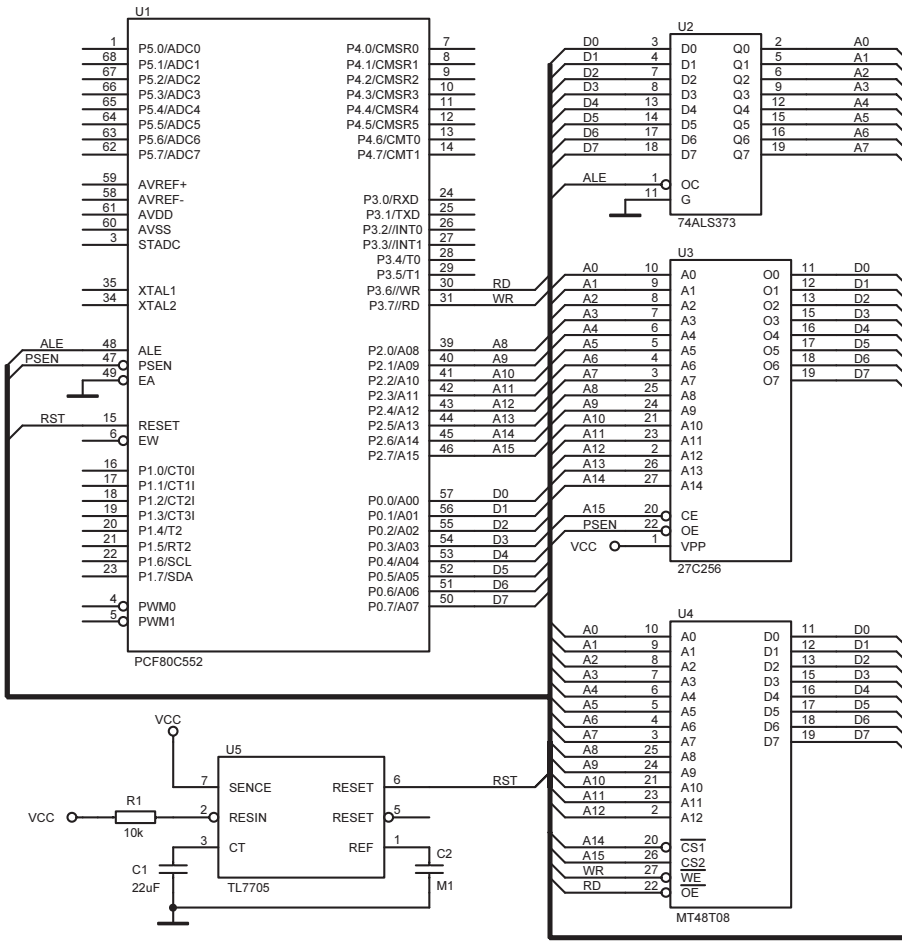
Ve třetím řešení si ukážeme využití záchytného systému časovače 2 procesoru 8052 ve spojení s přerušovacím systémem. Předpokládejme, že využijeme časovač 2 v záchytném režimu čítající vnitřní oscilátor o kmitočtu 12 MHz /12, a měřený signál přivedeme na vstup T2EX. Na začátku činnosti spustíme časovač 2 ve funkci čítače vnitřního oscilátoru, povolíme záchytný režim na sestupnou hranu signálu T2EX a povolíme přerušení od sestupné hrany na vstupu T2EX (nikoliv od přetečení časovače T2). Obslužný program přerušení potom převezme nově zachycenou hodnotu čítače T2, odečte ji od hodnoty předcházející s uvážením možnosti případného přetečení časovače T2 přes modul 65536. Vypočtenou hodnotu uloží do zobrazovaných registrů. Pro správnou činnost je nutné, aby do příštího zachycení (přerušení) byla výsledná hodnota vypočtena. Toto řešení umožňuje provádět měření každé periody signálu (měření první periody bude chybné).

```
        ORG     0000h
        LJMP    START
        ORG     0BH             ; Adresa obsluhy přerušení od časovače 2
        JMP     MPERIOD
        ORG     30H
START:  MOV     T2CON,#0Dh      ; Časovač 2 - 16bitový záchytný režim,
                                ; časovač 2 spuštěn, Vnější zachycení povol.
        MOV     IE,#0A0h        ; Povol přerušení od časovače 2
        MOV     SP,#STACK       ; Nastav ukazatel zásobníku
        ; ......
        ; Hlavní program
        ; ......
        ; Obslužné programy
        ;NICI: ——
MPERIOD: PUSH   ACC
        PUSH    PSW
        MOV     A,RCAP2L        ; Vezmi nižší část nové hodnoty časovače
        CLR     C               ; Nuluji příznak přenosu
        SUBB    A,OLDL          ; Perioda = Aktuální stav - předcházející stav
        MOV     PERIOD+1, A     ; Ulož nižší část rozdílu
        MOV     A,RCAP2H        ; Vezmi vyšší část nové hodnoty časovače
        SUBB    A,OLDH          ; odečti vyšší část staré hodnoty
        MOV     PERIOD, A       ; Ulož vyšší část rozdílu, Rozdíl = Perioda
        MOV     OLDL, RCAP2L    ; Ulož aktuální stav jako starý
        MOV     OLDH, RCAP2H    ;
        POP     PSW
        POP     ACC
        RETI                    ; Návrat z přerušení
        DSEG                    ; Rezervace datové paměti
        ORG 30h
PERIOD: DS      2H              ; Paměť periody signálu
OLDL    DS      1H              ; Paměť nižší části časovače 2
OLDH    DS      1H              ; Paměť vyšší části časovače 2
STACK:  DS      1H
        END
```

Příklad 9

Připojte k mikroprocesoru 80C552 vnější paměť programu o kapacitě 32 kB, vnější zálohovanou paměť o kapacitě 8 kB a hodiny reálného času běžící nezávisle na napájecím napětí mikroprocesorového systému.

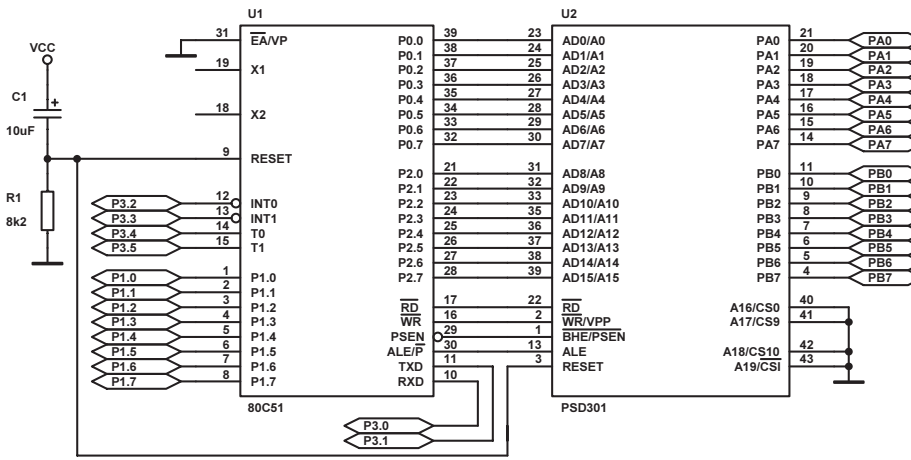
K řešení problému můžeme přistoupit klasickým způsobem, popsaným již několikráte v Amatérském rádiu, využívajícím specializovaný obvod hodin reálného času jako je RTC72421/423, DS1307, MK41T56 nebo V3021, obvodu pro kontrolu napájecího napětí a řízení záložní baterie jako je MAX690 až MAX695, datové paměti



Obr. 40 Připojení vnější programové a datové paměti

RAM typu 6164 a paměti programu obvykle EPROM typu 27C256. Druhým a jednodušším řešením, často nevyužívaným pro vyšší finanční náročnost, je využití specializovaného obvodu TIMEPEEK. Jedná se o paměť RAM s velmi malým příkonem, která má u sebe integrován obvod pro kontrolu napájecího napětí. Při poklesu napájecího napětí dochází k deaktivaci paměti RAM (nelze do ní přistoupit) a její přepnutí na záložní lithiovou baterii integrovanou nad vlastní pamětí. U některých typů jsou v horních 8 bytech paměti integrovány hodiny reálného času jako je např. M48T08, M48T18, DS1643, atd. Využití takového obvodu redukuje úlohu na připojení vnější paměti programu a dat k mikroprocesorovému systému.

Jak vyplývá z časového diagramu *obr. 27*, musíme nejprve k procesoru připojit obvykle úrovňově řízený registr 74HCT373, 74HCT573, který bude na výstupech držet spodní část adresy vysílané procesorem v koincidenci s výstupem ALE. Tyto výstupy společně s bránou P2 vytváří 16bitovou adresovou sběrnici. Brána P0 realizuje 8bitovou datovou sběrnici multiplexovanou s vysílanou spodní částí adresy a signály PSEN, RD a WR představují řídicí signály pro přístup do vnějšího programového a datového prostoru. Nyní musíme rozhodnout o tom, do kterých adresových prostorů požadované paměti připojíme. Protože se jedná o procesor bez vnitřní paměti programu, bude muset programová paměť ležet v adresovém prostoru od adresy 0000H. U připojení vnější datové paměti musíme rozhodnout, zda bude k systému připojena ještě nějaká datová paměť nebo periferie adresovaná do vnějšího adresového prostoru. Pokud by k systému byly připojovány další paměti nebo periferie, potom musíme zbývající adresové vodiče (A13, A14, A15) dekódovat a vytvořit aktivační signály CS pro jednotlivé paměti. Je-li k systému připojena jenom jedna paměť, potom je dekódování

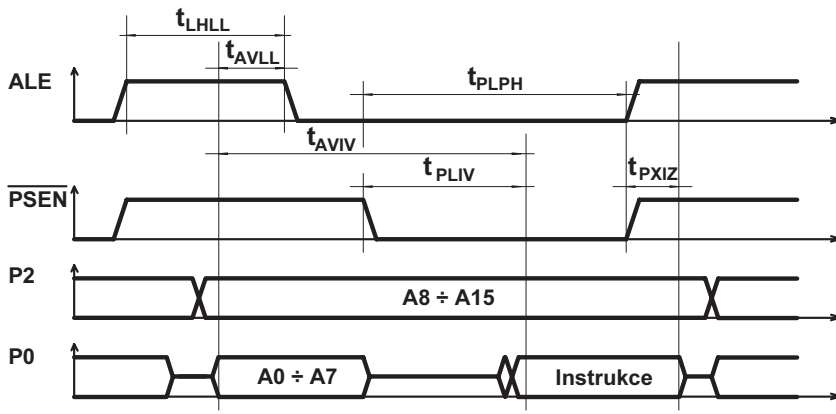


Obr. 41 Systém s obvodem PSD301

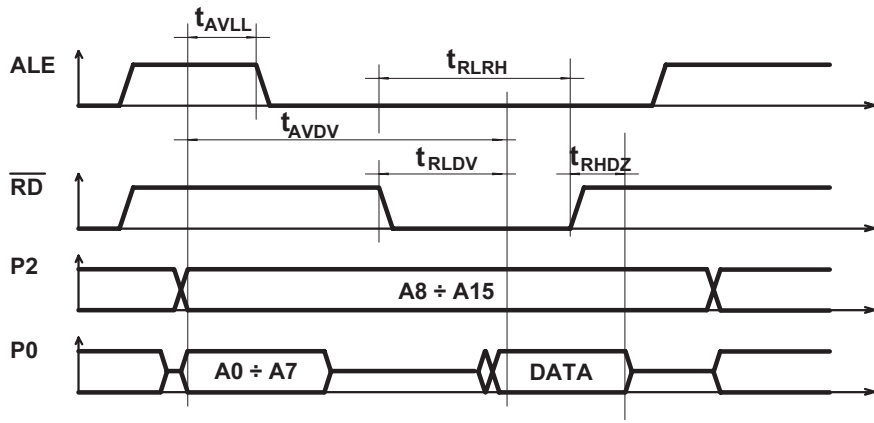
nevyužitých adresovacích vodičů zbytečné. V důsledku toho se však paměť RAM bude v celém adresovém prostoru opakovat 8 x (tj. 0000H÷1FFFH, 2000H÷3FFFH, ... , E000H÷FFFFH). Na *obr. 40* je zobrazeno jedno z možných zapojení systému, u kterého jsou jako aktivační signály použity adresovací vodiče A14 a A15. Díky tomu bude programová paměť umístěna v adresovém prostoru 0000H÷7FFFH a zálohovaná paměť s hodinami reálného času se bude zrcadlit v prostorech 8000H÷9FFFH a A000H÷BFFFH (vodič A13 není dekódován).

Hlavní nevýhodou při připojení vnější datové nebo programové paměti je ztráta dvou vstupně/výstupních bran P0 a P2. Ztráta 16 vývodů vede v mnoha aplikacích k nutnosti použití rozšiřujících bran pomocí klasických nebo programovatelných obvodů, případně i k použití procesoru s větším počtem bran (80C535, 80C552, 80C537, atd.). Je-li v aplikaci třeba šetřit místem, potom je možné místo klasického řešení využít specializovaný obvod z řady PSD3xx. Jedná se o programovatelný obvod, který obsahuje paměť EPROM s kapacitou 32 kB až 128 kB, paměť RAM s kapacitou 2 kB, registr pro ukládání multiplexované adresy a dvě vstupně/výstupní brány, které ztratíme připojením vnějších pamětí. Spojením procesoru 80C51 a obvodu PSD301 tak získáme systém s vnější programovou i datovou pamětí (nezálohovanou, bez hodin) se třemi vstupně/výstupními branami *obr. 41*.

Posledním problém, který musíme vyřešit, je stanovení nezbytných časových parametrů paměti EPROM a RAM, které musí splňovat pro daný procesor s připojeným krystalem. Na *obr. 42* a *obr. 43* jsou zobrazeny časové parametry procesoru (Philips 80C552). Nejprve musíme stanovit, zda jsou splněny časové parametry pro zápis spodní části adresy do registru 74HCT373, které jsou určeny časy signálu ALE $t_{AVLL} = 1/f_{osc} - 25$ [ns] a $t_{AVLL} = 2/f_{osc} - 40$ [ns]. Doba předstihu t_{setup} je



Obr. 42 Časování při čtení z programové paměti



Obr. 43 Časování při přístupu do vnější datové paměti

výrobcem stanovena na minimálně 6 ns a doba zapisovacího impulzu na minimálně 20 ns. Z porovnání vyplývají tyto vztahy

$$6 \geq 1/f_{osc} - 25 \qquad 20 \geq 2/f_{osc} - 40 \quad [ns]$$

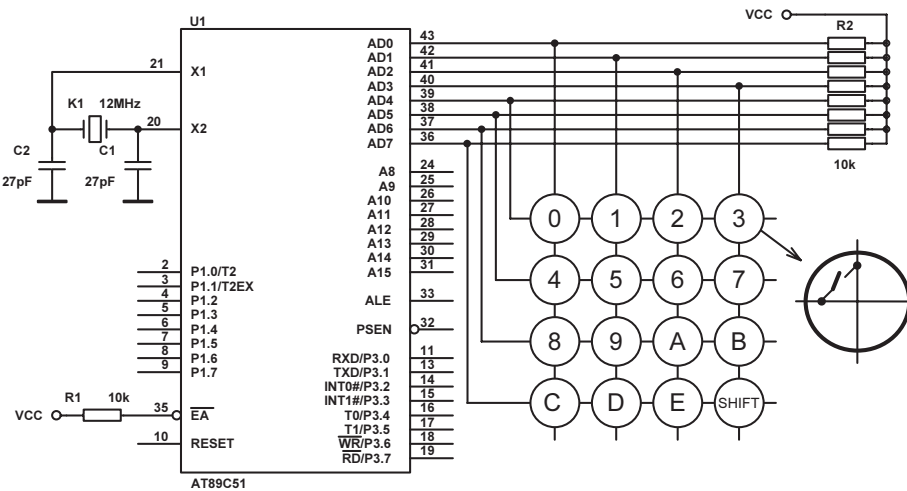
Použitý obvod bude vyhovovat až do kmitočtu 30 MHz. Pro programovou paměť EPROM musí být splněny doby vybavení od adresových vodičů a signálu CS a od aktivačního signálu OE. V našem případě je aktivační signál tvořen adresou A15 a proto doba vybavení je dána časem $t_{AVIV} = 5/f_{osc} - 80$ [ns]. Zároveň však musí být splněna i doba vybavení od aktivačního signálu OE ($t_{PLIV} = 3/f_{osc} - 60$ [ns]) třístavového budiče. Protože čas vybavení třístavového budiče bývá třikrát až čtyřikrát menší než od signálu CS, je pro určení časových parametrů paměti EPROM podstatný čas t_{AVIV}. Například paměť s dobou přístupu (vybavení) 120 ns můžeme připojit k procesoru s hodinovým kmitočtem $f_{osc} = 5 / (120 + 60) = 25$ MHz. Bude-li však použito doporučené zapojení paměti EPROM, u kterého jsou vstupy CS a OE spojeny se signálem PSEN, potom se doba vybavení bude počítat pouze z času t_{PLIV}. Vyšší nároky na rychlost paměti EPROM budou vyváženy méně častou aktivací paměti a tím i nižší spotřebou. Analogická situace je i doby potřebné k vybavení informace z datové paměti RAM. Doba vybavení od adresy je dána časem $t_{LLDV} = 9/f_{osc} - 165$ [ns] a doba vybavení od aktivačního signálu OE je dána $t_{RLDV} = 5/f_{osc} - 90$ [ns]. Například pro paměť s dobou přístupu 200 ns můžeme použít k procesoru s hodinovým synchronizačním signálem $f_{osc} = 9 / (200 + 165) = 24{,}6$ MHz.

Příklad 10

Navrhněte podprogram pro připojení maticové klávesnice 4x4 tlačítka k bráně P0 procesoru AT89C51. Výstupem z podprogramu bude informace o platné kombinaci kláves v příznaku přenosu C a odpovídajícím kódu ve střadači.

Připojení tlačítek nebo klávesnice je velmi častá úloha při realizaci hlavně měřících přístrojů. Jednotlivá tlačítka, na jedné straně uzemněná, se obvykle připojují na vstupní brány, které jsou pravidelně čteny. Nechceme-li pravidelně číst vstupní brány, potom můžeme tlačítka připojit ještě na vstupy logického členu AND, jehož výstup připojíme na vstup vnějšího přerušení. Je-li počet tlačítek velký, potom se obvykle zapojují do matice mající určitý počet řádků a sloupců. Zapojení tlačítek a potřebné logiky (členů AND) do řádků a sloupců, které budou připojeny pouze na vstupní brány procesoru, umožňuje indikovat zmáčknutí jednoho nebo dvou tlačítek najednou. Případné zmáčknutí tří tlačítek může, ale také nemusí být identifikovatelné. Z tohoto důvodu se častěji používá zapojení dle obr.44. Jednotlivá tlačítka jsou zapojena mezi řádky a sloupce matice. Přivedeme-li například pouze na jeden řádek úroveň log.0, potom přečtením všech sloupců snadno zjistíme, která tlačítka v daném řádku jsou zmáčknuta. Postupným výběrem všech řádků snadno zjistíme všechna zmáčknutá tlačítka na klávesnici.

U navrhovaného podprogramu je matice rozdělena do čtyř řádků a čtyřech sloupců. Řádky matice tlačítek připojíme na výstupy brány P0.4 až P0.7, sloupce matice na vstupy brány P0.0 až P0.3. Podprogram musí vyjma popsaného obec-



Obr. 44 Připojení klávesnice k bráně P0 procesoru 89C51

ného algoritmu správně obsloužit zmáčknuté tlačítko a odstranit případné zákmity. Používáme-li brána P0 jako vstupní nebo výstupní, musíme ji doplnit kolektorovými odpory vůči napájení (tzv. pull-up odpory).

```
                    ;PODPROGRAM pro obsluhu klávesnice 4x4 tlačítka

ODSTUP  EQU     40H
HORNI   BIT     20H

KLAVES: MOV     P0,#0FF         ;sloupce=Log.1=vstupy,řádky=Log.1
        XRL     P0,#00010000B   ;první řádek=Log.0
CEK1:   MOV     A,P0            ;čti sloupce
        CALL    ZPOZD           ;čekej a opakovaně čti sloupce
        JNZ     CEK1            ;jsou-li hodnoty shodné pokračuj
        ANL     A,#0FH          ;maskuj sloupce
        MOV     R1,A
        XRL     P0,#00110000B   ;druhý řádek=Log.0
CEK2:   MOV     A,P0            ;čti sloupce
        CALL    ZPOZD           ;čekej a opakovaně čti sloupce
        JNZ     CEK2            ;jsou-li hodnoty shodné pokračuj
        ANL     A,#0FH          ;maskuj sloupce
        SWAP    A
        ADD     A,R1
        XRL     P0,#01100000B   ;třetí řádek=Log.0
CEK3:   MOV     A,P0            ;čti sloupce
        CALL    ZPOZD           ;čekej a opakovaně čti sloupce
        JNZ     CEK3            ;jsou-li hodnoty shodné pokračuj
        ANL     A,#0FH          ;maskuj sloupce
        MOV     R2,A
        XRL     P0,#11000000B   ;čtvrtý řádek=Log.0
CEK4:   MOV     A,P0            ;čti sloupce
        CALL    ZPOZD           ;čekej a opakovaně čti sloupce
        JNZ     CEK4            ;jsou-li hodnoty shodné pokračuj
        ANL     A,#0FH          ;maskuj sloupce
        SWAP    A
        ADD     A,R2
        MOV     P0,#0FFH        ;všechny řádky do Log.1

        ;Část dekódující, která tlačítka jsou zmáčknuta. V 16 bitech
        ;registru R2,R1 je informace o zmáčknutém tlačítku (log.0).

DEKOD:  MOV     A,R2
        CLR     HORNI           ;zmáčknuté SHIFT
        JNB     ACC.7, SHIFT    ;je-li 15 bit Log.0 je zmáčknuto SHIFT
        SETB    HORNI           ;nezmáčknuté SHIFT
```

```
        MOV     R0,#15          ;adresa tabulky pro znaky bez SHIFT
        JMP     TEST
SHIFT:  MOV     R0,#31          ;adresa tabulky pro znaky s SHIFT
        ADD     A,ACC
        MOV     R3,#0           ;počítadlo zmáčknutých kláves
        MOV     R2,#7           ;počitadlo posunu doleva
TE1:    ADD     A,ACC           ;v příznaku přenosu je nejprve bit 14
        DEC     R0              ;snižuj adresu
        JC      TE2             ;není-li klávesa zmáčknuta pokračuj
        INC     R3              ;zvyš počitadlo zmáčknutých kláves
        MOV     B,R0            ;ulož adresu kódu klávesy
TE2:    DJNZ    R2,TE1
        MOV     R2,#8           ;počitadlo posunů doleva
        MOV     A,R1            ;do střadače bity 7 až 0
TE3:    ADD     A,ACC           ;v příznaku přenosu je nejprve bit 7 DEC
R0                              ;snižuj adresu
        JC      TE4             ;není-li klávesa zmáčknuta pokračuj
        INC     R3              ;zvyš počitadlo zmáčknutých kláves
        MOV     B,R0            ;ulož adresu kódu klávesy
TE4:    DJNZ    R2,TE3
```

;Vyhodnocení testu zmáčknutých kláves

```
        MOV     A,R3
        JNZ     TE5
        JB      HORNI,TE6       ;skoč není-li zmáčknuté SHIFT
        CLR     A
        MOV     DPTR,#KODSIF
        MOVC    A,@A+DPTR
        SETB    C               ;platný kód ve střadači
        RET
TE6:    CLR     C               ;není-li zmáčknuta ani jedna klávesa
        MOV     A,#0H
        RET
TE5:    DEC     A
        JNZ     TE6             ;vyjma SHIFT je zmáčknuto více kláves
        MOV     DPTR,#TABKOD
        MOV     A,B             ;dej adresu kódu znaku
        MOVC    A,@A+DPTR
        SETB    C               ;platný kód ve střadači
        RET

ZPOZD:  MOV     R0,#ODSTUP
        DJNZ    R0,$            ;ekvivalentní   NAV: DJNZ R0,NAV
        MOV     B,A
        MOV     A,P0
```

```
            CLR     C
            SUBB    A,B
            RET

TABKOD:     DB  00H,01H,02H,03H     ;kódy kláves v prvé řadě
            DB  04H,05H,06H,07H     ;kódy kláves v druhé řadě
            DB  08H,09H,10H,11H
            DB  12H,13H,14H,15H
            DB  20H,21H,22H,23H     ;kódy kláves v prvé řadě s SHIFT
            DB  24H,25H,26H,27H     ;kódy kláves v druhé řadě  s SHIFT
            DB  28H,29H,30H,31H
            DB  32H,33H,34H
KODSIF:     DB  40H                 ;kód samotné klávesy SHIFT
```

3.1. Vývoj programů pro procesory 8051

Při realizaci programu pro daný typ procesoru je potřeba nejen znát dobře architekturu procesoru a možnosti jeho instrukčního souboru, ale také vlastnosti a možnosti programového vybavení, na kterém bude program vytvářen. Kromě těchto znalostí je vhodné dodržovat určité základní postupy při tvorbě programového vybavení, které můžeme rozdělit do následujících kroků:

- Nejprve musíme logicky správně popsat programovanou úlohu a navrhnout vhodné algoritmy pro její řešení. Pro snazší realizaci je vhodné úlohu rozdělit do menších funkčních celků, které budou později realizovány podprogramy, funkcemi nebo moduly.
- Vybrat vhodný programovací jazyk k napsání tzv. ***zdrojového programu***, případně zdrojových souborů, které reprezentují jednotlivé moduly. Pro realizaci efektivních (obvykle rychlejších a paměťově méně náročnějších) programů je vhodný ***jazyk symbolických adres JSA*** (označovaný též JSI - jazyk symbolických instrukcí). Tento jazyk je velmi blízký vlastnímu strojovému kódu procesoru a dnes se používá jako nejnižší programovací úroveň. Pokud se nevyužívají připravené podprogramy a knihovní moduly je vývoj programu a jeho testování časově zdlouhavé. Časově efektivnější je programování ve vyšším programovacím jazyce (jazyk C, Basic, atd.), které ale vede na paměťově rozsáhlejší programy. Jsou-li napsány zkušeným programátorem, potom jsou obvykle o něco pomalejší, než programy napsané v JSA. Zdrojový program v JSA nebo jazyce C je možné vytvořit v libovolném tzv. ASCII editoru, který do textu nepřidává žádné řídící a informační znaky.
- Připravené zdrojové texty se překládají pomocí překladačů. Je-li zdrojový text v jazyce C, potom je nejprve přeložen pomocí ***kompileru*** do jazyka symbolických adres (JSA). Zdrojové programy vytvořené kompilerem nebo

přímo v JSA se překládají pomocí ***assembleru*** do tzv. relativního modulu (souboru s příponou OBJ). Tyto moduly lze spojit pomocí ***linkeru*** (spojovacího a přemísťovacího programu) do výsledného absolutního programu, adresovaného pro danou konkrétní konfiguraci mikroprocesorového systému. Tento program je možné nahrát do obvodového emulátoru nebo programového simulátoru a ověřit jeho činnost.

- Je-li výsledný program odladěn a vykonává požadovanou činnost, potom je nahrán do vnitřní paměti samotného procesoru nebo do pamětí EPROM, EEPROM, FLASH nebo NVRAM. Pro programátory pamětí a procesorů je třeba výsledný absolutní modul programu transformovat do tzv. INTEL HEX nebo podobného formátu. K tomu se obvykle používá program, který bývá označen OH.EXE.

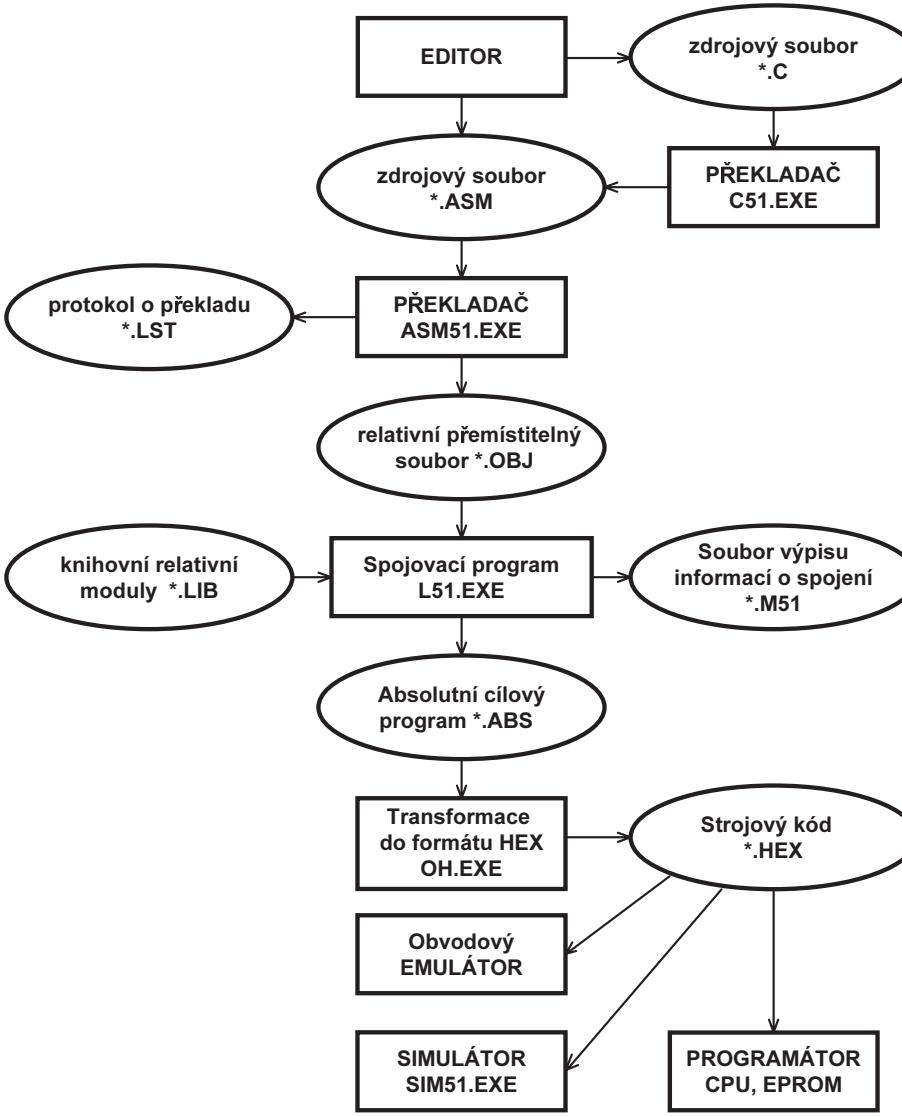
3.1.1. Modulární tvorba programů

Modulární programování přináší výhody v jednodušším a hlavně přehlednějším vytváření rozsáhlých programů. Umožňuje snazší ladění a úpravy vytvářených podprogramů, včetně možnosti jejich násobného použití a snadného začleňování do programových knihoven. Důležitou podmínkou je správné definování vstupů a výstupů podprogramů a předávání parametrů mezi moduly, kde modul je jeden nebo několik spojených segmentů. Programátor pak může snadno jednotlivé moduly odladit a ověřené moduly spojit do výsledného i velmi rozsáhlého programu nebo začlenit do knihoven. Pokud se ve výsledném programu vyskytnou chyby, opravy je možné opět provádět v jednotlivých modulech nebo podprogramech. Postup při tvorbě programu je zobrazen na obr.45 pro programování s programovými segmenty, moduly, knihovnami a programy. Přípony uvedené v obrázku jsou obvykle generovány implicitně.

Již z předcházející části je zřejmé, že v napsaných programech se objevují příkazy, které s vlastním jazykem symbolických adres nemají nic společného. Jedná se tzv. ***direktivy asembleru***, které se někdy označují jako pseudoinstrukce. Jsou to příkazy, které většinou řídí překladač a umožňují inicializovat a rezervovat paměťový prostor. Tyto příkazy lze obecně rozdělit do skupin definující symboly, rezervující paměťový prostor, popisující vlastnosti segmentu nebo modulu, ovládající čítač instrukcí a výběr segmentu nebo registrové banky. V následující části budou popsány některé základní direktivy, které se při tvorbě programu používají.

Definice symbolů

Mezi základní direktivy definující v programu symboly a jejich umístění v daném typu paměti mikroprocesorového systému patří: SEGMENT, EQU, SET, BIT, DATA, IDATA, XDATA a CODE. Nyní si jednotlivé direktivy popíšeme podrobněji.



Obr. 45 Tvorba programu v jazyce symbolických adres a jazyce C

Direktiva **SEGMENT** slouží k označení ucelené části programové nebo datové paměti a identifikaci jejího umístění. Direktiva má tento formát:

Jméno_relativního_segmentu SEGMENT *typ_segmentu*

kde ***typ segmentu*** určuje paměťový prostor v němž musí být segment umístěn. Typ segmentu je určen pro procesor 8051 jedním z těchto klíčových slov:

CODE	Programová paměť
XDATA	Vnější datová paměť
DATA	Vnitřní přímo adresovatelná datová paměť (adresy 0 - 127)
IDATA	Vnitřní nepřímo adresovatelná datová paměť (adresy 0 - 255)
BIT	Vnitřní paměť s přímo adresovatelnými bity

Příklad:

```
ORG     80H
STACK   SEGMENT IDATA
        RSEG                STACK
        DS                  10H     ;rezervace 16 bytů pro zásobník
        .....                       ;další inicializace
        .....                       ;začátek programu
        MOV SP,#STACK-1             ;inicializace ukazatele zásobníku
```

Direktiva **EQU** slouží k přiřazení konkrétní hodnoty adresy, konstanty, hodnoty výrazu nebo symbolu asembleru danému symbolu. Formát direktivy je následující:

Jméno_symbolu EQU *výraz,*

kde výraz představuje konkrétní nebo symbolickou hodnotu např. 345h, ADR1+1, 2*POCITAT nebo $.

Direktiva **SET** je identická s direktivou EQU s tím rozdílem, že s její pomocí lze jednomu *jménu symbolu* v rámci jednoho programu přiřazovat postupně různé hodnoty a výrazy. U direktivy EQU to možné není.

Direktivy **BIT, DATA, IDATA, XDATA** a **CODE** slouží k přiřazení konkrétní hodnoty nebo výrazu, který představuje adresu daného paměťovému prostoru. Formát direktiv je následující:

Jméno_symbolu	**BIT**	*výraz* (bitová adresa)
Jméno_symbolu	**DATA**	*výraz* (přímá bytová vnitřní adresa)
Jméno_symbolu	**IDATA**	*výraz* (nepřímá bytová vnitřní adresa)
Jméno_symbolu	**XDATA**	*výraz* (nepřímá bytová vnější adresa 16bitů)
Jméno_symbolu	**CODE**	*výraz* (bytová adresa prog.paměti 16bitů)

Inicializace nebo rezervace paměti

Mezi základní direktivy inicializující nebo rezervující prostor v paměti mikroprocesorového systému patří: DS, DB, DW a DBIT. Formát těchto direktiv je následující:

[*návěští*:]	**DS**	*výraz*
[*návěští*:]	**DB**	*výraz* nebo *seznam výrazů*
[*návěští*:]	**DW**	*výraz* nebo *seznam výrazů*
[*návěští*:]	**DBIT**	*výraz*

Direktiva **DS** slouží k rezervaci paměti (bytů) mimo bitový segment, direktiva **DB** slouží k přiřazení 8-bitového výrazu nebo výrazů na paměťová místa, direktiva **DW** slouží k přiřazení 16-bitového výrazu nebo výrazů na paměťová místa a direktiva **DBIT** rezervuje bity v bitové oblasti interní paměti procesoru. Návěští uvedená v hranatých závorkách jsou nepovinná.

Vlastnosti modulu a symbolů

Vytvořené moduly mohou být relativní (přemístitelné) nebo absolutní (jejich umístění v paměťovém prostoru je pevně dané). U jednotlivých modulů a symbolů mohou být definovány vlastnosti vůči svému okolí. Direktiva **PUBLIC** umožňuje zveřejnit (zpřístupnit) symbol deklarovaný v daném modulu pro ostatní moduly. Direktiva **EXTERN** informuje linker o tom, že příslušný modul nebo symbol v modulu bude deklarován v některém z ostatních modulů. Direktiva **NAME** slouží k identifikaci modulu.

Řízení stavu programového čítače assemblerem

Mezi základní direktivy řídící stav programového čítače asembleru patří direktiva ORG a END. Direktiva **END** určuje konec zdrojového programu a jakákoliv část programu v JSA následující za touto direktivou nebude do překladu zahrnuta. Direktiva ORG má tento formát:

[*návěští*:] **ORG** *výraz,*

kde výraz představuje hodnotu na kterou bude nastaven při překladu programový čítač PC. Není-li na začátku programu výraz ORG použit, potom překlad začíná v programovém segmentu od adresy 0000h. Návěští uzavřené v hranaté závorce je opět nepovinné.

Direktivy výběru segmentu nebo registrové banky

Mezi základní direktivy určující typ segmentu patří: **RSEG, CSEG, DSEG, XSEG, ISEG, BSEG**. První písmeno v označení segmentu určuje typ segmentu, s kterým se bude pracovat. Segmenty CSEG (programový), DSEG (datový), XSEG

(vnější datový), ISEG (vnitřní datový) a BSEG (bitový) se používají při absolutním adresování daného segmentu. Segment RSEG se využívá pro relativní adresování tzn., že linker může při skládání programu příslušný segment umístit dle potřeby. K využívání dané registrové banky slouží direktiva **USING (0÷3)**. Příkaz nezajistí v JSA přepnutí banky, ale zajití při použití označení registrů AR0 až AR7, že operace bude provedena s registrem ze zvolené banky registrů.

Obecná pravidla pro zápis programu

Jméno symbolu musí začínat písmenem nebo speciálním znakem s výjimkou otazníku nebo mezery. Jméno může tvořit posloupnost až 255 znaků, ale pouze prvních 31 znaků je významných. Jako symbolická jména nemohou být použity implicitně vyhrazená jména, jako jsou mnemonické zkratky instrukcí, operátory asembleru, názvy direktiv a atributy segmentů. Při definici symbolů se používají tzv. ***vyhrazené symboly***, které mají předem určený význam nebo způsob zápisu. Například: $ = aktuální stav PC, @ nepřímá adresa, # přímá data a symboly (mnemonické zkratky) registrů a jejich bitů (registry např. PSW, A, apod., bity např. C, F0, P1.5, atd.).

Návěští je symbol, za kterým obvykle bezprostředně následuje dvojtečka. Jméno návěští se nesmí shodovat jménem nějakého symbolu. Návěští může být před prázdnou řádkou, před instrukcí procesoru a před direktivami pro rezervaci paměti DS a DBIT. Symbolická jména (bez dvojtečky) mohou být pouze před direktivami přiřazení (EQU, SET, CODE, DATA, IDATA, XDATA, BIT, SEGMENT). Hodnota návěští může být buď absolutní nebo relativní v závislosti na aktuálním segmentu. S návěštím je možné pracovat jako s jakýmkoliv symbolem, jako s hodnotou výrazu, který je návěští přiřazen, nebo adresou. Návěští nesmí být definováno vícenásobně.

Přímá data (***konstanty***) jsou vždy uvedeny znakem # a mohou být obvykle zadávány v binárním (např. 01110101B), dekadickém (např. 156 nebo 156D) nebo hexadecimálním (např. 0AEFH) tvaru. Konstanta v jakémkoliv vyjádření musí začínat číslicí.

4. Mikroprocesor 8xC251SB

Obvod 8xC251SB je prvním mikroprocesorem z řady MCS251 (Intel) 8bitových procesorů rozšiřujících vlastnosti a hlavně výkon široce používaných procesorů z řady 8051. Uvedený procesor je zajímavý hlavně tím, že v jednom z módů, do kterého lze naprogramovat (režim kódové kompatibility), je obvod 8xC251SB vývodově i programově slučitelný se stávajícími procesory řady 8051. S minimálními zásahy do existujícího zařízení i programu lze zvýšit výkon až 6násobně. Vlastní procesory z řady MCS251 budou mít tyto společné vlastnosti:

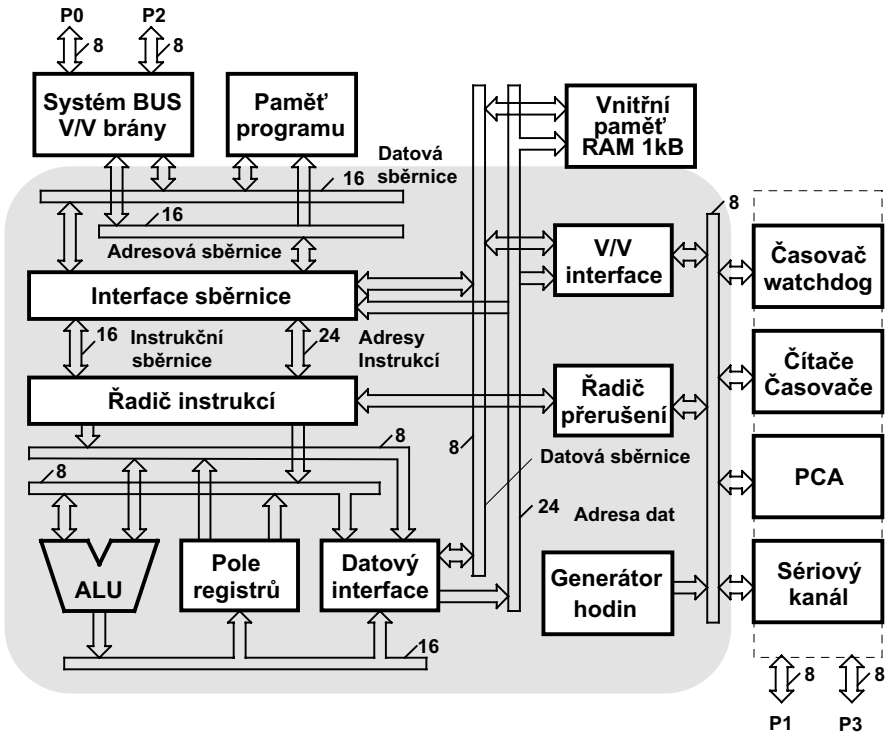
- 24bitovou adresu umožňující lineárně adresovat až 16 Mbitů paměti
- registrové jádro CPU s registry přístupnými po bytech, slovech a slovech dvojnásobné délky
- stránkování paměti pro zvýšení rychlosti přístupu k vnějším pamětem
- dvojnásobné překrývání instrukcí (pipeline)
- rozšíření zásobníku až na 64 kB
- strojový cyklus tvořený dvěma hodinovými cykly
- kódovou kompatibilitu s 8051
- výrazně rozšířený instrukční soubor

Na *obr. 46* je zobrazen blokový diagram 8xC251SB, kde je šedou plochou označeno jádro procesoru, které bude společné pro celou rodinu 251. Jednotlivé typy se budou od sebe lišit počtem i typy periferií na čipu. První zástupce je vybaven čtyřmi 8bitovými branami, třemi čítači/časovači, čítačem podporovaným programovatelným polem (PCA), čítačem watchdog a sériovým kanálem. Každý V/V vývod je obecně použitelný nebo může být využit jako vývod alternativní funkce, která je na něj vyvedena. Z *obr. 46* je zřejmé, že čtení instrukcí z vnitřní programové paměti probíhá po dvou bytech najednou, z vnější paměti pouze po bytech. Pro zrychlení čtení z vnější paměti je vhodné konfigurovat procesor do stránkového módu. Čtení instrukce ze stejné stránky potom trvá pouze jeden strojový cyklus (dva hodinové cykly) a čtení z různých stránek dva strojové cykly (čtyři hodinové cykly). Řadič přerušení přijímá sedm maskovatelných žádostí až od jedenácti zdrojů přerušení.

4.1. Organizace paměti

Mikroprocesor 8xC251 má tři adresové prostory, které jsou tvořeny paměťovým prostorem s kapacitou až 16 MB (00:0000H ÷ FF:FFFFH), 512 byty specializovaných registrů (S:000H ÷ S:1FFH) a 64 byty souboru registrů. Na *obr. 47* je zobrazen celý adresový prostor procesoru 8xC251, který se skládá z 256 64kB oblastí, které nadále budeme formálně označovat 00: až FF:. Hodnoty uložené

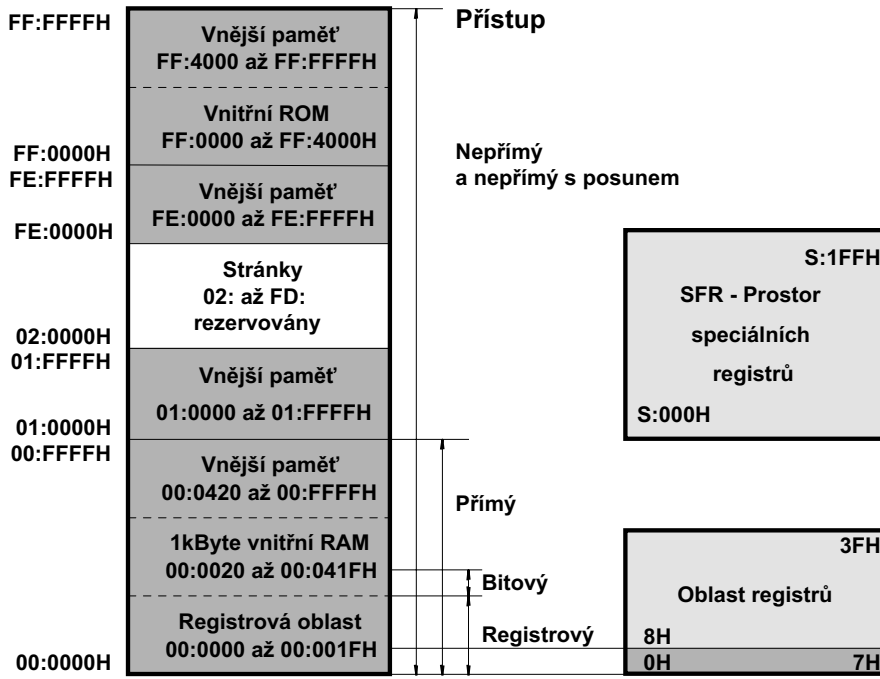
v paměti mohou být, vyjma oblasti 00:, adresovány pouze nepřímo nebo nepřímo s posunem. V oblasti 00:0020H až 00:041FH (bude záviset na typu obvodu) je umístěna na čipu paměť RAM, která slouží k ukládání dat. Tuto paměť, jako jedinou část ze čtyřech bloků (00:, 01:, FE: a FF:, nelze využívat jako paměť programovou a tudíž v ní nelze spustit program. Data jsou adresovatelná přímo, nepřímo i nepřímo s posunem. Oblast této paměti od adresy 00:0020H až 00:007FH je bitově adresovatelná. V oblasti FF:0000H až FF:3FFFH je umístěna vnitřní programová paměť, která ve shodě s procesory 8051 je ovládána vývodem EA (EA=0 - pouze vnější paměť). Oblast speciálních registrů je ovládána přes adresy S:000H až S:1FFH, které jsou v asembleru C251 povinné pro odlišení od adres 00:0000H až 00:01FFH vnitřní datové paměti procesoru.



Obr. 46 Vnitřní bloková struktura procesoru 8xC251SB

V registrové oblasti jsou obsazeny pozice (adresy) 00H÷1FH a 38H÷3FH. Oblast 20H÷37H je zatím rezervována. K registrům, které jsou umístěny v jádře CPU, lze v závislosti na jejich umístění přistupovat po bytech, slovech nebo slovech s dvojnásobnou délkou. Po bytech lze přistupovat k registrům R0 až R15, které

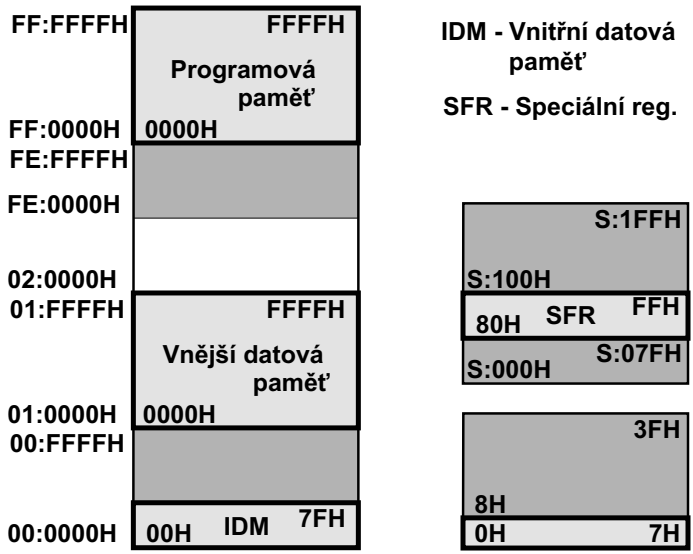
leží na pozicích 00H÷0FH. Na pozice 00H÷1FH lze přistupovat po slovech (16 bitech) přes registry označené WR0, WR2, WR4, až WR30 obsahující vždy dva byty (tj. WR0 pozici 0H a 1H, WR2 pozici 2H a 3H, až WR30 pozici 1EH a 1FH). Nakonec lze přistupovat pomocí deseti 32bitových registrů označených DR0 (0,1,2,3), DR4 (4,5,6,7), .., DR28 (1CH,1DH,1EH,1FH) a DR56 (38H÷3BH) a DR60 (3CH÷3FH). Vyjma přístupu do bytového souboru přes názvy jednotlivých registrů, je umožněn přístup na pozice 00H÷1FH přes nejnižší adresy vnitřní paměti RAM (tj. 00:0000H÷00:001FH), nebo přes registry R0 až R7 jedné ze čtyřech aktuálních bank registrů.



Obr. 47 Struktura paměťového prostoru řady 8xC251 (vybar.části 8XC251SB)

Z popsaných registrů mají některé registry výsadní postavení. Registr R10 představuje registr B, registr R11 představuje akumulátor (ACC), registr DR56 je rozšířený ukazatel DPX (DPXL, DPH, DPL) a DR60 je rozšířený ukazatel zásobníku (SPH, SP). Střadač i registr B jsou přístupné i přes adresu z oblasti speciálních registrů (ACC = S:0E0H a B = S:0F0H). Stejně tak tomu je i s dílčími registry rozšířených ukazatelů dat a zásobníku. SP představuje 8bitový ukazatel zásob-

níku v módu 8051, v architektuře C251 se prodlužuje na 16 bitů pro umístění zásobníku po celé paměťové oblasti 00:. Speciální registry SFR jsou umístěny v jádře procesoru nebo v jeho periferiích. Umístění jednotlivých registrů je v dodatku tohoto textu.



Obr. 48 Rozmístění adresových prostorů 8051 v architektuře MCS251

Na *obr. 48* je zobrazeno umístění paměťových oblastí procesoru 8051 v architektuře procesoru 8xC521. Je-li procesor modifikován do režimu slučitelného s 8051, potom přístupy k jednotlivým oblastem zůstávají stejné jako na procesoru 8051.

Architektura MCS251 podporuje sedm typů adresovacích módů:

- *registrové adresování,* kde operandem je registr např. R0÷R15, WR0, WR2, ...nebo DR0, DR4, ... (v módu 8051 pouze R0÷R7).
- *bezprostřední adresování,* kde operand je přímo uveden v instrukci např. #data16.
- *přímé adresování.* V instrukci je přímo uvedena adresa operandu např. 00:0020H.
- *nepřímé adresování.* V instrukci je obsažen registr, který obsahuje adresu operandu např. @WRj (16bitová adresa v WRj), @DRj (24bitová adresa v DRj).

- *nepřímé adresování s posunem (bázové).* Adresa operandu je dána součtem 16bitového posunu a nepřímé adresy např. @WRj+8000H. Součet 16bitových adres je realizován v modulu 2^{16} a bude vždy ležet uvnitř prvních 64 kB paměti. Při použití slova dvojnásobné délky k adresování a posunu např. @Dk+01:0020H, potom výsledná adresa bude ležet v celém adresovém prostoru (nejvyšší byte musí být 0).
- *relativní adresování* je využíváno při skokových instrukcích.
- *bitové adresování.* U tohoto typu adresování v architektuře MCS251 dochází ke změně v možnostech symbolického zápisu, kdy je zrušena možnost zápisu přímé adresy. Na bit je možno se obrátit symbolickým jménem, označením bitu registru nebo bitu adresy např. povolení masky přerušení od sériového kanálu *SETB ES, SETB IE0.4 nebo SETB S:A8.4H.*

4.2. Přerušovací systém

Mikroprocesor 8xC251 má v zásadě stejný přerušovací systém jako procesor 8051. Zvětšil se pouze počet zdrojů, které mohou přerušení vyvolat, o jeden výstup z čítačem řízeného programovatelného pole (EC) a jeden výstup od časovače 2 nebo vnějšího vstupu časovače 2 (ET2). Osmým zdrojem přerušení, které oproti ostatním nelze zakázat, je instrukce TRAP (programové přerušení). Druhou odlišností je zvětšení počtu úrovní priority jednotlivých přerušení ze dvou na čtyři úrovně. Tím se uživateli výrazně zvětšují možnosti individuální konfigurace priorit jednotlivých přerušení. Výrobcem definované priority v dané úrovni zůstávají stejné jako na 8051, nová přerušení mají nejnižší priority (časovač 2 po sériovém kanále, PCA až po časovači 2). Adresy počátků obslužných podprogramů jednotlivých přerušení pokračují v duchu 8051 tj. sériový kanál FF:0023H, časovač 2 FF:002BH a PCA FF:0033H.

Nové periferie vytvářejí svoje žádosti o přerušení, stejně jako sériový kanál, jako logický součet dvou nebo více návěští. V případě časovače 2 je žádost o přerušení vytvořena jako logický součet bitu TF2 (přetečení časovače 2) a bitu EXF2 (vnější návěští 2). V případě programovatelného pole PCA je žádost tvořena logickým součtem pěti návěští událostí CCFx a návěští CF (přetečení časovače). Protože všem návěštím je přiřazena jedna adresa obslužného programu, je nutné v tomto programu zjistit zdroj přerušení a teprve potom jej vynulovat. Z tohoto důvodu musí být žádosti časovače2, PCA i sériového kanálu nulovány programově. Protože přerušovací systém procesoru 251 má čtyři úrovně priority, je procesor vybaven dvěma registry úrovně priority IPH (vyšší) a IPL (nižší). Odpovídající bity z obou registrů představují binární číslo od 0 (nejnižší) do 3 (nejvyšší) určující úroveň priority daného přerušení.

Bit	Zdroj přerušení	Bit	Zdroj přerušení
0	Vnější přerušení INT0	1	Přetečení časovače 0
2	Vnější přerušení INT1	3	Přetečení časovače 1
4	Sériový kanál	5	Časovač 2
7	PCA	8	nevyužit

Tabulka 7 Obsazení bitů priority v registrech IPH0 a IPL0

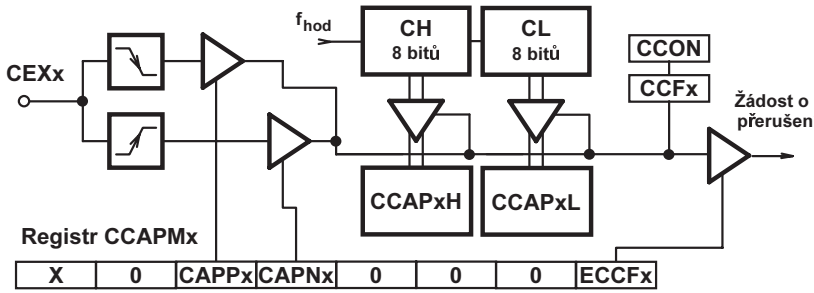
Doba potřebná k přijetí přerušení závisí na typu přerušení, na úrovni a rozpracování instrukce, na bloku paměti, ve které probíhá program. Doba odezvy (tj. doba přechodu od žádosti do obslužného programu se skládá z minimální doby a proměnné doby. Minimální doba je dána dobou potřebnou ke zjištění žádosti o přerušení 4 hodinové cykly (doba vzorkování žádostí), dobou k vyhodnocení žádosti 1 cyklus a dobou potřebnou k vyprázdnění pipeline, uložení návratové adresy a nezbytných hodnot 11 cyklů. Proměnná část se pohybuje od jednoho až do 8 cyklů a proto celková doba potřebná k vyvolání přerušení se bude pohybovat od 17 do 24 hodinových cyklů.

4.3. Periferie 8xC251SB

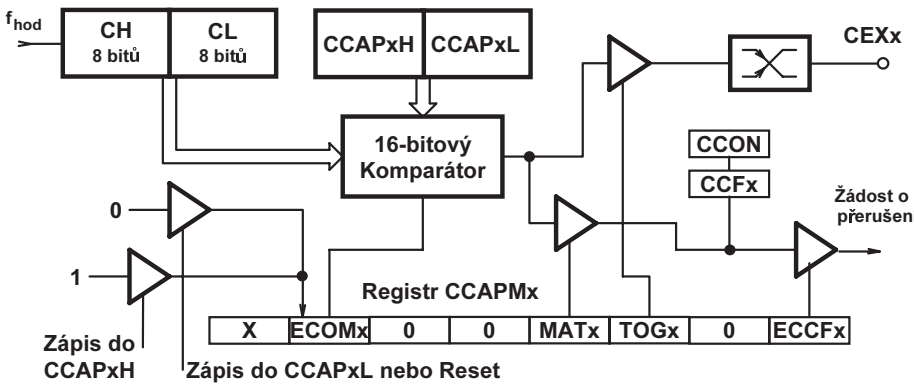
Mikroprocesor 8XC251SB je vybaven čtyřmi vstupně/výstupními branami označenými P0 až P3. Všechny brány jsou obousměrné (P1, P3 pseudoobousměrné) a na rozdíl od procesorů 8051 mají všechny vodiče alternativní funkci. Jediná výrazná odlišnost je ve využití bran P0 a P2 k připojení a adresování vnějších pamětí. V tzv. nestránkovém režimu je situace shodná s procesorem 8051, ve stránkovém režimu slouží brána P0 k přenosu dolní části adresy (bez registru) a brána P2 slouží k multiplexovanému přenosu dat a horní části adresy, která je zapisována do úrovňově řízeného registru. Tím je dosaženo situace, kdy při práci s pamětí na stejné stránce (uvnitř 256bytů) nemusí být vysílána hodní část adresy a instrukce je zrychlena.

Procesor je, jak již bylo řečeno vybaven třemi čítači, časovačem watchdog a PCA. V případě čítačů je situace stejná jako u procesorů řady 8052. Pro zajištění časování, operací s čítačem a generování PŠM (pulzně šířkové modulace) je mikroprocesor 8xC251SB vybaven programovatelným polem podporovaným čítačem. PCA se skládá z 16bitového čítače/časovače a pěti 16bitových komparačních nebo záchytných modulů. Čítač/časovač je společný pro všechny moduly a je dosažitelný v oblasti SFR pod dvojicí registrů CH a CL. Pět párů registrů SFR označených CCAPxH a CCAPxL obsahuje 16bitovou komparační nebo zachycenou hodnotu. Pro konfiguraci činnosti PCA slouží registry CMOD (řídicí

registr čítače a časovače) a CCON (řídicí registr operací). Každý modul má přiřazen registr CCAPMx (x= =1,2,3,4,5), který určuje mód činnosti modulu. Čítač/ /časovač tvořený registry CH a CL má vstupní hodinový signál f_{hod} volitelný ze tří vnitřních a jednoho vnějšího zdroje (f_{osc} /12, f_{osc} /4, přetečení časovače 0 a vnější signál z vývodu P1.2/ECI), kde f_{osc} je kmitočet oscilátoru procesoru. Jeden ze čtyř zdrojů je vybrán pomocí bitů CPS1 a CPS0 registru CMOD. Každý modul může být v závislosti na svém řídicím registru CCAPMx nastaven do záchytného a komparačního režimu.



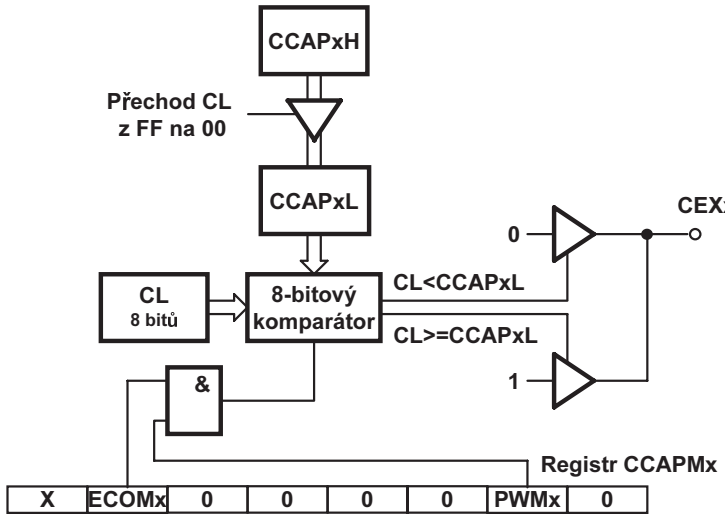
Obr. 49 Záchytný systém PCA



Obr. 50 Komparační systém PCA v módu programový čítač a rychlý výstup

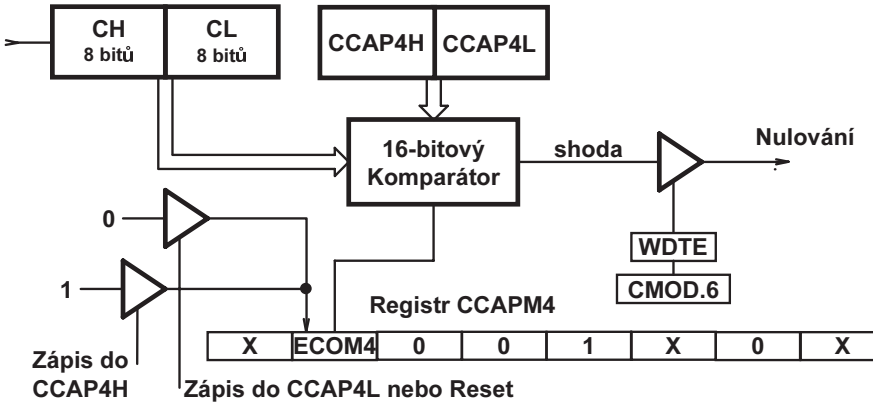
V záchytném režimu *obr. 49*, který se většinou využívá k měření délky pulzů, se při vzestupné hraně (CAPPx=1, CAPNx= =0), sestupné hraně (CAPPx=0, CAPNx= =1) nebo obou hranách (CAPPx =1, CAPNx=1) zapíše obsah čítače CH,CL do registrů CCAPxH a CCAPxL.

V komparačním režimu *obr. 50* může být každý modul využíván ve funkci: 16bitový programový časovač (výstupem je žádost o přerušení), rychlý výstupní mód (výstup je z vývodu CEXx), nebo pulzně šířkový modulátor *obr. 51.* V případě modulu 4 je povolena ještě činnost ve funkci časovače watchdog *obr. 52*. Dojde-li ke shodě čítače PCA a nastavené hodnoty v registrech modulu 4, potom je generován nulovací impulz o minimální délce nutné k v nulování procesoru (tj. 64 hodinových cyklů).



Obr. 51 PCA - 8bitová pulzně šířková modulace

Sériový kanál a jeho módy zůstávají u procesoru C521 stejné jako na procesoru 8051. Jeho vlastnosti byly vylepšeny pouze drobnými úpravami jako je zavedení bitu (FE) indikujícího chybu rámce přenášeného znaku (přesně chybu stop bitu) v mó ech 1, 2 a 3. Bit byl sloučen s bitem SM0 v registru SCON (bit.7) s tím, že je-li nastaven bit SMOD0 v registru PCON, potom bit SCON.7 představuje bit FE. Je-li bit SMOD0 vynulován, potom má nejvyšší bit SCON původní funkci (SM0). Chyba rámce (bit FE) musí být nulován programově. Dalším vylepšením je zavedení registrů SADDR a SADEN, které v multiprocesorové komunikaci umožňují automatické vyhledávání adresy přijímaného zařízení. Dojde-li při příjmu k zachycení adresy, teprve potom je nastaven příznak RI. V opačném případě není procesor přerušován příkazy (adresami) určenými pro ostatní zařízení (procesory). Každé zařízení má svoji adresu, která je uložena v registru SADDR. Registr SADEN obsahuje byte, kterým se určují (log. 0) bity adresy, jejichž hodnota není při příjmu adresy zajímavá. Tím je umožněno, aby nadřízený procesor mohl přenášet hodnoty do dvou nebo více podřízených procesorů. Vhod-



Obr. 52 PCA - Časovač watchdog

nou volbou adres a maskovacích bytů, tak lze vytvořit adresy pro komunikaci s jedním, dvěma nebo se všemi podřízenými procesory.

Jako každý moderní procesor umožňuje obvod 8xC251 přechod do módů se sníženou spotřebou. V módu IDLE jsou vývody ALE a PSEN v log. 1 a procesor může být do normální činnosti přiveden přijetím povoleného přerušení od vnitřní periferie, vnějšího zdroje. Nebo vynulováním. V případě power down módu jsou vývody ALE i PSEN v log. 0 (**!!! Pozor !!!**) a procesor může být do normální činnosti přiveden přijetím povoleného vnějšího přerušení nebo vynulováním. Pro testování zařízení lze procesor uvést do emulačního stavu (OECE), kdy je elektricky izolován od testovaného systému. Tím je umožněno připojení emulátoru nebo testovacího procesoru.

4.5. Konfigurace procesoru

Procesor MCS251 obsahuje čtyři konfigurační byty CONFIG0 až CONFIG3 umístěné v oblasti OTPROM a přístupné na adresách 80H až 83H při podmínkách P0=29H, P1=high(adresa), P2=data, P3=low(adresa), PROG=1, PSEN=0, RST=1a Vpp=5V. Byty CONFIG2 a CONFIG3 jsou rezervované a umístění konfiguračních bitů v CONFIG0 a CONFIG1 je dáno *tabulkou 8* a *9*.

CONFIG0							Bity
7	6	5	4	3	2	1	0
---	---	XSA	XALE	RD1	RD0	PAGE	SRC

Tabulka 8

Bity 7 a 6 - jsou rezervované, zapište do nich log. 1

WSA - Čekací stav A, vynulováním je nastaven jeden čekací stav (2 hodinové cykly) při přístupu do paměťových oblastí 00:, FE: a FF:. Nastavením (log. 1) nejsou při přístupu čekací stavy do daných oblastí

XALE - Nastavením bitu (log. 1) je ALE široké jeden hodinový cyklus. Vynulováním je prodlouží na tři hodinové cykly.

RD1,RD0 - Funkční nastavení signálů RD a PSEN.
RD1=0, RD0=0 Rezervováno
RD1=0, RD0=1 RD=A16, PSEN pro všechny adresy, adresový prostor 128kB
RD1=1, RD0=0 RD=pouze P3.7, PSEN pro všechny adresy, jeden vývod navíc.
RD1=1, RD0=1 RD pro Adr≤7F:FFFFH, PSEN pro Adr≥80:0000H, adresový prostor totožný s 8051

PAGE - Pro log. 0 ***stránkový mód*** (A15:8/D7:0=P2, A7:0=P0). Pro log. 1 ***nestránkový mód*** (A7:0/D7:0=P0, A15:8=P2).

SRC - Pro log. 1 ***zdrojový mód*** (MCS251). Pro log. 1 ***binární mód***. Kód kódově kompatibilní s procesorem 8051.

Pro zajištění plné slučitelnosti s 8051 programujte obvod v pouzdře PLCC44 na hodnotu ***CONFIG0=1101 1110B.***

CONFIG1							Bity
7	6	5	4	3	2	1	0
---	---	---	INTR	WSB	---	---	EMAP

Tabulka 9

Bity 7 a 5 - jsou rezervované, zapište do nich log. 1

INTR - Pro log. 1 ukládají přerušení 4 byty do zásobníku (3 byty ukazatele PC a registr PSW1). Pro log. 0 ukládají přerušení 2 byty do zásobníku (2 spodní byty ukazatele PC).

WSB - Čekací stav B, vynulováním je nastaven jeden čekací stav (2 hodinové cykly) při přístupu do paměťové oblasti 01:. Nastavením (log. 1) není při přístupu do oblasti generován čekací stav.

Bity 2 a 1 - jsou rezervované, zapište do nich log. 1

EMAP - Mapování paměti EPROM
Pro log. 0 je horních 8 kB programové paměti (FF:2000H až FF:3FFFH) mapováno do oblasti 00:E000H až 00:FFFFH.
Pro log. 1 je horních 8 kB programové paměti mapováno pouze v oblasti.

Pro zajištění plné slučitelnosti s 8051 programujte obvod v pouzdře PLCC44 na hodnotu ***CONFIG1=1110 0111B.***

Oproti procesoru 8051 jsou na vývodech 1, 12, 23 a 34 vyvedeny další napájecí a zemnicí vývody. Tyto vývody u procesoru 80C251SB nemusí být zapojeny. Procesor je však vyroben v rychlé CMOS technologii, u které jsou problémy s indukčností přívodních vývodů, která způsobuje velké napěťové špičky. Proto se u moderních obvodů umísťuje větší počet napájecích vývodů do středu všech stran čipu a doporučuje se je všechny zapojit.

4.6. Instrukční soubor

Instrukční soubor procesoru 251 byl rozšířen o operace se slovy a slovy dvojnásobné délky, které na procesoru 8051 vyjma instrukce INC DPTR nebyly implementovány. Další výraznou změnou je ztráta výsadního postavení střadače ACC jako registru, který se účastnil všech aritmetických a logických operací. Nyní lze provádět aritmetické a některé logické operace mezi 8bitovými a 16bitovými registry bez účasti střadače. Kromě instrukcí sčítání, odečítání, porovnání, logického součtu, logického součinu a operace EX-OR, jsou zavedeny nové instrukce násobení (8 x 8 = 16 bitů a 16 x 16 = 32 bitů) a dělení (8 x 8 = 8 + 8zb bitů a 16 x 16 = 16 +16zb bitů) mezi registry. Nové instrukce rotace umožňují logickou i aritmetickou rotaci 8bitového a 16 bitového registru. Pro přesun 8bitové hodnoty do 16bitového registru jsou zavedeny instrukce přesunu bez a nebo s važováním znaménkového rozšíření. Instrukční soubor obsahuje deset nových instrukcí podmíněného větvení programu. Byly zavedeny podmínky *rovno (JE), nerovno (JNE)* a dvojice podmínek pro čísla se znaménkem a bez znaménka jako jsou *větší (JG, JSG), menší (JL, JSL), větší nebo rovno (JGE, JSGE)* a *menší nebo rovno (JLE, JSLE).* Se zvětšením adresového prostoru byly rozšířeny instrukce volání, návratu, PUSH a POP pro práci s 24bitovou adresou. *V tabulce 9* je přehled všech instrukcí procesoru 80C251. Barevně jsou zvýrazněny instrukce procesoru 8051.

Aritmetické instrukce

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Binární		Zdrojový	
			Byty	Stavy	Byty	Stavy
ADD	**A, Rr**	(A)←(A) + (Rr)	1	1	2	2
	A, @Ri	(A)←(A) + ((Ri))	2	1♣	2	1♣
	A, adr8	(A)←(A) + (adr8)	1	2	2	3
	A, #data	(A)←(A) + data	2	1	2	1
ADD **SUB = -** **CMP = ?**	**Rnd, Rns**	(Rnd)←(Rnd) + (Rns)	3	2	2	1
	WRjd, WRjs	(WRjd)←(WRjd) + (WRjs)	3	3	2	2
	DRkd, DRks	(DRkd)←(DRkd) + (DRks)	3	5	2	4
	Rn, #data	(Rn)←(Rn) + data	4	3	3	2
	WRj,#data16	(WRj)←(WRj) + data16	5	4	4	3
	DRk, #0data16	(DRk)←(DRk) + data16 s 0 rozšíř.	5	6	4	5
	Rn, adr8	(Rn)←(Rn) + (adr8)	4	3♣	3	2♣
	WRj, adr8	(WRj)←(WRj) + (adr8)	4	4	3	3
	Rn, adr16	(Rn)←(Rn) + (adr16)	5	3	4	2
	WRj, adr16	(WRj)←(WRj) + (data16)	5	4	4	3
	Rn, @WRj	(Rn)←(Rn) + ((WRj))	4	3	3	2
	Rn, @DRk	(Rn)←(Rn) + ((DRk))	4	4	3	3
ADDC **SUBB**	**A, Rr**	(A)←(A) ± (Rr) ± ©	1	1	2	2
	A, adr8	(A)←(A) ± (adr8) ± ©	2	1♣	2	1♣
	A, @Ri	(A)←(A) ± ((Ri)) ± ©	1	2	2	3
	A, #data	(A)←(A) ± data ± ©	2	1	2	1
CMP	**DRk, #1data16**	porovnej DRk a 16-bit.hod se zn.	5	6	4	5
INC **DEC**	**A**	(A)←(A) ± 1	1	1	1	1
	Rr	(Rr)←(Rr) ± 1	1	1	2	2
	dir8	(adr8)←(adr8) ± 1	2	2♥	2	2♥
	@Ri	((Ri))←((Ri)) ± 1	1	3	2	4
	Rn, #short	(Ri)←(Ri) ± short	3	2	2	1
	WRj, #short	(WRj)←(WRj) ± short	3	2	2	1
	DRk, #short	(DRk)←(DRk) ± short	3	4	2	3
INC	**DPTR**	DPTR←DPTR + 1	1	1	1	1
DA	**A**	dekadická korekce po sčítání	1	1	1	1
MUL	**AB**	A=low(A*B), B=high(A*B)	1	5	1	5
	Rnd, Rns	nd=0,2, ... ,14, (Rnd)=low(sou.), (Rnd + 1)=high(sou.) nd=1,3, ... ,15, (Rnd-1)=low(sou.), (Rnd)=high(sou.)	3	6	2	5
	WRjd, WRjs	nd=0,4, ... ,28, (WRj)=low(sou.), (WRj + 2)=high(sou.) nd=2,6, ... ,30, (WRj-2)=low(sou.), (WRj)=high(sou.)	3	12	2	11

DIV	**AB**	A=(A/B), B=zbytek(A/B)	1	10	1	10
	Rnd, Rns	(Rnd)=(Rnd/Rns), (Rnd +1)=zbytek nd=0,2, ... ,14 (Rnd - 1)=(Rnd/Rns), (Rnd)=zbytek nd=1,3, ... ,15	3	11	2	10
	WRjd, WRjs	(WRjd)=(WRjd/WRjs), (WRjd+2)=zbyt. pro nd=0,4, ... ,28 (WRjd-2)=(WRjd/WRjs), (WRjd)=zbyt. pro nd=2,6, ... ,30	3	21	2	20

Logické operace

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Binární		Zdrojový	
			Byty	Stavy	Byty	Stavy
	A, Rr	(A)←(A) ∩ nebo ∪ nebo ⊕ (Rr)	1	1	2	2
	A, @Ri	(A)←(A) oper. ((Ri))	2	1♣	2	1♣
ANL = ∩	**A, adr8**	(A)←(A) oper. (adr8)	1	2	2	3
součin	**A, #data**	(A)←(A) oper. data	2	1	2	1
	adr8, A	(adr8)←(adr8) oper. (A)	2	2♥	2	2♥
ORL = ∪	**adr8, #data**	(adr8)←(adr8) oper. data	3	3♥	3	3♥
součet	**Rnd,Rns**	(Rnd)←(Rnd) oper. (Rns)	3	2	2	1
	WRjd, WRjs	(WRjd)←(WRjd) oper. (WRjs)	3	3	2	2
XRL = ⊕	**Rn, #data**	(Rn)←(Rn) oper. data	4	3	3	2
EX-OR	**WRj,#data16**	(WRj)←(WRj) oper. data	5	4	4	3
	Rn, adr8	(Rn)←(Rn) oper. (adr8)	4	3♣	3	2♣
	WRj, adr8	(WRj)←(WRj) oper. (adr8)	4	4	3	3
	Rn, adr16	(Rn)←(Rn) oper. (adr16)	5	3	4	2
	WRj, adr16	(WRj)←(WRj) oper. (adr16)	5	4	4	3
	Rn, @WRj	(Rn)←(Rn) oper. ((WRj))	4	3	3	2
	Rn, @DRk	(Rn)←(Rn) oper. ((DRk))	4	4	3	3
CLR	**A**	Nuluj střadač	1	1	1	1
CPL	**A**	Jednotkový doplněk střadače	1	1	1	1
RL	**A**	Rotace střadače doleva (8-bitová)	1	1	1	1
RLC	**A**	Rotace střadače doleva (9-bit.CY)	1	1	1	1
RR	**A**	Rotace střadače doprava (8-bitová)	1	1	1	1
RRC	**A**	Rotace střadače doprava (9-bit.CY)	1	1	1	1
SLL	**Rn**	Posun reg.(bytu) doleva Rn.0=0	3	2	2	1
	WRj	Posun reg.(slova) doleva WRj.0=0	3	2	2	1
SRA	**Rn**	Aritmet. posun reg. (bytu) doprava	3	2	2	1
	WRj	Aritmet. posun reg.(slova) doprava	3	2	2	1
SRL	**Rn**	Posun reg.(bytu) doprava Rn.7=0	3	2	2	1
	WRj	Pos. reg.(slova) doprava WRj.15=0	3	2	2	1
SWAP	**A**	Prohození půlslabik střadače	1	2	1	2

Přesunové instrukce

Instrukce	Operandy	*POPIS INSTRUKCE*	Binární		Zdrojový	
	Cílový, Zdrojový		Byty	Stavy	Byty	Stavy
MOV	**A, Rr**	(A)←(Rr)	1	1	2	2
	A, adr8	(A)←(adr8)	2	1♣	2	1♣
	A, @ Ri	(A)←((Ri))	1	2	2	3
	A, #data	(A)←data	1	2	2	3
	Rr, A	(Rr)←(A)	1	2	2	3
	Rr, #data	(Rr)←data	1	2	2	3
	adr8, A	(adr8)←(A)	1	2	2	3
	adr8, Rr	(adr8)←(Rr)	1	2	2	3
	adr8d, adr8s	(adr8d)←(adr8s)	1	2	2	3
	adr8, @Ri	(adr8)←((Ri))	1	2	2	3
	adr8, #data	(adr8)←data	1	2	2	3
	@Ri, adr8	((Ri))←(adr8)	1	2	2	3
	@Ri, #data	((Ri))←data	1	2	2	3
	DPTR, #data	DPTR←data	1	2	2	3
	Rnd,Rns	(Rnd)←(Rns)	3	3	2	2
	WRjd, WRjs	(WRjd)←(WRjs)	3	2	2	1
	DRkd, DRks	(DRkd)←(DRks)	3	3	2	2
	Rn, #data	(Rn)←data	4	3	3	2
	WRj,#data16	(WRj)←data16	5	3	4	2
	DRk, #0data16	(DRk)←data16 s rozšíř. nulami	5	5	4	4
	DRk, #1data16	(DRk)←data16 se znam.rozšíř.	5	5	4	4
	DRk, adr8	(DRk)←(adr8)	4	6	3	5
	DRk, adr16	(DRk)←(adr16)	5	6	4	5
	Rn, adr8	(Rn)←(adr8)	4	3♣	3	2♣
	WRj, adr8	(WRj)←(adr8)	4	4	3	3
	Rn, adr16	(Rn)←(adr16)	5	3	4	2
	WRj, adr16	(WRj)←(adr16)	5	4	4	3
	Rn, @WRj	(Rn)←((WRj))	4	2	3	2
	Rn, @DRk	(Rn)←((DRk))	4	4	3	3
	WRjd, @WRjs	(WRjd) ←((WRjs))	4	4	3	3
	WRj, @DRk	(WRj) ←((DRk))	4	5	3	4
	adr8, Rn	(adr8) ←(Rn)	4	4♣	3	3♣
	adr8, WRj	(adr8)←(WRj)	4	5	3	4
	adr16, Rn	(adr16) ←(Rn)	5	4	4	3
	adr16, WRj	(adr16)←(WRj)	5	5	4	4
	@WRj, Rn	((WRj)) ←(Rn)	4	4	3	3
	@DRk, Rn	((DRk)) ←(Rn)	4	5	3	4

MOV	**@WRjd, WRjs**	((WRjd)) ←(WRjs)	4	5	3	4
	@DRk, WRj	((DRk)) ←(WRj)	4	6	3	5
	adr8, DRk	(adr8) ←(DRk)	4	7	3	6
	adr16, DRk	(adr16)←(DRk)	5	7	4	6
	Rn, @WRj+dis16	(Rn) ←((WRj)+dis16) uvnitř 64kB	5	6	4	5
	WRj, @WRj+dis16	(WRj) ←((WRj)+dis16) v 64KB	5	7	4	6
	Rn, @DRk+dis24	(Rn) ←((WRj)+dis16) uvnitř 16MB	5	7	4	6
	WRj, @DRk+dis24	(WRj) ←((WRj)+dis16) v 16MB	5	8	4	7
	@WRj+dis16,Rn	((WRj)+dis16)←(Rn) uvnitř 64kB	5	6	4	5
	@WRj+dis16,WRj	((WRj)+dis16)←(WRj) v 64KB	5	7	4	6
	@DRk+dis24,Rn	((WRj)+dis24)←(Rn) uvnitř 16MB	5	7	4	6
	@DRk+dis24,WRj	((WRj)+dis24)←(WRj) v 16MB	5	8	4	7
MOVH	**DRk(hi), #data 16**	data16 do horních bytů DRk	5	3	4	2
MOVS	**WRj, Rn**	Rn do WRj se znam.rozšířením	3	2	2	1
MOVZ	**WRj, Rn**	Rn do WRj s rozšířením nulami	3	2	2	1
MOVC	**A, @A+DPTR**	(A)=((A)+DPTR)	1	6	1	6
	A, @A+PC	(A)=((A)+PC)	1	6	1	6
MOVX	**A, @Ri**	(A)=((Ri))	1	4	2	5
	A, @DPTR	(A)=(DPTR)	1	5	1	5
	@Ri, A	((Ri))=(A)	1	4	1	4
	@DPTR, A	(DPTR)=(A)	1	5	1	5
XCH	**A, Rr**	Prohození obsahu, (A)↔(Rr)	2	3	2	3
	A, adr8	Prohození obsahu, (A)↔(adr8)	2	3♥	2	3♥
	A, @Ri	Prohození obsahu, (A)↔((Ri))	1	4	1	4
XCHD	**A, @Ri**	Vyměn.půlslabik (low), (A)↔((Ri))	1	4	1	4
PUSH	**adr8**	Ulož obsah adr8 do zásobníku	2	2	2	2
	#data	Ulož data do zásobníku	4	4	3	3
	#data16	Ulož data16 do zásobníku	5	5	4	5
	Rn	Ulož byte Rn do zásobníku	3	4	2	3
	WRj	Ulož slovo WRj do zásobníku	3	6	2	5
	DRk	Ulož dvojslovo DRj do zásobníku	3	10	2	9
POP	**adr8**	Vyjmi byte ze zásobníku do adr8	2	3♣	2	3♣
	Rn	Vyjmi byte ze zásobníku do Rn	3	3	2	2
	WRj	Vyjmi slovo ze zásobníku do WRj	3	5	2	4
	DRk	Vyjmi dvojslovo ze zásob. do DRk	3	9	2	8

Vysvětlivky jsou na str. 94.

Instrukce větvení programu

Instrukce	Operandy	POPIS INSTRUKCE	Binární		Zdrojový	
	Cílový, Zdrojový		Byty	Stavy	Byty	Stavy
ACALL	**addr11**	Volání podprog. uvnitř 4kB	2	9	2	9
ECALL	**@ DRk**	Nepřímé volání prog. s 24-bit adr.	3	12	2	11
	addr24	Volání podprog. s 24-bit adresou	5	14	4	13
LCALL	**@ WRj**	Nepřímé volání prog. s 16-bit adr.	3	9	2	8
	addr16	Volání podprog. s 16-bit adresou	3	9	3	9
RET		Návrat z podprogramu	1	6	1	6
ERET		Rozšířený návrat z podprogramu	3	10	2	9
RETI		Návrat z přerušení	1	6	1	6
AJMP	**addr11**	Krátký skok v rámci 4 kB	2	3	2	3
EJMP	**addr24**	Skok na 24-bit adresu	5	6	4	5
	@ DRk	Nepřímý skok na 24-bit adr. v DRk	3	7	2	6
LJMP	**addr16**	Skok na 16-bit adresu	3	6	2	5
	@ WRj	Nepřímý skok na 24-bit adr. v WRj	3	4	3	4
SJMP	**rel**	Skok o relativní hodnotu	2	3	2	3
JMP	**@A + DPTR**	Nepřímý skok na DPTR+(A)	1	5	1	5
JC	**rel**	Skoč je-li (CY)=1	2	1/4	2	1/4
JNC	**rel**	Skoč je-li (CY)=0	2	1/4	2	1/4
JB	**bit 51, rel**	Skoč je-li (bit51)=1/0	3	2/5	3	2/5
JNB	**bit, rel**	Skoč je-li (bit)=1/0	5	4/7	4	3/6
JBC	**bit 51, rel**	Skoč je-li (bit51)=1, (bit51)←0	3	4/7	3	4/7
	bit, rel	Skoč je-li (bit)=1, (bit)←0	5	7/10	4	6/9
JZ	**rel**	Skoč je-li (A)=0	2	2/5	2	2/5
JNZ	**rel**	Skoč je-li (A)≠0	2	2/5	2	2/5
JE	**rel**	Skoč je-li (Z)=1	3	2/5	2	1/4
JNE	**rel**	Skoč je-li (Z)=0	3	2/5	2	1/4
JG	**rel**	Skoč je-li (CY)=0	3	2/5	2	1/4
JLE	**rel**	Skoč je-li (CY)=1 ∪ (Z)=1	3	2/5	2	1/4
JSL	**rel**	Skoč je-li (CY)=1 (hod. znam.)	3	2/5	2	1/4
JSLE	**rel**	Skoč je-li (CY)=1 ∪ (Z)=1, znam.	3	2/5	2	1/4
JSG	**rel**	Skoč je-li (CY)=0 (hod. znam.)	3	2/5	2	1/4
JSGE	**rel**	Skoč je-li (CY)=0 ∪ (Z)=1,znam.	3	2/5	2	1/4
TRAP	**---**	Skoč na adresu přerušení trap	2	10	1	9
NOP	**---**	Prázdná operace	1	1	1	1

Bitové operace

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Binární		Zdrojový	
			Byty	Stavy	Byty	Stavy
CLR	**CY**	Nuluj CY	1	1	1	1
	bit 51	Nuluj bit51	2	2♥	2	2♥
	bit	Nuluj bit	4	4	3	3
SETB **CPL**	**CY**	Nastav, invertuj CY	1	1	1	1
	bit 51	Nastav, invertuj bit51	2	2♥	2	2♥
	bit	Nastav, invertuj bit	4	4♥	3	3♥
ANL **ORL**	**CY, bit 51**	CY=CY∩ nebo ∪ bit51	2	1♣	2	1♣
	CY, bit	CY=CY∩ nebo ∪ bit	4	3	3	2
ANL/ **ORL/**	**CY, bit 51**	CY=CY∩ nebo ∪ negace bit51	2	1♣	2	1♣
	CY, bit	CY=CY∩ nebo ∪ negace bit	4	3	3	2
MOV	**CY, bit 51**	Přesun obsahu bitu51 do CY	2	1♣	2	1♣
	CY, bit	Přesun obsahu bitu51 do CY	4	3♣	3	2♣
	bit 51, CY	Přesun obsahu CY do bitu51	2	2♥	2	2♥
	bit, CY	Přesun obsahu CY do bit	4	4♥	3	3♥

Sdružené instrukce

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Binární		Zdrojový	
			Byty	Stavy	Byty	Stavy
CJNE	**A, adr8, rel**	Je-li (A) ≠ (adr8), skoč na rel.adr	3	2/5	3	2/5
	A, #data, rel	Je-li (A) ≠ data, skoč na rel.adr	3	2/5	3	2/5
	Rr, #data, rel	Je-li (Rr) ≠ data, skoč na rel.adr	3	2/5	4	3/6
	@Ri, #data, rel	Je-li ((Ri)) ≠ data, skoč na rel.adr	3	3/6	4	4/7
DJNZ	**Rr, rel**	Rr=Rr-1, není-li 0 skoč o rel.adr	3	2/5	3	3/6
	adr8, rel	adr8=adr8-1,není-li 0 skoč o rel.adr	3	3/6	3	3/6

Vysvětlivky jsou na str. 94.

Vysvětlivky

Rr	- registr aktivní banky (R0 ÷ R7) = 8051
Ri	- registr aktivní banky R0 a R1 = 8051
Rn	- registr aktivní banky (R0 ÷ R15) = 80251
@Ri	- nepřímý přístup do datového prostoru, adresa je obsažena v R0, R1
@WRj	- nepřímý přístup do paměti 00:0000H až 00:FFFFH, j=0,2, .. 30
@DRk	- nepřímý přístup do paměti 00:0000H až FF:FFFFH, j=0,4, .. 28,56,60
adr8	- osmibitová přímá adresa vnitřního datového prostoru
adr16	- šestnáctibitová přímá adresa paměťového prostoru
#data	- osmibitová konstanta
#data16	- šestnáctibitová konstanta
short	- 1,2 nebo 4
adr11	- jedenáctibitová adresa programové paměti
adr16	- šestnáctibitová adresa programové paměti
@DPTR	- nepřímý přístup do vnější datové paměti
bit51	- osmibitová adresa bitu, bit = symbolické označení bitu
rel.adr	- osmibitová hodnota pro relativní adresaci (od -128 do 127)
symbol	- přímá adresa registru nebo paměťového místa (adresa střadače ≡ ACC)
(symbol)	- je-li symbol uzavřen v závorce jedná se o obsah příslušného registru nebo paměťového místa
((symbol))	- obsah adresy adresované obsahem příslušného registru nebo paměťového místa

♣ Je-li paměť adresována branami P0 až P3, pak přidejte k době 1 stav

♥ Je-li paměť adresována branami P0 až P3, pak přidejte k době 2 stavy

Je-li uvedeno **n/m** stavů, potom **n** stavů platí pro neplatnou a **m** pro platnou podmínku

Závěrem lze říci, že procesor 80C251SB výrazně snižuje nevýhody procesoru 8051, kterými je neexistence 16bitových operací. Díky 6x rychlejšímu časování a bohaté škále 16bitových operací se jedná o výkonný procesor blížící se svým výkonem k 16bitovým procesorům. Rychlost procesoru nyní omezuje 8bitová vnější sběrnice. Proto se výkon procesoru se zvyšuje, jestliže je program uložen ve vnitřní paměti procesoru, odkud jsou instrukce čteny po slovech. Dalšího zvýšení výkonu dosáhneme konfigurací procesoru do architektury MCS251 (kódově odlišné od procesoru 8051). Pak se však již nejedná o naši oblíbenou řadu 8051.

5. Mikroprocesory Philips XA

Mikroprocesory řady P51XA jsou dalším typem 16bitových procesorů, které vycházejí a architektury procesoru 8051. Oproti procesorům Intel však nejsou ani obvodově, ani programově přímo záměnné. Procesor byl vyvinut firmou Philips Semiconductors pod označením XA (eXtended Architecture) pro uživatele, kteří potřebují větší výkon, kapacitu přímo adresovatelné paměti větší než 64 kB nebo budou navrhovat systémy s multitaskovým operačním systémem. Výkon rozšířené architektury je 10 až 100krát vyšší, než u procesoru 80C51.

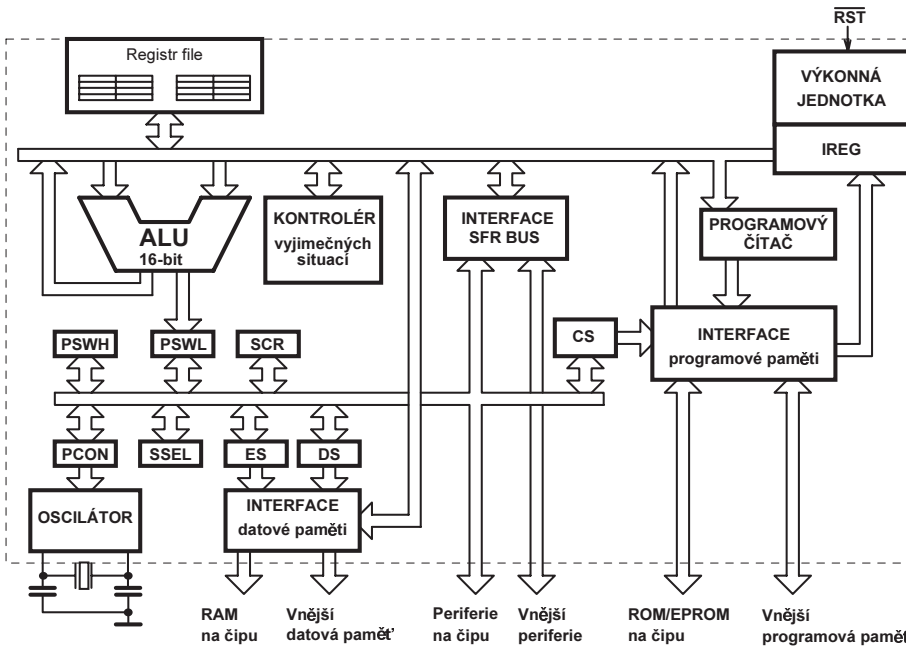
Procesory řady P51XA mají plně statickou 16bitovou centrální procesorovou jednotku s až 24bitovým adresovatelným prostorem pro program i data. Jádro procesoru obsahuje osm 16bitových registrů, z nichž každý se může účastnit aritmetické a logické operace jako zdrojový i cílový registr nebo může být použit jako ukazatel paměti (pointer). Instrukční soubor je rozšířen o bezznaménkové násobení 16x16 bitů, dělení 32/16 bity a 32bitový posun. Svoji architekturou i výkonem podporuje realizaci systémů s časově přepínanými úlohami a programů v reálném čase (např. číslicovou filtraci) až s 32 přerušeními, z toho 16 programovými.

V dalším textu se omezíme na obecné vlastnosti procesorů řady P51XA, případně popíšeme konkrétní vlastnosti prvních zástupců této třídy procesorů P51XAGxx, které mají následující vlastnosti:

- 20bitovou adresu umožňující lineárně adresovat až 1 Mbit paměti
- registrové jádro CPU s registry přístupnými po bytech, slovech
- výrazně rozšířený instrukční soubor
- napájecí napětí 2,7 až 5,5 V (EPROM a OTP 5 V ±5 %)
- 32 kilobytů programové paměti EPROM/ROM na čipu
- 512 bytů paměti RAM na čipu procesoru
- tři čítače/časovače
- hlídací obvod (watchdog timer)
- 2 rozšířené sériové kanály UART
- čtyři 8bitové I/O brány s programovatelnou konfiguraci
- 44vývodové pouzdro PLCC nebo LQFP

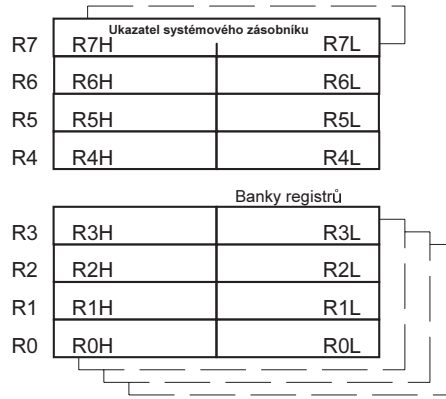
Jádro procesoru XA využívá částečné překrývání instrukcí (pipeline) a vykonává tak některé operace současně (paralelně). Čtení instrukce a její dekódování je prováděno současně s vykonáním předcházející instrukce. Tím je zrychleno vykonávání instrukcí, které pro většinu operací registr - registr trvá 3 hodinové cykly tj. 100 ns při hodinovém kmitočetu 30 MHz. Ke zvýšení výkonu napomáhá blok čtení instrukcí a dekódování, který obsahuje 7 bytů přednačtených instrukcí (obdoba procesorů řady 8086). Díky tomu nemusí výkonná jednotka čekat na čtení instrukce a má stále připravenou novou instrukci k vykonání. Tento blok zároveň rozlišuje, zda přístup do programové paměti bude na čipu nebo do vnější paměti.

Na *obr. 53* jsou zobrazeny hlavní funkční bloky jádra procesoru řady XA, které je obklopeno vnitřními periferiemi, pamětí dat a programu závislými na typu procesoru. Aritmetická jednotka procesoru XA je 16bitová a umožňuje práci s 8bitovými i 16bitovými operandy a v několika případech (instrukcích) i s 32bitovými hodnotami (posun a dělení). K jádru procesoru XA je přidružena řada speciálních funkčních registrů SFR umožňujících konfiguraci procesoru i jednotlivých periferních obvodů. Na rozdíl od procesoru 8051 je procesor vybaven 8bitovým konfiguračním registrem SCR, který umožňuje nastavení základních operačních módů XA, kódovým (CS), datovým (DS) a zvláštním (ES) segmentem, které obsahují číslo aktivní programové a datové paměťové stránky (64 kB). Registr segmentového výběru SSEL potom určuje, který segmentový ukazatel bude v instrukci použit k adresování (DS nebo ES pro přístup do datové paměti, CS nebo čítač instrukcí PC pro přístup do programové paměti).



Obr. 53 Hlavní funkční bloky jádra procesoru XA

Architektura procesorů XA i jejich instrukční soubor je optimalizován na operace se souborem registrů. Ve struktuře může být definováno až 16 16bitových registrů R0 až R15, z nichž zatím bývá implementováno pouze 8 (R0 až R7). Protože jádro procesoru neobsahuje střadač, nahrazují jeho funkci registry R0 až R7. Registr R7 navíc představuje ukazatel zásobníku *obr. 54*. Registry se mohou



Obr. 54 Struktura souboru registrů

účastnit 16bitových operací (např. MOV R7,R4), 8bitových operací (např. MOV R7H,R5L) a jsou i bitově přístupné. Aritmetické i logické operace mohou být realizovány mezi registry nebo registry a datovou pamětí. Spodní čtyři registry R0 až R3 jsou bankovány do čtyřech skupin v závislosti na nastavení bitů RS1 a RS0 v horní části PSW. Registry z neaktivních bank nejsou adresově dostupné. Této vlastnosti se využívá převážně ke zrychlení přístupu do podprogramu obsluhy přerušení. Dvojice registrů je potom využívána některými instrukcemi jako 32bitový posun, násobení a dělení. Hlavní rozdíl od procesoru 8051 spočívá v tom, že horní čtveřice není bankována. Nejvyšší registr R7 zastává funkci ukazatele zásobníku. Všechny registry vyjma registrů ze tří neaktivních bank jsou bitově adresovatelné.

Přerušovací systém procesoru je výrazně odlišný od procesoru 8051 a to jak počtem a typy přerušení, tak rozsahem úrovní přerušení. V rozšířené části PSWH je uložena maska přerušení, která určuje aktuální prioritu zpracovávaného kódu (programu nebo přerušení). Má-li přerušení vyšší prioritu je zpracováno, v opačném případě musí čekat. Protože lze aktuální úroveň priority programu i přerušení měnit, získává programátor velkou variabilitu a přizpůsobivost přerušovacího systému řešené úloze. Kontrolér výjimečných událostí je jednou částí přerušovacího systému s prioritou vyšší než mají obvodová a programová přerušení. Zdroji těchto neodkladných přerušení jsou abnormální události, které v procesoru nastaly. Mezi tyto události patří nulování procesoru, dělení nulou, přetečení zásobníku, návrat z přerušení atd. Zcela odlišné je i zpracování vlastního přerušení včetně nulovacího signálu. Procesor obsahuje dva nezávislé zásobníky, z nichž jeden je systémový a druhý uživatelský. Systémový zásobník leží vždy v prvním segmentu datové paměti a uživatelský je umístěn v segmentu daném aktuální hodnotou registru DS. Běžící program má přístup pouze k jednomu zásobníku, jehož maximální rozsah je 64 kB. Odlišností od procesoru 8051 je snižující se

adresa ukazatele zásobníku při jeho plnění. Ukazatel zásobníku ukazuje na poslední obsazenou hodnotu, a proto je při ukládání nejprve jeho hodnota dekrementována o hodnotu 2 a teprve potom je realizován zápis 16bitové hodnoty. Data v zásobníku zabírají vždy sudý počet bytů a jsou ukládána po slovech.

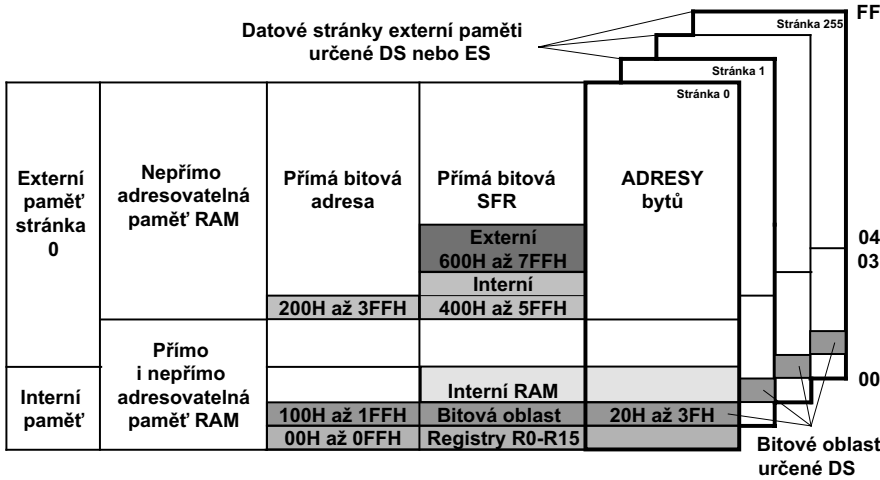
Procesor je jako většina nejmodernějších procesorů vybaven emulačním a testovacím systémem na čipu procesoru. Systém slouží k testování a analýze systému s procesorem XA-G3 bez nutnosti jeho vyjmutí ze systému. Do emulačního módu je procesor přiveden touto posloupností signálů. Nejprve je procesor vynulován signálem $\overline{\text{RST}}$, potom jsou vývody ALE, P1.3, P1.2, P1.1 a P1.0 nastaveny do log. 0 a vývod PSEN do log. 1. Je-li následně signál $\overline{\text{RST}}$ nastaven do log. 1 při nastavených podmínkách, potom procesor přechází do módu emulace označovaného ONCE.

5.1. Organizace paměti

Procesor XA má harvardskou strukturu s odděleným programovým a datovým paměťovým prostorem. Maximální přímo adresovatelný prostor je až 16 MB, ale nebude všemi procesory z této skupiny podporován. Na začátku datové paměti obr.55 je umístěna vnitřní paměť procesoru, jejíž velikost bude od 0,5 kB do 1 kB. Nad touto pamětí již začíná externí datová paměť, která je rozdělena do segmentů (stránek) po 64 kB. V interní datové paměti jsou umístěny registry R0 až R7 (případně až R15), za kterými je umístěna bitově adresovatelná část paměti (32 bytů) a za ní následuje vnitřní datová paměť omezená kapacitou odpovídající danému typu procesoru ($\leq$ 1 kB). První kilobyte datové paměti na každé stránce lze adresovat přímo i nepřímo (000h až 3FFH). Na stránce určené registrem DS je adresový prostor 20H až 3FH bitově přístupný přes bitové adresy 100H až 1FFH. Dochází-li k překrývání vnější a vnitřní datové paměti, potom přednost má vnitřní datová paměť. Přístup do vnější zakryté paměti je možný pouze přes instrukci MOVX. Je-li využíván větší prostor než 64 kB, potom je nezbytné využívat segmentových registrů DS nebo ES k adresování paměti.

Programová paměť procesoru se skládá z paměti ROM nebo EPROM umístěné na čipu a externí paměti programu. Začátek vnější paměti programu je dán velikostí vnitřní paměti daného typu procesoru. Většina procesorů XA bude umožňovat vytvoření vnější sběrnice pro připojení programové a datové paměti. Bude-li docházet k překrývání vnitřní a vnější programové paměti, potom celá vnější programová paměť je využitelná pouze po nastavení vývodu procesoru EA = log. 0. Přístup k uloženým konstantám (obsahy tabulek) je možný pouze přes instrukci MOVC s využitím čítače instrukcí (PC) (bit SSEL=0) nebo programového segmentu (CS) (bit SSEL=1). Přístup k datům může být 1, 4, 5, 8 nebo 16bitový.

Speciální funkční registry SFR umožňující konfiguraci periferních obvodů, vstupů a výstupů leží v separátním datovém prostoru přístupném pouze pomocí přímé adresy. Díky tomu je každý funkční registr programem přístupný nezávisle



Obr. 55 Organizace datové paměti procesoru XA

na jakémkoliv ukazateli nebo segmentovém registru. Adresy SFR jsou vždy obsaženy v instrukci a nachází se v adresovém prostoru 400H,7FFH *obr. 55*. Tento prostor je rozdělen na dvě části po 512 bytech. První přísluší registrům na čipu procesoru, druhá je rezervována pro adresování registrů vnějších periferních obvodů. Tato vlastnost však nebyla na stavajících typech procesorů implementována. Registry umístěné v prvních 64 paměťových místech (adresy 400H až 43FH) jsou bitově adresovatelné pomocí bitových adres 200H až 3FFH. Všechny bitově adresovatelné funkční registry procesoru proto musí ležet v tomto prostoru.

5.2. Adresovací módy

Přímo adresovatelný prostor procesoru XA je 16 MB (24bitová adresa) a každá adresa umožňuje adresovat 1 byte (8 bitů). Díky své struktuře a možnosti připojení 8 nebo 16bitové vnější datové a programové paměti, umožňuje procesor práci s 8bitovými daty (byty) i s 16bitovými daty (slovy). Přenášené slovo podléhá uspořádání Little-Endian, podle kterého nižší byte je umístěn na nižší adrese (A0 = 0) a vyšší na vyšší adrese (A0 = 1). Proto je 16bitový přístup do paměti realizován vždy se sudou adresou (A0 = 0). V opačném případě je automaticky adresa upravena na nejbližší nižší sudou adresu, se kterou je operace realizována. Interně umožňuje procesor pracovat i s 32bitovým operandem, který může být umístěn pouze v jedné ze čtyř nebo osmi dvojic registrů tj. (R0, R1), (R2, R3), ..., (R6, R7), případně až (R14, R15). Procesor XA umožňuje přímé a nepřímé adresování datové paměti, nepřímé adresování s posunem (offsetem)

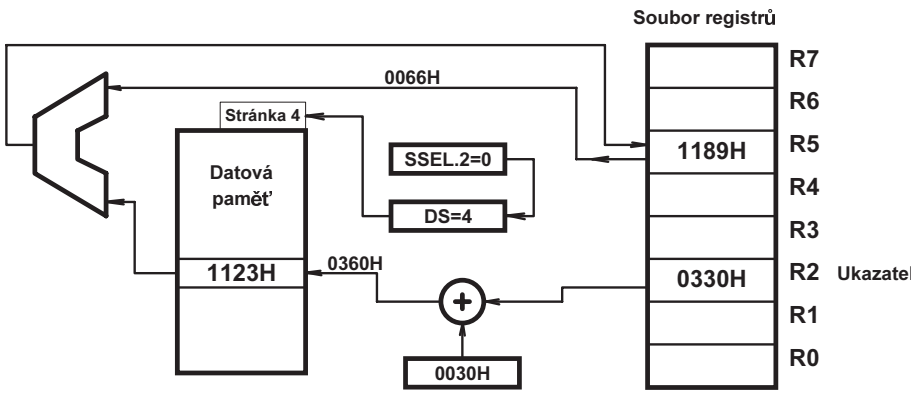
a pro realizaci některých úloh z číslicového zpracování signálu i nepřímé adresování s automatickou inkrementací. Sedm adresovacích módů umožňuje přístup k operandům o šířce 1, 4, 5, 8 nebo 16 bitů.

- ***Přímé adresování.*** První 1 kB každého segmentu datové paměti je přístupný přes instrukce obsahující přímou 10bitovou adresu v rozsahu 000H až 3FFH. Celá 24bitová adresa je vytvořena z obsahu registru DS (A23 až A16), bity A15 až A10 jsou nulové a bity A9 až A0 jsou převzaty z instrukčního slova.

- ***Registrové adresování.*** Je adresování umožňující přenos bytů nebo slov v oblasti souboru registrů (např. ADD R6, R4).

- ***Nepřímé adresování.*** Instrukce využívající nepřímé adresování (např. ADD R6, [R4]) obsahují identifikaci 16bitového ukazatele adresového pole. Ukazatel (R4), který je tvořen jedním registrem Rn (n = 0 až 6) ze souboru registrů, adresuje operand v rámci 64 kB jedné stránky paměti. Úplná 24bitová adresa se skládá z 8bitového segmentového registru a 16bitového ukazatele. Segmentovým registrem může být buď registr DS (SSEL.n = 0) nebo registr ES (SSEL.n = 1) pro přístup do datové paměti. Pro přístup do programové paměti určuje horní adresové bity registr PC (SSEL.n = 0) nebo registr CS (SSEL.n = 1), kde n představuje pořadové číslo registrového ukazatele a SSEL.n představuje n-tý bit registru SSEL. Adresa pro práci se slovem musí být opět sudá. Je-li při tomto adresování využito inkrementace registru (např. ADD R1, [R3+]), potom obsah registru bude inkrementován v závislosti na šířce přenášených dat. Pro bytovou operaci bude registr zvětšen o hodnotu 1, pro operaci se slovem bude registr zvětšen o hodnotu 2.

- ***Nepřímé adresování s offsetem.*** Tento typ adresování, který se využívá při adresování maticových polí nebo k adresování proměnných přenášených do podprogramů přes zásobník, je shodný s předcházejícím typem s tím rozdílem, že



Obr. 56 Nepřímé adresování s ofsetem pro instrukci ADD R5, [R2+30H]

k 16bitovému ukazateli tvořeného registrem je přičten 8 nebo 16bitový posun se znaménkem. Teprve takto získaná hodnota adresy modulo 2^{16} určuje operand v 64 kB stránce (např. ADD R5, [R2+30H]) *obr. 56*.

- ***Adresování speciálních funkčních registrů.*** Toto adresování je shodné s přímým adresováním, pouze adresa se pohybuje v intervalu 400H až 7FFH (např. ADD R0L, 458H nebo ADD ROL, TL2). Adresování bylo již popsáno v předcházejícím odstavci.

- ***Bezprostřední adresování.*** Při tomto adresování je operand explicitně určen v instrukci a představuje 4, 8 nebo 16bitovou konstantu (např. MOV R6L,#0B9H nebo MOV.b R6,#0B9H). Čtyřbitová konstanta je využívána pouze v instrukcích ADDS a MOVS.

- ***Bitové adresování.*** Při tomto adresování obsahuje instrukce 10bitovou adresu bitového operandu. Adresové pole bitů je rozděleno do tří částí: 256 bitů je definováno v souboru registrů (registry R0 až R3 z aktivní banky), 256 bitů je definováno v přímo adresovatelné části datové paměti (adresy 20H až 3FH) v aktivní stránce paměti určené registrem DS a 512 bitů je definováno v oblasti SFR.

5.3. Čítače a časovače XA-G3

Procesor XA je vybaven dvěma standardními 16bitovými čítači/časovači T0 a T1. K nim je přiřazen třetí 16bitový časovač T2, který může být konfigurován jako vzestupný nebo sestupný. Čítače/časovače mohou sloužit k měření a generování časových intervalů, k čítání vnějších událostí, k pravidelnému generování přerušení a generování pulzně šířkové modulace. Procesor je vybaven generátorem, který vytváří časovou základnu hodinového signálu pro všechny časovače procesoru. Hodinový kmitočet časovačů lze konfigurovat bity **PT1** a **PT0** z registru **SRC** na hodnotu Fosc/4, Fosc/16 a Fosc/64. Jsou-li T0, T1 a T2 konfigurovány jako čítače vnějších událostí, potom jejich hodnota je zvětšena s příchodem sestupné hrany vnějšího signálu. Tento signál je testován jednou za dva hodinové cykly, a proto maximální čítaný kmitočet může dosáhnout hodnoty Fosc/4. Na střídě vstupního signálu nezáleží. Pro zajištění správné činnosti čítače však musí log. 0 i log. 1 trvat alespoň dva cykly oscilačního kmitočtu.

Čítač 0 a čítač 1

Tyto čítače jsou kontrolovány standardními bity z registru TMOD. Rozdíl proti procesoru 80C51 spočívá v tom, že mód 0 (původně 13bitový čítač) byl ve struktuře XA nahrazen mnohem užitečnějším 16bitovým čítačem s obvodovým přednastavením (16bitová verze módu 2). Do struktury byly přidány čtyři 8bitové registry RTH0, RTL0 a RTH1 a RTL1, které obsahují přednastavenou hodnotu. Dojde-li v tomto módu k přetečení čítače, je nastaveno návěští TFn v registru TCON a registry THn a TLn jsou naplněny obsahem RTHn a RTLn. Další rozdíl

proti standardnímu módu 2 spočívá v tom, že přednastavená hodnota je uložena v registru RTLn a nikoliv v registru THn. Časový interval v módech 0 a 2 vypočteme ze vztahů

$$Čas = F_{osc}/(N*(65536 – RTHn,RTLn)) = F_{osc}/(N*(256 – RTLn))$$

kde N je hodnota 4, 16 nebo 64, nastavená časovací jednotkou. Novinkou, která zjednoduší řadu aplikací, je možnost vyvedení výstupu čítače. Vývody T0 a T1, které se využívají jako vstupy vnějších událostí, mohou být využity jako výstupy přetečení. O této konfiguraci rozhoduje bit **T0OE** a **T1OE** v registru **TSTAT** *obr. 57*. Výstup může sloužit ke generování PŠM v závislosti na změnách přednastavené hodnoty nebo jako výstup periodického signálu v rozmezí Fosc/8 až Fosc/8388608 v závislosti na přednastavené hodnotě a předděliči interního oscilátoru. Pro kmitočet oscilátoru 30 MHz je rozsah od 3,58 Hz do 3,75 MHz.

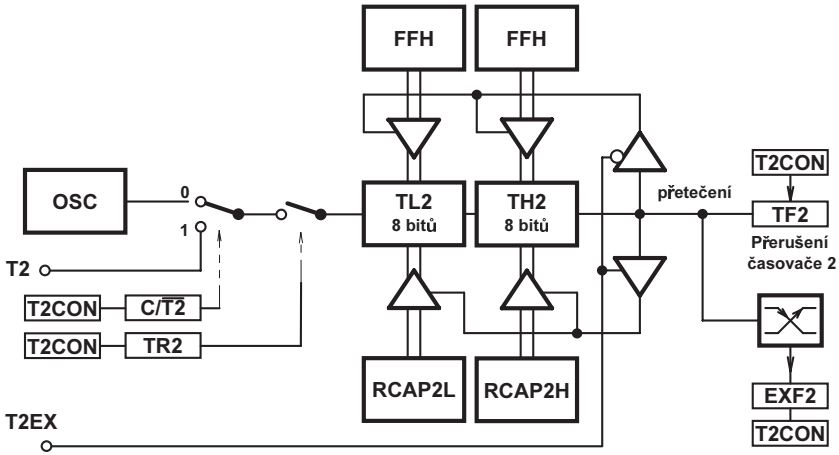
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
---	---	---	---	---	T1OE	---	T0OE	Adresa RAM = 411H

Obr. 57 Rozložení bitů v registru TSTAT

Čítač 2

Čítač 2 může pracovat ve třech módech již zmíněných na str. 37 - 16bitový záchytný systém, 16bitový čítač s přednastavením a generátor přenosové rychlosti. Ve struktuře procesoru XA je funkce záchytného systému stejná, jako bylo popsáno v části o procesoru 8052. Režim čítače s přednastavením je funkčně rozšířen na základě bitu **DCEN** v registru **T2MOD**. Je-li bit **DCEN** = 0, čítač 2 čítá vzestupně v módu 16bitového čítače s přednastavením. V opačném případě se směr čítání řídí hodnotou přivedenou na vývod procesoru **T2EX**. Je-li T2EX = 1, potom čítač čítá vzestupně a po přetečení z hodnoty FFFFH se přednastaví na hodnotu uloženou v registrech T2CAPH a T2CAPL. Zároveň je nastaven příznak přetečení TF2, od kterého může být vyvoláno přerušení. Je-li T2EX = 0, čítač čítá sestupně a zároveň je porovnáván s hodnotou uloženou v registrech T2CAPH a T2CAPL. Dojde-li ke shodě stavu čítače s přednastavenou hodnotou, je čítač přednastaven na hodnotu FFFFH. Příznak TF2 (nyní podtečení) je nastaven a může být na jeho základě vyvoláno přerušení *obr. 58*. Využití časovače 2 jako generátoru přenosové rychlosti je shodné s popisem u procesoru 8052. Odlišnost spočívá v tom, že u struktury XA jsou sériové kanály dva. Proto byly do registru T2MOD přidány bity RCLK1 a TCLK1, které umožňují přiřadit časovač 2 jako generátor přenosové rychlosti i druhému sériovému kanálu.

Jednotlivé dodatečné funkce časovače nebo čítače 2 se volí nastavením nebo vynulováním bitů v registru **T2MOD** *obr. 59*, jehož jednotlivé bity jsou bitově přístupné a mají následující význam.



Obr. 58 Režim časovače 2 s přednastavením pro DCEN = 1

♣ **RCLK1** - Přijímající příznak sériového kanálu 1. Je-li RCLK1 = 1, časovač 2 se využívá jako zdroj hodinového signálu přijímající části sériového kanálu 1.

♣ **TCLK1 -** Vysílací příznak sériového kanálu 1. Je-li TCLK1 = 1, časovač 2 se využívá jako zdroj hodinového signálu vysílací části sériového kanálu 1. V opačném případě je zdrojem hodinového signálu časovač T1.

♣ **T2OE** - Je-li T2OE = 0, vývod T2 se stává hodinovým signálem čítače 2 v módu čítače. Je-li T2OE = 1, vývod T2 se stává výstupem, který se mění s každým přetečením T2.

♣ **DCEN** - Směr čítání časovače 2. Je-li DCEN = 1, čítač 2 čítá vzestupně. Je-li DCEN = 0, čítač 2 čítá oběma směry v závislosti na vývodu T2EX (T2EX = 1 vzestupně, T2EX = 0 sestupně).

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
---	---	RCLK1	TCLK1	---	---	T2OE	DCEN	Adresa RAM = 419H

Obr. 59 Rozložení bitů v registru T2MOD

Stejně jako v případě časovačů T0 a T1 lze čítač 2 využívat jako programovatelný generátor se střídou 1 : 1, jehož výstup lze připojit na vývod T2 (P1.6) pomocí bitu **T2OE** v registru **T2MOD**.

Časovač watchdog

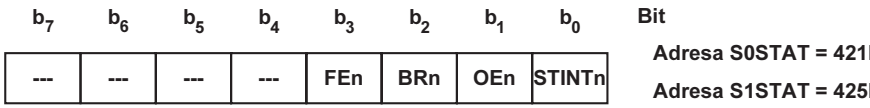
Procesor XA je jako většina procesorů určených pro průmyslové aplikace vybaven ochranným systémem proti nesprávné interpretaci kódového slova. Jestliže k takové situaci dojde, nejčastěji v důsledku přepěťové špičky v napájecím napětí, nebude časovač watchdog včas přednastaven a následně vygeneruje nulovací signál procesoru. Časovač watchdog je tvořen 8bitovým dekrementujícím čítačem s 13-bitovým předděličem. Předdělič je konfigurovatelný třemi bity PRE2, PRE1 a PRE0 definující tyto dělící poměry: 32 (PRE2 = PRE1 = PRE0 = 0), 64, 128, 256, 512, 1024, 2048, 4096 (PRE2 = PRE1 = PRE0 = 1). Do vlastního předděliče je přiveden hodinový signál z generátoru hodinového signálu určeného pro čítače T0, T1 a T2 (F_{osc}/4, F_{osc}/16 nebo F_{osc}/64). Díky tomu je minimální nastavitelná doba pro vynulování procesoru čítačem watchdog $t_{min} = F_{osc} * 4 * 32$ (WD = 0, N = 4) a maximální doba $t_{max} = F_{osc} * 64 * 4096 * 256$ (WD = 255, N = 64). Obecně můžeme tuto dobu vyjádřit vztahem $t_{WD} = F_{osc} * N * P * (WD + 1)$, kde WD je přednastavená hodnota čítače watchdog, N je hodnota předděliče oscilačního kmitočtu a P je dělící poměr předděliče. Přednastavení se provádí dvěma po sobě následujícími zápisy hodnoty A5H do registru WFEED1a hodnoty 5AH do registru WFEED2. Jestliže tyto dva zápisy nebudou provedeny ihned po sobě, bude následovat vynulování procesoru. Je-li v procesoru používán přerušovací systém, nemusí být tyto zápisy provedeny po sobě a procesor bude vynulován. Z tohoto důvodu doporučuje výrobce před uvedeným zápisem do registrů zakázat globální masku přerušení a po jejich provedení ji opět povolit. Například takto:

```
CLR     EA              ; Zakázaná přerušení
MOV.b   WFEED1, #0A5H   ; První slovo přednastavení
MOV.b   WFEED2, #05AH   ; Druhé slovo přednastavení
SETB    EA              ; Povol přerušení
```

5.4. Sériové kanály

Procesor XA-G3 obsahuje 2 rozšířené sériové kanály (UART). První odlišnost od standardního procesoru spočívá v rozdělení vektoru přerušení pro příjem a vysílání pro každý sériový kanál. Každý kanál byl vybaven detekcí přerušení přijímané posloupnosti, která pracuje nezávisle na sériovém kanálu. Výstup logiky je vyveden do příznaku BRn registru SnSTAT a indikuje případ, kdy všechny přijaté bity včetně stopbitu jsou nulové. Sériový kanál je dále vybaven indikací špatného rámce FEn, která indikuje chybnou hodnotu STOP bitu. Posledním novým návěštím je chyba přepsání (Overrun Error), indikující ztrátu znaku v přijímaném řetezci, tj. znaku, který nebyl včas obslužným programem přečten. Umístění jednotlivých příznaků v registru SnSTAT je zobrazeno na *obr. 60*.

♣ **FEn -** Chyba rámce (Framing error) je nastavena do log. 1, jestliže při příjmu znaku je přijata nesprávná (log. 0) hodnota STOP bitu. Nuluje se programově.



Obr. 60 Rozložení bitů v registru SnTSTAT

♣ **BRn** - Příznak detekce přerušení je nastaven, jestliže je přijat znak se všemi nulovými bity včetně STOP bitu. Příznak je nastaven na 8 bitu v módu 1 a na 9 bitu pro módy 2 a 3. Nuluje se programově.

♣ **OEn** - Příznak přepisu přijatého znaku je nastaven do log. 1, jestliže nově přijatý znak je uložen do přijímacího registru, aniž by byl předcházející znak programem přečten (příznak RI z registru SnCON zůstal nastaven). Nuluje se programově.

♣ **STINTn** - Příznak povolení výše uvedených návěští současně s generovaným požadavkem o přerušení RIn. Může být vynulován pouze zápisem do tohoto registru.

Každý UART je synchronizován hodinovým signálem získaným dělením kmitočtu oscilátoru procesoru (mod 0 a 2) nebo odvozeným z přetečení časovače 1 nebo 2 (mód 1 a 3). V módu 1 a 3 je standardně využíván časovač T1, který lze nahradit naprogramováním bitů R0CLK a T0CLK v registru T2CON a R1CLK a T1CLK v registru T2MOD. Přijímající i vysílající registr je přístupný přes speciální funkční registr označený SnBUF.

Mód 0 je stejný jako u 80C51 s tím, že přenosová rychlost je pevně dána 1/16 kmitočtu oscilátoru. V módu 2 je přenosová rychlost pevně dána kmitočtem $F_{osc}/64$. V módech 1 a 3 je přenosová rychlost určena přednastavenou hodnotou (8 nebo 16 bitů) a přednastaveným poměrem předděliče oscilátoru ($F_{osc}/4$, $F_{osc}/16$ nebo $F_{osc}/64$). Protože synchronizační hodiny pro sériový kanál jsou 16násobek přenosové rychlosti, je maximální přenosová rychlost v módu 1 a 3 dána hodnotou $F_{osc}/64$.

5.5. Multiprocesorová komunikace

Procesor je vybaven logikou klasické multiprocesorové komunikace 80C51, která je rozšířena o logiku automatického rozpoznávání adresy. Ta umožňuje nahradit programové rozpoznávání přenášených adres obvodovými prostředky. Logika rozpoznávání je aktivní v módu 2 a 3 sériového kanálu při nastaveném bitu SM2 = 1 v registru SCON. Je-li adresa pro daný procesor povolena, je nastaven příznak příjmu RIn daného sériového kanálu. Ostatní adresy jsou ignorovány a procesor není přerušován. Použití automatického rozpoznávání adresy umožňuje selektivní komunikaci s jedním nebo více podřízenými procesory. Automatické rozpoznávání se skládá z číslicového komparátoru a registrů SADDR a SADEN.

Registr SADDR obsahuje adresu daného podřízeného procesoru. Přijatá adresa je před porovnáním s SADDR vynásobena (logicky) registrem SADEN. Tento registr určuje, které bity přijaté adresy jsou pro vyhodnocení důležité a které jsou nezajímavé. Jako příklad mějme tři podřízené procesory Slave0, Slave1 a Slave2. Každý procesor může mít v registru SADDR uloženu svoji jedinečnou adresu nebo stejnou hodnotu, která ve spolupráci s adresovou maskou SADEN umožňuje efektivní selektivní komunikaci. V tab.10 je zobrazena situace, umožňující všechny kombinace přístupu k podřízeným procesorům. Všechny procesory musí mít v adresových bitech b7 až b5 hodnotu log. 1, v bitech b4 a b3 hodnotu log. 0. Dále je zřejmé, že procesor Slave0 musí mít v bitu b0 hodnotu log. 0, procesor Slave1 musí mít b1 = 0 a Slave2 musí mít bit b2 = 0. Díky této vlastnosti lze adresovat všechny procesory adresou 11100000, procesory 0 a 1 adresou 11100100, procesory 1 a 2 adresou 11100001, procesory 0 a 2 adresou 11100010. Každý z procesorů může být adresován takto: Slave0 - 11100110, Slave1 - 11100101 a Slave2 - 11100011. Při využívání všech bitů ke kombinačnímu adresování je jedinečná adresa procesoru dána vymaskováním registru SADDR registrem SADEN. Po vynulování procesoru jsou všechny bity těchto registrů nulové a díky tomu jsou všechny adresy povolené k přijetí.

Procesor	Slave 0	Slave1	Slave2
SADDR	11100000	11100000	11100000
SADEN	11111001	11111010	11111100
Přístupová adresa	11100XX0	11100X0X	111000XX

Tabulka 10

5.6. Přerušovací systém

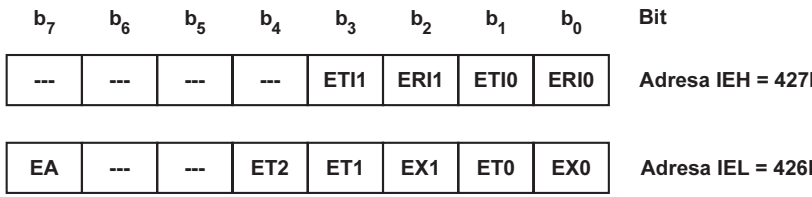
Přerušovací systém procesorů XA je výrazně složitější a variabilnější než u klasického procesoru 51. Architektura procesorů XA rozeznává čtyři typy přerušení: krizová, TRAP, programová a od vnějších nebo vnitřních událostí. ***Krizová (výjimečná) přerušení*** jsou většinou vyvolána abnormální situací, která v systému nastane. Ve struktuře přerušovacího systému se počítá s 16 takovýmito přerušeními, z nichž v současné době je obsazeno 7 přerušení a 9 jich zůstává v rezervě. Mezi tato přerušení patří nulování procesoru (Reset), dělení nulou (Divide by zero), přetečení zásobníku (Stack overflow), trasování (Trace), bod zastavení (Breakpoint), uživatelský návrat z přerušení (User RETI) a nemaskované přerušení (NMI). Všechna tato přerušení jsou nemaskovatelná, a proto jsou vždy přijata. Jejich vykonání však závisí na prioritě právě zpracovávaného programu (míněno programu nebo přerušení).

Druhou skupinu tvoří ***přerušení TRAP***, která jsou určena jako podpora k volání systémových (všeobecně použitelných) podprogramů. Kromě toho umožňují

přerušení TRAP přechod mezi uživatelským a systémovým módem. Přerušení TRAP jsou generována instrukcí **TRAP #n**, kde n∈⟨0,15⟩. Tato přerušení jsou spuštěna jenom po vykonání instrukce TRAP, a proto není potřeba definovat jakékoliv vzájemné priority mezi jednotlivými přerušeními. Přerušení TRAP jsou bezprostřední, nemaskovatelná přerušení, která vyzvednou odpovídající vektor přerušení a spustí podprogram v systémovém módu. Na konci podprogramů před instrukcí RETI není třeba nulovat jakýkoliv bit.

Třetí skupinu přerušení představují ***přerušení od vnějších nebo vnitřních periferií.*** Tato skupina představuje přerušení, která jsou implementována na standardním procesoru 8051. Její architektura je v podstatě shodná, ale variabilita je výrazně větší. Skupina se skládá z maximálně 31 přerušení, z nichž je u procesoru XA-G3 využito zatím 9 a zbývajících 22 zůstává v rezervě. Z devíti použitých se jedná o dvě vnější přerušení, o tři od jednotlivých časovačů a nakonec o čtyři příslušející příjmu a vysílání obou sériových kanálů. Rozdělení zdrojů přerušení od sériových kanálů je první odlišností standardního procesoru. Druhou výraznou odlišnost představuje možnost přiřazení jedné z osmi úrovní priority každému zdroji přerušení. Tím je uživateli umožněno vytvořit prakticky libovolný prioritní řetěz s možností individuálního i globálního maskování jednotlivých přerušení. Maskovací bity jsou uloženy v registrech IEL a IEH *obr. 61*. Úroveň priority každého přerušení určuje čtveřice nebo trojice bitů v jednom z registrů IPA0 až IPA5. U procesoru XA-G3 určuje úroveň priority trojice bitů. Je-li do těchto bitů uložena hodnota 000, úroveň priority je 0 a přerušení nemůže být nikdy vyvoláno. Hodnota 001 nastavuje úroveň 9 až nakonec hodnota 111 nastavuje úroveň priority 15. Možnost zakázat přerušení nulovou úrovní priority přerušení je další odlišností od standardního přerušení 8051. Umístění trojice bitů v registrech IPAn je následující: registr IPA0 bity b6, b5, b4 - priorita T0, bity b2, b1, b0 - priorita INT0, registr IPA1 bity b6, b5, b4 - priorita T1, bity b2, b1, b0 - priorita INT1, registr IPA2 bity b2, b1, b0 - priorita T2, registr IPA3 bity b6, b5, b4 - priorita TxD0, bity b2, b1, b0 - priorita RxD0, registr IPA4 bity b6, b5, b4 - priorita TxD1, bity b2, b1, b0 - priorita RxD1. Význam bitů v registrech TCON, T2CON a S0CON zůstává shodný s popisem u procesoru 8051 a 8052.

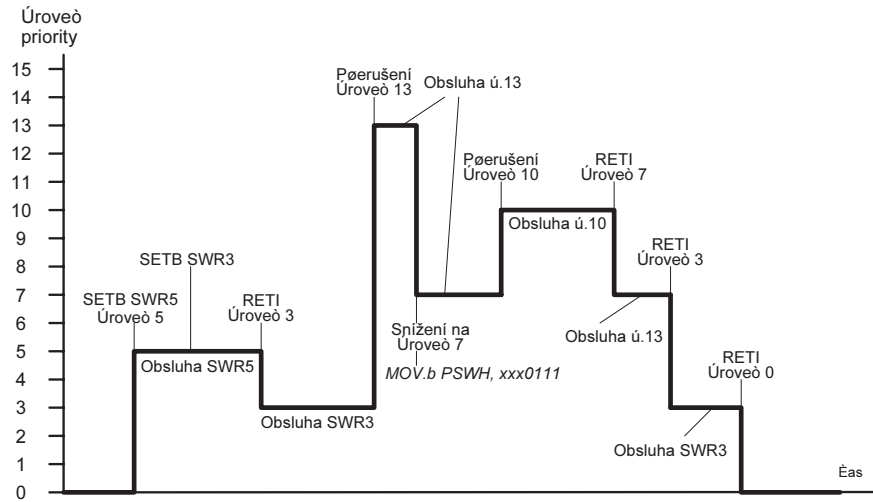
Programová přerušení jsou ekvivalentní přerušením obvodovým (událostí), ale jejich žádosti jsou vyvolány programově. Programových přerušení je stan-



Obr. 61 Rozložení bitů v registrech IEH a IEL

dardně 7 a jsou vyvolána nastavením příslušného bitu v registru žádostí o programové přerušení SWR. Kromě tohoto registru, který je bitově adresovatelný, existuje ještě bitově neadresovatelný registr povolení programových přerušení SWE. V *tab. D5* je zobrazeno umístění bitů jednotlivých přerušení v registrech SWR a SWE. Protože bit povolení programového přerušení není bitově adresovatelný, je třeba využít bytové operace např. OR.B SWE, #01. Vlastní přerušení je potom generováno instrukcí SETB SWR1. Na konci obslužného programu přerušení je třeba vynulovat žádost o přerušení (např. CLR SWR1) před provedením instrukce RETI. Pozor na rozdíl mezi instrukcemi SETB SWR1 (přerušení 1) a SETB SWR.1 (přerušení 2). Programová přerušení mají pevně definované priority přerušení shodné s číslem příslušného přerušení (tj. SWR1 má prioritu 1, SWR2 má prioritu 2, atd.). Vlastní programové přerušení bude zpracováno jedině tehdy, když jeho priorita bude vyšší, než priorita právě zpracovávaného programového kódu.

Velký rozsah úrovní priority a možnost nastavení jiné úrovně priority uvnitř již probíhajícího přerušení dává návrháři systému možnosti v přizpůsobení programového řešení dané aplikaci. Předefinování úrovně priority umožňuje návrháři realizovat například časově kritickou odezvu při vysoké úrovni priority a zbývající časově nekritickou část s úrovní výrazně nižší. Tím může být umožněno i přijetí přerušení s nižší úrovní, které by jinak muselo čekat na dokončení celého přerušení s vysokou prioritou. Tato situace se označuje jako „inverzní priorita“ a je zobrazen na *obr. 62*.



Obr. 62 Inverze úrovně priority v přerušení s úrovní 13

5.7. Zásobníky procesoru

Procesor XA má implementovány dva zásobníky, uživatelský a systémový. Uživatelský může být umístěn kdekoliv v datové paměti, zatímco systémový musí být umístěn pouze v prvních 64 kB tj. na stránce 0. Každý ze zásobníků má svůj ukazatel SSP-systémový a USP-uživatelský. Ukazatel právě aktivního zásobníku (USP nebo SSP) se objevuje jako slovo v registru R7. Který ze zásobníků je aktivní, rozhoduje bit SM v registru PSW (SM = 0 - uživatelský, SM = 1 - systémový). Skutečná adresa uživatelského zásobníku je určena registrem DS a hodnotou USP, systémový zásobník je určen pouze hodnotou SSP (horních 8 bitů je vždy nulových). Zápis i čtení hodnot je prováděno vždy po slovech nezávisle na tom, zda ukládaná hodnota je byte nebo slovo. Adresa je vždy zmenšena nebo zvětšena o hodnotu 2. Z tohoto pohledu je neefektivní ukádání řady bytů instrukcemi PUSH (např. PUSH R1H, PUSH R1L), protože jejich uložení zabere dvojnásobek paměti. Existence nepřímého adresování s ofsetem výrazně usnadňuje předávání parametrů přes zásobník do volaných podprogramů, jak je často využíváno v jazyce C (např. MOV [R7+offset], Rn, MOV Rn, [R7+offset]).

Jak vyplynulo z úvodní informace, je procesor vybaven výjimečným přerušením při přetečení zásobníku. K této situaci dojde, jestliže obsah ukazatele zásobníku zmenší svoji hodnotu z 80H na 7EH. Zároveň zůstává v datové paměti ještě prostor 64 bytů pro obsloužení procesu přetečení zásobníku. Nejhorší případ nastane při ukládání 8 slov s okamžitým generováním přerušení od přetečení zásobníku. K tomu může ještě přijít nemaskované přerušení, které uloží do zásobníku další 3 slova. Prostor 64 bytů je k dispozici pro obsluhu krizové situace. Podtečení zásobníku procesor XA nekontroluje stejně jako přetečení u něhož byl ukazatel zásobníku umístěn pod adresu 80h. Je-li ukazatel nastaven nebo programem přestaven na lichou adresu, potom procesor řeší situaci stejně jako při ukládání slova do datové paměti tj. ignoruje z adresy nejnižší bit A0. Procesor podporuje dva formáty pro ukládání dat do zásobníku. Ve svém přirozeném módu ukládá do zásobníku 3 slova. První ukládanou hodnotou je spodních 16 bitů PC (návratové adresy), následované horní částí PC bity 23 až 16 (vyšších 8 bitů je nulových). Nakonec je uložena hodnota PSW. Pracuje-li procesor v jednostránkovém módu (bit PZ = 1 v registru SCR), potom jsou ukládány pouze dvě slova 16bitová adresa (PC) a PSW. Je velmi důležité zajistit, aby oba formáty nebyly pomíchány a proto je vhodné definovat obsah SCR v inicializaci procesoru.

5.8. Nulování procesoru

Procesor XA se nuluje přivedením úrovně log. 0 na vývod $\overline{\text{RST}}$. Nulovací vstup je vybaven Smithovým obvodem, a proto pro jednoduché aplikace postačí odpor 100 kΩ s kapacitou 1 mF k realizaci nulovacího obvodu. Úkolem obvodu je pro rychle nabíhající zdroj napětí podržet úroveň log. 0 po dobu alespoň 10 ms, za

kterou oscilátor stabilizuje svoji činnost a CPU detekuje nulovací podmínku. Poté již procesor inicializuje svoji vnitřní nulovací posloupnost, která potřebuje 10 hodinových cyklů, během kterých zapíše nulové hodnoty do řady periferních obvodů a speciálních funkčních registrů. Základní nastavení registrů je následující: CS = DS = ES = 0, SSEL = 0 všechny přístupy jsou přes registr DS, vynulovány jsou všechny registry v souboru registrů. Ukazatele zásobníků USP i SSP jsou nastaveny na hodnotu 100H. Konfigurační registr procesoru SCR je vynulován a díky tomu je nastaven do módu 24bitové adresy (PZ = 0), svého přirozeného módu (CM = 0) a generátor hodinového signálu pro periferní obvody je nastaven na $F_{osc}/4$ (PT1 = PT0 = 0). Samozřejmě jsou zakázána všechna maskovatelná přerušení nastavením EA = 0 v registru IE. Nulování neovlivňuje vnitřní a vnější paměť RAM a nastavuje všechny vývody do vstupního stavu (kvazi-obousměrné řízení s hodnotou portu FFH). Při náběžné hraně $\overline{RST}$ jsou testovány bit $\overline{EA}$ / WAIT a BUSW, určující konfiguraci programové paměti a šířku vnější datové sběrnice. Je-li EA = 1 (mód single-chip), potom procesor bude vykonávat program z interní programové paměti, pro EA = 0 (mód bez ROM) bude program vykonáván pouze z externí programové paměti. Po resetu se vývod $\overline{EA}$ / WAIT stává vývodem pro synchronizaci přenosů po vnější sběrnici. Nakonec je vykonáno ***nulovací přerušení***, které z programové paměti z adres 1h,0h a 3h,2h přečte budoucí obsah registru PSW a adresu počátečního kódu. Počátek programu bude vypadat následně:

```
            code_seg                ; počátek programového segmentu
            org     0               ; program začíná na adrese 000000h
                                    ; tabulka vektorů přerušení
                                    ; nulovací přerušení
            DW      počáteční_ PSW  ; definice 16-bitové konstanty
            DW      startup_c       ; definice 16-bitové konstanty - adresy

            org     120h            ; prostor za tabulkou vektorů přerušení
startup_c:                          ; počátek inicializace procesoru
```

Nastavení hodnoty v registru PSWL nebývá obvykle důležité, nastavení PSWH je předurčené nastavením systémového módu (bit15 = 1) a prioritou programového kódu. Aby inicializační část programu nebyla přerušena nemaskovatelným přerušením NMI, je vhodné nastavit nejvyšší prioritu programového kódu bit11 = = bit10 = bit9 = bit8 = 1. Tím je zamezeno přijetí maskovaných přerušení i nemaskovatelného přerušení (NMI), dokud není dokončena počáteční procedura obsahující nastavení systémové architektury, vstupů a výstupů a inicializace zásobníků. Počáteční inicializace je zakončena prostým větvením nebo skokem do aplikačního programu. Instrukce RETI nesmí být použita, protože v zásobníku není uložena návratová adresa. Procesor však může být vynulován i programově pomocí instrukce RESET. Tato instrukce má stejný efekt jako výskyt vnějšího

nulovacího signálu, vyjma uložení stavu EA a programové šířky sběrnice BUSW. Ty procesory XA, které jsou vybaveny čítačem watchdog, mohou být ještě obvodově vynulovány při jeho přetečení, když program ztratí kontrolu nad touto jednotkou (dojde k disinterpretaci programu).

5.9. Módy se sníženou spotřebou

Procesor XA podporuje standardní módy se sníženou spotřebou tj. Idle a Power-down mód s možností nastavení v registru PCON. Novinkou je, jako u většiny moderních procesorů, možnost opustit Power-down mód kromě signálu RST i úrovní log. 0 přivedenou na vstup povoleného vnějšího přerušení. Tato možnost výrazně zpříjemňuje použití Power-down módu. Procesor však nezačne na přerušení reagovat okamžitě, protože ke své činnosti musí nejprve spustit a nechat ustálit hodinový oscilátor. Aby aktivující vnější přerušení nemuselo v úrovni log. 0 setrvávat celých 10 ms, vytváří si procesor tuto dobu pomocí vlastního čítače. Čítač počítá 9892 hodinových impulzů oscilátoru před spuštěním vlastního programu. Po přechodu procesoru do obou módů zůstává stav výstupních bran nezměněn. Z konstrukčního hlediska je důležité, že oproti procesorům 8051 zůstávají signály ALE a PSEN v neaktivním stavu (log. 1). V Power-down módu může být napájecí napětí sníženo až na 2 V k udržení obsahu paměti RAM i registrů SFR.

5.10. Vstupně/výstupní brány

Každý vývod V/V brány na procesoru XA-G3 může být uživatelsky konfigurován do čtyřech variant: kvaziobousměrný vývod shodný se standardním vývodem brány 80C51, výstup s otevřeným kolektorem (obdoba brány P0 firmy Atmel), dvoustavový Push-Pull a stav výstupu s vysokou impedancí. Po vynulování procesoru je nastavena klasická kvaziobousměrná konfigurace. Jedná-li se o procesor bez vnitřní paměti ROM (EA = 0), budou vývody bran realizujících datovou sběrnici (P0 a P2) konfigurovány na typ Push-Pull.

PnCFGB	PnCFGA	Výstupní mód
0	0	otevřený kolektor
0	1	kvaziobousměrný
1	0	vysoká impedance
1	1	Push-Pull

Tabulka 11

Konfiguraci V/V bran zajišťují vždy dva speciální funkční registry pro každou bránu, označené PnCFGA a PnCFGB, kde n určuje číslo příslušné brány. Každý vývod má na stejné bitové pozici v obou registrech rezervovaný bit určující jednu ze čtyř výstupních konfigurací *tab. 11*. Zápis do registrů může způsobit zákmity při přechodech, proto modifikaci registrů provádějte současně.

5.11. Vnější sběrnice

Většina procesorů řady XA bude mít možnost vytvořit vnější sběrnici pro připojení vnější datové a programové paměti. Procesor podporuje 8 nebo 16bitový vnější datový přenos až s 24bitovým přímo adresovatelným paměťovým prostorem. Kromě standardních řídících signálů ALE, $\overline{\text{PSEN}}$, $\overline{\text{RD}}$ a $\overline{\text{WR}}$, je procesor vybaven ještě signály $\overline{\text{WRH}}$ a $\overline{\text{WAIT}}$. Signály ALE, $\overline{\text{PSEN}}$ a $\overline{\text{RD}}$ mají stejný význam, který byl již popsán u procesoru 8051. Signál $\overline{\text{WR}}$ má stejný význam, ale při 16bitových zápisech řídí pouze zápis do spodních 8 bitů. Logicky s tím řídí signál $\overline{\text{WRH}}$ zápis horních 8 bitů do vnější datové paměti při 16bitovém zápisu. Pro připojení pomalých externích periferií je procesor navíc vybaven vstupním signálem WAIT, který určuje zpomalení přenosů po vnější sběrnici.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
---	---	---	WAITD	BUSD	BC2	BC1	BC0	Adresa SFR = 46AH

Obr. 63 Uspořádání bitů v registru BCR

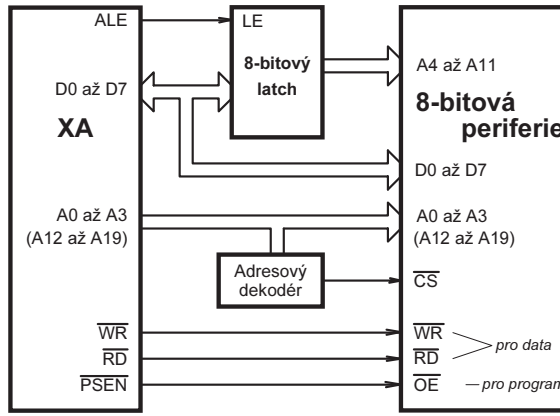
Jak již bylo naznačeno, má procesor XA konfigurovatelnou šířku vnější datové sběrnice na 8 nebo 16 bitů. Je-li program procesoru po spuštění nejprve čten z vnitřní programové paměti (EA = 1), šířka vnější sběrnice může být nastavena programově pomocí bitů BC2, BC1 a BC0 (8 bitů $\Rightarrow$ BC2 = 0 a BC1 = 1, 16 bitů $\Rightarrow$ BC2 = 1 a BC1 = 1) v registru BCR. Pracuje-li procesor pouze s vnější pamětí (EA = 0), potom šířku sběrnice musí znát dříve, než ji poprvé použije. K tomu slouží vývod BUSW/P3.5, který je testován při přechodu signálu $\overline{\text{RST}}$ (náběžná hrana) do neaktivního stavu. Není-li vývod zapojen, potom vnitřní odpor nastaví na vývodu úroveň log. 1 a datová sběrnice se stává 16 bitovou. Připojíme-li vývod BUSW přes odpor menší než 2k7 na zem, datová sběrnice se stává 8bitovou. Jak signál EA, tak signál BUSW musí tři hodinové cykly po resetu zůstat v definované úrovni, aby jejich hodnoty byly spolehlivě zachyceny do vnitřních registrů. Stejně jako je konfigurovatelná šířka datové sběrnice, je konfigurovatelná i šířka adresové sběrnice tak, aby zbývající adresové vodiče mohly být využity pro jiné účely.

Význam jednotlivých bitů v registru BCR *obr. 63* je následující:

- **WAITD** - Zákaz signálu WAIT. Je-li tento bit nastaven (log. 1), potom vnější datový přenos ignoruje hodnotu na vývodu WAIT.
- **BUSD** - Zákaz sběrnice. Nastavením bitu do log. 1 je trvale potlačena vnější sběrnice. Bit lze využít jako preventivní ochranu proti operacím předčítání instrukcí na konci adresového prostoru vnější programové paměti, které může způsobit nastavení nežádoucí hodnoty na adresovém vývodu, využívaném k jinému účelu.
- **BC2 až BC0** - Tyto tři bity rozhodují o konfiguraci šířky vnější programové a datové sběrnice. Význam bitů je zřejmý z *tabulky 12*.

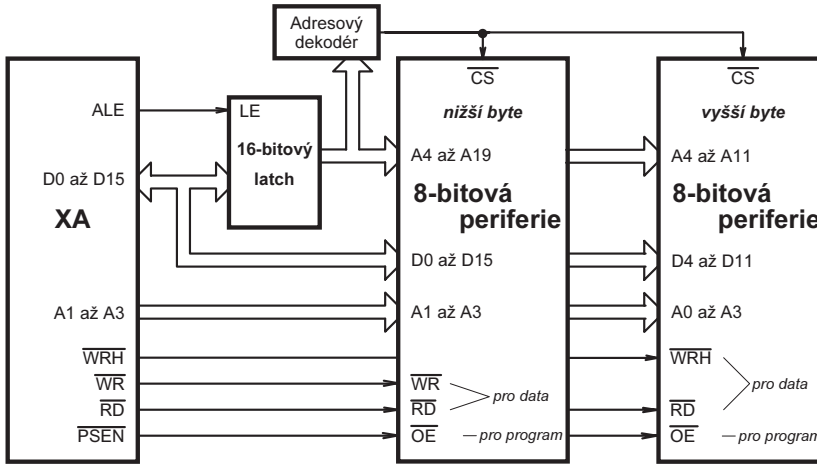
Bity			Šířka sběrnice		Bity			Šířka sběrnice	
BC2, BC1, BC0			datová	adresová	BC2, BC1, BC0			datová	adresová
0	0	0	8	12	0	0	1	8	16
0	1	0	8	20	0	1	1	8	24
1	0	0	Rezerva	Rezerva	1	0	1	Rezerva	Rezerva
1	1	0	16	20	1	1	1	16	24

Tabulka 12



Obr. 64 Typická 8bitová vnější sběrnice uP XA

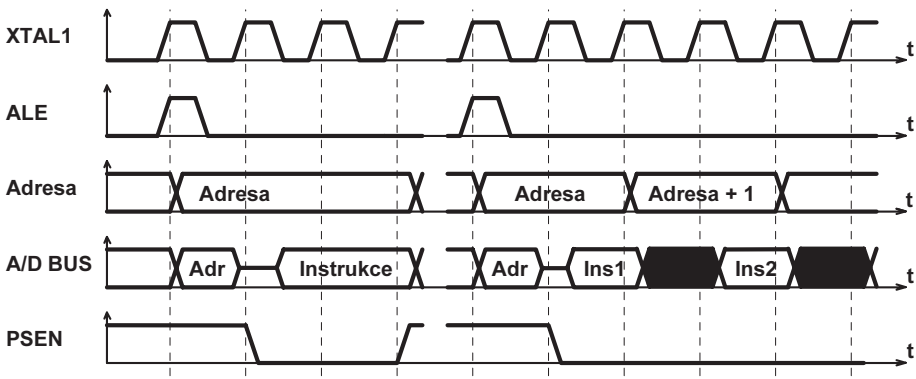
Na *obr. 64* je naznačeno typické 8bitové připojení vnější paměti nebo periferie, na *obr. 65* je zobrazeno 16bitové připojení vnější paměti nebo periferie. Připojení **programové paměti** je standardní, aktivační signál signál PSEN se připojuje na vývod OE paměti EPROM. Není-li aktivační signál CS aktivován z výstupu adresového dekodéru, může být propojen též s vývodem PSEN. Je-li program umístěn pouze ve vnější paměti, potom na vývodu BUSW při náběžné hraně signálu RESET musí být úroveň odpovídající konfiguraci sběrnice. Protože procesor provádí předčítání instrukcí do fronty, může v případě větvení na samém konci adresového prostoru paměti EPROM docházet ke čtení instrukcí z adres mimo paměť EPROM. Tím mohou změnit svoji hodnotu vyšší adresové vodiče, využívané v aplikaci k jinému účelu. Aby bylo možné této situaci zabránit, je procesor vybaven bitem BUSD v registru BCR. Jakmile procesor zjistí, že bude realizováno větvení (skok, volání podprogramu), zastaví stávající předčítání instrukcí, frontu smaže a začne předčítat nové instrukce. Procesor umožňuje zrychlení čtení instrukcí (tzv. Burst Code Read), který nevyžaduje cyklus ALE. Zrychlení je umožněno tím, že do adresového registru jsou ukládány adresy A4 až A11 nebo A4 až A19 *obr. 65*



Obr. 65 Typická 8bitová vnější sběrnice uP XA

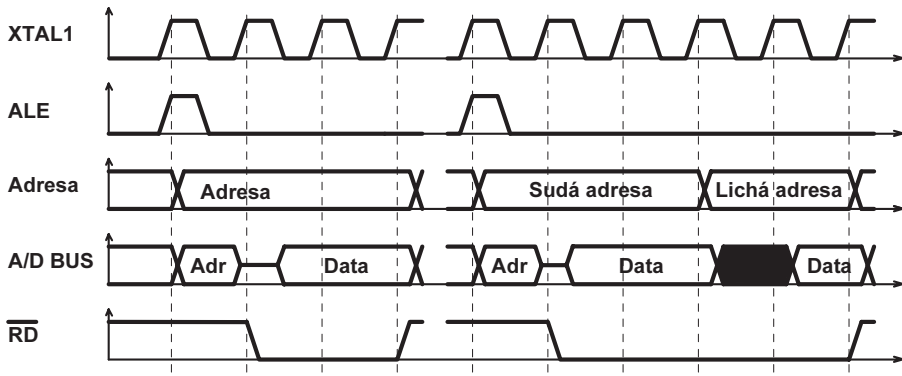
a adresy A0 až A3 jsou generovány přímo procesorem. Díky tomu přístup až k 16 bytům nebo 8 slovům nevyžaduje nový zápis vyšších adres a tím i cyklus se signálem ALE. Protože čtení nebo zápis do datové paměti se provádí vždy s cyklem ALE, bude po této operaci vždy následovat instrukce s tímto cyklem. Na *obr. 66* je zobrazen typický cyklus čtení z vnější programové paměti a cyklus Burst módu.

Čtení a zápis do datové paměti je řízen signály $\overline{RD}$, $\overline{WR}$ a $\overline{WRH}$, které v kombinaci s 8 nebo 16bitovým přístupem umožňují několik základních cyklů. Typické čtení jednoho bytu nebo slova začíná na sběrnici cyklem ALE, následovaného



Obr. 66 Typické časování vnější programové paměti a časování v Burst módu pro parametry ALEW = 0; CRA1,0 = 01; CR1,0 = 01

aktivací signálu $\overline{RD}$ (levá polovina *obr. 67*). Je-li čteno slovo (16 bitů) na 8bitové sběrnici, časové průběhy signálů budou podle pravé poloviny *obr. 67*. V případě čtení bytu na 16bitové sběrnici se vykoná jeden cyklus čtení, ze kterého se využije požadovaný byte. Zápis hodnot do paměti začíná vždy cyklem ALE, který je následován jedním nebo oběma zápisovými impulzy $\overline{WR}$ a $\overline{WRH}$. Zápis 8bitové hodnoty na 16bitové sběrnici je provázen signálem $\overline{WR}$ nebo $\overline{WRH}$ podle toho, zda adresa je sudá nebo lichá. Při zápisu slova na 8bitové sběrnici je nejprve zapsán nižší byte na sudou adresu a potom vyšší byte na adresu lichou.



Obr. 67 Typické časování vnější datové paměti a časování v Burst módu pro parametry ALEW = 0; DRA1,0 = 01; DR1,0 = 01

5.11.1. Časování vnější sběrnice

Procesor umožňuje separátní programové nastavení šířky impulzu signálu ALE, signálu $\overline{PSEN}$, signálů $\overline{RD}$, $\overline{WR}$ a $\overline{WRH}$. Tyto časy se dají naprogramovat v takovém rozsahu, že umožňují připojit většinu pamětí RAM, EPROM a periferních obvodů pro široký rozsah kmitočtů oscilátoru. Díky tomu většinou není třeba vnějších registrů a generátorů WAIT stavů. Pro odstranění problémů s delší deaktivační dobou obvodů umožňuje procesor vkládat za operaci se sběrnicí jeden prázdný hodinový cyklus. Časování signálů na sběrnici zajišťuje nastavení registrů BTRH a BTRL. Na *obr. 68* a na *obr. 69* je zobrazeno umístění jednotlivých bitů v těchto registrech.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
DW1	DW0	DWA1	DWA0	DR1	DR0	DRA1	DRA0	Adresa RAM = 469H

Obr. 68 Rozložení bitů v registru BTRH

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Bit
WM1	WM0	ALEW	---	CR1	CR0	CRA1	CRA0	Adresa RAM = 468H

Obr. 69 Rozložení bitů v registru BTRL

Význam jednotlivých bitů a jejich nastavení je následující:

♣ **DW1 a DW0** - Zápis druhé poloviny 16bitových dat na 8bitové sběrnici. Dobu trvání cyklu zápisu lze naprogramovat na 2 hodinové cykly (DW1 = DW0 = 0), na 3 hodinové cykly (DW1 = 0, DW0 = 1) a až na 5 hodinových cyklů (DW1 = DW0 = 1).

♣ **DWA1 a DWA0** - Cyklus zápisu včetně signálu ALE. Dobu trvání cyklu zápisu lze naprogramovat na 2 hodinové cykly (DWA1 = DWA0 = 0), na 3 hodinové cykly (DWA1 = 0, DWA0 = 1) a až na 5 hodinových cyklů (DWA1 = DWA0 = 1).

♣ **DR1 a DR0** - Čtení druhé poloviny 16bitových dat na 8bitové sběrnici. Dobu trvání cyklu čtení lze naprogramovat na 1 hodinový cyklus (DR1 = DR0 = 0), na 2 hodinové cykly (DR1 = 0, DR0 = 1) a až na 4 hodinových cyklů (DR1 = DR0 = 1).

♣ **DRA1 a DRA0** - Cyklus čtení včetně signálu ALE. Dobu trvání cyklu čtení lze naprogramovat na 2 hodinové cykly (DRA1 = DRA0 = 0), na 3 hodinové cykly (DRA1 = 0, DRA0 = 1) a až na 5 hodinových cyklů (DRA1 = DRA0 = 1).

♣ **CR1 a CR0** - Cyklus čtení instrukce bez ALE. Dobu trvání cyklu čtení instrukce lze naprogramovat na 1 hodinový cyklus (CR1 = CR0 = 0), na 2 hodinové cykly (CR1 = 0, CR0 = 1) a až na 4 hodinové cykly (CR1 = CR0 = 1).

♣ **CRA1 a CRA0** - Cyklus čtení instrukce včetně signálu ALE. Dobu trvání cyklu čtení lze naprogramovat na 2 hodinové cykly (CRA1 = CRA0 = 0), na 3 hodinové cykly (CRA1 = 0, CRA0 = 1) a až na 5 hodinových cyklů (CRA1 = CRA0 = 1).

♣ **ALEW** - Šířka impulzu signálu ALE. Pro hodnotu 0 je šířka ALE rovna polovině hodinového cyklu, pro hodnotu 1 je rovna 1,5 hodinového cyklu.

♣ **WM1** - Šířka zápisového impulzu WR. Šířka WR je pro hodnotu 0 rovna jednomu hodinovému cyklu a pro hodnotu 1 dvěma hodinovým cyklům.

♣ **WM0** - Přesah dat za zápisovým impulzem. Pro hodnotu 0 je čas přesahu roven prakticky nule a pro hodnotu 1 je roven jednomu hodinovému cyklu.

Některá (nesprávná) nastavení časových registrů způsobí u procesoru negenerování některých řídících signálů. Nepovolenou kombinací je ta, u které součet definovaných částí přesahuje definovanou délku celého cyklu. Taková kombinace např. způsobí generování adresy, signálu ALE, ale není generován signál RD. Při zápisu musí být splněny dvě podmínky:

$WM1 + WM0 \leq DW1:0$ pro cyklus bez ALE

$ALEW + WM1 + WM0 \leq DWA1:0$ pro cyklus s ALE

5.12. Instrukční soubor

Instrukční soubor je nadstavbou instrukčního souboru 80C51 a je dostatečně přizpůsobivý k podpoře mnoha řídicích aplikací. Soubor umožňuje operace typu registr-registr, registr - přímo a nepřímo adresované paměťové místo, registr - přímá data a přímo nebo nepřímo adresované paměťové místo - přímá data. Obsahuje instrukce nepodmíněného skoku a volání přes celý adresový prostor nebo relativní v rámci dané stránky (16 bitů). Instrukce podmíněného větvení a volání podporují pouze krátké (8bitové) relativní skoky. Na jinak velmi rozsáhlém instrukčním souboru je příjemné, že využívá stejných mnemotechnických zkratek instrukcí jako je tomu u procesoru 80C51. Protože uvedené operace mohou používat 8, 16 nebo 32bitové operandy, je třeba v některých případech odlišit instrukce pracující s byty od instrukcí pracující se slovy nebo dvěma slovy. Například u instrukce MOV R2,#1 není jasné zda bude vykonána 16bitově nebo 8bitově. Proto jsou zavedena doplnění, která tento rozdíl jednoznačně řeší. Instrukce MOV.b [R2], #1 - pracuje s bytem, instrukce MOV.w [R2], #1pracuje se slovem. Jenom instrukce DIV.d umožňuje pracovat s 32bitovou hodnotou. Pro již zmíněnou rozsáhlost instrukčního souboru bude uveden pouze stručný komprimovaný přehled instrukcí procesoru XA.

Aritmetické instrukce

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
	Rd, Rs	(Rd)←(Rd) + (Rs)	2	3
ADD	**Rd, [Rs]**	(Rd)←(Rd) + ((Rs))	2	4
	[Rd], Rs	((Rd))←((Rd)) + (Rs)	2	4
	Rd, [Rs+offset8]	(Rd)←(Rd) + ((Rs)+offset8)	3	6
	[Rd+offset8], Rs	((Rd)+offset8)←(Rs) + ((Rd)+offset8)	3	6
ADDC	**Rd, [Rs+offset16]**	(Rd)←(Rd) + ((Rs)+offset16)	4	6
	[Rd+offset16], Rs	((Rd)+offset16)←(Rs) + ((Rd)+offset16)	4	6
	Rd, [Rs+]	(Rd)←(Rd) + ((Rs)) , (Rs)←(Rs)+1÷2	2	5
	[Rd+], Rs	((Rd))←((Rd)) + (Rs) , (Rd)←(Rd)+1÷2	2	5
CMP	**direct, Rs**	(direct)←(direct) + (Rs)	3	4
	Rd, direct	(Rd)←(Rd) + (direct)	3	4
	Rd, #data8	(Rd)←(Rd) + data8	3	3
	Rd, #data16	(Rd)←(Rd) + data16	4	3
SUB	**[Rd], #data8**	((Rd))←((Rd)) + data8	3	4
	[Rd], #data16	((Rd))←((Rd)) + data16	4	4
	[Rd+], #data8	((Rd))←((Rd)) + data8 , (Rd)←(Rd) + 1	3	5
	[Rd+], #data16	((Rd))←((d)) + data16 , (Rd)←(Rd) + 2	4	5
SUBB	**[Rd+offset8], #data8**	((Rd)+offset8)←((Rd)+offset8) + data8	4	6
	[Rd+offset8], #data16	((Rd)+offset8)←((Rd)+offset8) + data16	5	6
	[Rd+offset16], #data8	((Rd)+offset16)←((Rd)+offset16) + data8	5	6
	[Rd+offset16], #data16	((Rd)+offset16)←((Rd)+offset16) + data16	6	6
	direct, #data8	(direct)←(direct) + data8	4	4
	direct, #data16	(direct)←(direct) + data16	5	4
ADDS	**Rd, #data4**	(Rd)←(Rd) + data4	2	3
ADDS	**[Rd], #data4**	((Rd))←((Rd)) + data4	2	4
ADDS	**[Rd+], #data4**	((Rd))←((Rd)) + data4 , (Rd)←(Rd) + 1	2	5

ADDS	**[Rd+offset8], #data4**	((Rd)+offset8)←((Rd)+offset8) + data4	3	6
ADDS	**[Rd+offset16], #data4**	((Rd)+offset16)←((Rd)+offset16) + data4	4	6
ADDS	**direct, #data4**	(direct)←(direct) + data4	3	4
ASL	**Rd, Rs**	(Rd)←(Rd) << (Rs)	2	
ASL	**Rd, #data4**	(Rd)←(Rd) << data4	2	
ASR	**Rd, Rs**	(Rd)←(Rd) >> (Rs)	2	
ASR	**Rd, #data4**	(Rd)←(Rd) >> data4	2	
DA	**Rd**	Dekadická korekce Rd	2	4
DIV.w	**Rd, Rs**	Dělení se znaménkem 16/8 bitů	2	14
DIV.w	**Rd, #data8**	Dělení se znaménkem 16bitů/data8	3	14
DIV.d	**Rd, Rs**	Dělení se znaménkem 32/16 bitů	2	24
DIV.d	**Rd, #data16**	Dělení se znaménkem 32bitů/data16	4	24
DIVU.b	**Rd, Rs**	Dělení bez znaménka 8/8 bitů	2	12
DIVU.b	**Rd, #data8**	Dělení bez znaménka 8 bitů/data8	3	12
DIVU.w	**Rd, Rs**	Dělení bez znaménka 16/8 bitů	2	12
DIVU.w	**Rd, #data8**	Dělení bez znaménka 16bitů/data8	3	12
DIVU.d	**Rd, Rs**	Dělení bez znaménka 32/16 bitů	2	22
DIVU.d	**Rd, #data16**	Dělení bez znaménka 32 bitů/data16	4	22
LEA	**Rd, Rs +offset8**	Čtení 16bitové efektivní adresy	3	3
LEA	**Rd, Rs +offset16**	Čtení 16bitové efektivní adresy	4	3

MUL.w	**Rd, Rs**	Násobení se znaménkem 16 * 16 bitů	2	12
MUL.w	**Rd, #data16**	Násobení se znaménkem 16bitů * data16	4	12
MULU.b	**Rd, Rs**	Násobení bez znaménka 8 * 8 bitů	2	12
MULU.b	**Rd, #data8**	Násobení bez znaménka 8bitů * data8	3	12
MULU.w	**Rd, Rs**	Násobení bez znaménka 16 * 16 bitů	2	12
MULU.w	**Rd, #data16**	Násobení bez znaménka 16bitů * data16	4	12
NEG	**Rd**	Dvojkový doplněk Rd	2	3
SEXT	**Rd**	Znaménkové rozšíření poslední operace	2	3

Logické operace

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
AND	**všechny jako pro ADD**	Logický součin	dtto	dtto
OR	**všechny jako pro ADD**	Logický součet	dtto	dtto
XOR	**všechny jako pro ADD**	Neekvivalence EX-OR	dtto	dtto
CPL	**Rd**	Jednotkový doplněk Rd	2	3
LSR	**Rd, Rs**	Logická rotace doprava o hodnotu Rs	2	
LSR	**Rd, #data4**	Logická rotace doprava o hodnotu data4	2	
NORM	**Rd, Rs**	Logická rotace doleva o hodnotu Rs dokud MSB Rd není rovno 1	2	
RL	**Rd, #data4**	Logická rotace doleva o hodnotu data4	2	
RLC	**Rd, #data4**	Logická rotace doleva přes C o hod.data4	2	
RR	**Rd, #data4**	Logická rotace doprava o hodnotu data4	2	
RRC	**Rd, #data4**	Logická rotace doprava přes C o hod.data4	2	

Přesunové instrukce

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
MOV	**všechny jako pro ADD**	Přesun	dtto	dtto
MOV	**[Rd+], [Rs+]**	((Rd))←((Rs)), (Rd)←(Rd) + 1÷2 , (Rs)←(Rs) + 1÷2	2	6
MOV	**direct, [Rs]**	(direct)←(Rs)	3	4
MOV	**[Rd], direct**	(Rd)←(direct)	3	4
MOV	**direct, direct**	(direct)←(direct)	4	4

MOV	**USP, Rs**	Obsah Rs do uživatel. ukazatele zásobníku	2	3
MOV	**Rd, USP**	Uživatel. ukazatel zásobníku do Rd	2	3
MOVC	**[Rd], [Rs+]**	Přesuv z prog.paměti WS:Rs do Rd plus inkrementace Rs	2	4
MOVC	**A, [A+DPTR]**	Přesun z prog. paměti adres. DPTR a A	2	6
MOVC	**A, [A+PC]**	Přesun z prog. paměti adres. PC a A	2	6
MOVS	**všechny jako pro ADDS**	Přesun data4 do reg. nebo paměti	dtto	dtto+1
MOVX	**Rd, [Rs]**	Přesun z externí paměti (Rd)←((Rs))	2	6
MOVX	**[Rd], Rs**	Přesun z externí paměti ((Rd))←(Rs)	2	6
PUSH **POP**	**direct**	Uložení/vyjmutí (byte/slovo) do/ze zásobníku	3	5
PUSHU **POPU**	**direct**	Uložení/vyjmutí (byte/slovo) do/ze uživatelského zásobníku	3	5
PUSH **POP**	**Rlist**	Násobné uložení/vyjmutí registrů (byte/slovo) do/ze zásobníku	2	
PUSHU **POPU**	**Rlist**	Násobné uložení/vyjmutí registrů (byte/slovo) do/ze uživatelského zásobníku	2	
XCH	**Rd, Rs**	Výměna obsahu (Rd)↔(Rs)	2	5
XCH	**Rd, [Rs]**	Výměna obsahu (Rd)↔((Rs))	2	6
XCH	**Rd, direct**	Výměna obsahu (Rd)↔přímo adr. místa	3	6

Bitové operace

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
ANL	**C, bit**	Log.součin, log.součet, přesun příznak*bit	3	4
ORL				
MOV	**bit, C**	Log.součin, log.součet, přesun bit*příznak	3	4
CLR	**bit**	Nulování bitu	3	4
SETB	**bit**	Nastavení bitu	3	4

Instrukce větvení programu

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
BCC	**rel8**	Větvení je-li příznak C=0	2	6p/3n
BCS	**rel8**	Větvení je-li příznak C=1	2	6p/3n
BEQ	**rel8**	Větvení je-li příznak Z=1	2	6p/3n
BNE	**rel8**	Větvení je-li příznak Z=0	2	6p/3n
BG	**rel8**	Větvení je-li větší (bez znam.)	2	6p/3n
BGE	**rel8**	Větvení je-li větší nebo rovno (bez znam.)	2	6p/3n
BGT	**rel8**	Větvení je-li větší (znam.)	2	6p/3n
BL	**rel8**	Větvení je-li menší nebo rovno (bez znam.)	2	6p/3n
BLE	**rel8**	Větvení je-li menší nebo rovno (znam.)	2	6p/3n
BLT	**rel8**	Větvení je-li menší než (znam.)	2	6p/3n
BMI	**rel8**	Větvení je-li příznak S=1	2	6p/3n
BPL	**rel8**	Větvení je-li příznak S=0	2	6p/3n
BNV	**rel8**	Větvení je-li příznak OV=0	2	6p/3n
BOV	**rel8**	Větvení je-li příznak OV=1	2	6p/3n
BR	**rel8**	Nepodmíněné větvení	2	3
CALL	**[Rs]**	Nepřímé volání uvnitř 64kB	2	8/5
CALL	**rel16**	Nepřímé volání ±64kB	2	7/4
CJNE	**Rd, direct,rel8**	Porovnání Rd s direct a skok při nerovnosti	4	10/7
CJNE	**Rd, #data8,rel8**	Porovnání Rd s data8 a skok při nerovnosti	4	9/6
CJNE	**Rd, #data16,rel8**	Porovnání Rd s data16 a skok při nerovnosti	5	9/6
CJNE	**[Rd], #data8,rel8**	Porovnání ((Rd)) s data8 a skok při nerov.	4	10/7

CJNE	**[Rd], #data16,rel8**	Porovnání ((Rd)) s data16 a skok při nerov.	5	10/7
DJNZ	**Rd, rel8**	Dekrementace Rd a skok při nenulovosti	3	8/5
DJNZ	**direct, rel8**	Dekrementace direct a skok při nenulovosti	4	9/6
FCALL	**adr 24**	Vzdálené volání přes až 16 MB	4	9/5
FJMP	**adr 24**	Vzdálený skok přes až 16 MB	4	6
JB	**bit, rel8**	Skok je-li bit nastaven	4	7/4
JBC	**bit, rel8**	Skok je-li bit nastaven, bit nuluj	4	7/4
JMP	**rel16**	Nepodmíněný dlouhý skok	3	6
JMP	**[Rs]**	Nepodmíněný nepřímý dlouhý skok 64 kB	2	7
JMP	**[A+DPTR]**	Nepodmíněný nepřímý dlouhý skok 64 kB	2	5
JMP	**[[Rs]]**	Dvojnásobně nepřímý nepod. skok 64 kB	2	8
JNB	**bit, rel8**	Skok je-li bit nulový	4	7/4
JNZ	**rel8**	Skok není-li akumlátor roven nule	2	7/4
JZ	**rel8**	Skok je-li akumlátor roven nule	2	7/4
NOP		Prázdná operace	1	3
RET		Návrat z podprogramu	2	8/6
RETI		Návrat z přerušení	2	10/8

Instrukce přerušení

Instrukce	Operandy Cílový, Zdrojový	POPIS INSTRUKCE	Byty	Cykly
BKPT		Volání breakpoint přerušení	1	23/19
RESET		Nulování procesoru	2	8
TRAP	**#data4**	Volání přerušení 1 až 16	2	23/19

Závěrem lze říci, že procesory XA představují moderní, velmi přizpůsobivé a výkonné procesory. S jejich použitím lze počítat jak v jednoduchých aplikacích, tak i ve velmi rozsáhlých a složitých systémech. Svými vlastnostmi a složitostí se však od našeho procesoru 8051 již dost vzdaluje.

6. Co se obvykle nepublikuje

Existuje řada vývojových i řídicích systémů u kterých se za provozu rekonfiguruje uspořádání datových a programových pamětí. Rád bych na závěr této knížky nechal čtenáře nahlédnout do zákulisí takovýchto systémů a popsal zásady, které je třeba při těchto operacích dodržovat. Jak název kapitoly naznačuje nejedná se o informace, které by výrobci a konstruktéři ochotně sdělovali.

Prvním problémem, s kterým se můžeme setkat je připojení větší programové nebo datové paměti k mikroprocesoru, než je jeho přímo adresovatelný prostor. Pokud není obvodově sloučen datový a programový prostor procesoru 8051 je zvětšování jeho datové paměti jednoduché a realizuje se tzv. stránkováním paměti za pomoci dalších adresových vodičů A16, A17 atd. vytvořeným na výstupní bráně procesoru. Daleko zajímavější je připojení programové paměti větší než 64 kB. Potřeba vytvoření vyšších adres na výstupní bráně zůstává, ale problematičtější je přepínání pamětí a obsluha přerušovacích rutin. Přepínání mezi pamětmi bude určeno změnou vyšších adresovacích vodičů tj. A16, A17 atd., ale jejich změna nemůže být libovolná. Je třeba mít na paměti, že po provedení instrukce zápisu do brány vytvářející vyšší adresové vodiče, bude následující instrukce čtena z již nové stránky programové paměti. To znamená, že na této nové stránce na adrese následující po instrukci zápisu musí ležet smysluplná instrukce požadovaného programu. U procesorů s překrýváním instrukcí (pipeline), je rozhodujícím okamžikem výkonná fáze instrukce měnící adresové vodiče. V průtokové struktuře však ještě zůstávají načtené a rozpracované instrukce, které budou vykonány a teprve po nich budou zařazeny instrukce z nové požadované stránky. Obdobná, ale hůře vypočitatelná situace nastává u procesorů s připravenou frontou instrukcí.

Požadované stránkování se obvykle realizuje v určité definované části adresového prostoru, která je společná pro všechny programové stránky. Společný prostor může být vytvořen obvodovými prostředky v jedné obvykle první stránce programové paměti nebo programově pomocí stejných obsahů v určitém adresovém prostoru programových stránek. Vytvoříme-li společný prostor obvodovými prostředky například v horních 8 kB adresového prostoru (adresy E000h až FFFFh), potom aktivační signály pro jednotlivé 64kB paměti EPROM typu 27C512 budou dány těmito vztahy:

$$\overline{CS1} = \overline{(A15 * A14 * A13)} * (A16 + A17)$$

$$\overline{CS2} = A15 * A14 * A13 + \overline{A16} + A17$$

$$\overline{CS3} = A15 * A14 * A13 + A16 + \overline{A17}$$

$$\overline{CS4} = A15 * A14 * A13 + \overline{A16} + A17$$

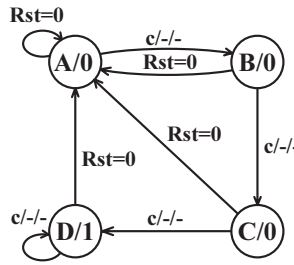
kde v paměti aktivované CS1 bude adresový prostor 0000h÷DFFFh představovat první stránku programové paměti a adresový prostor E000h÷FFFFh tzv. spo-

lečný prostor. U pamětí aktivovaných CS2, CS3 a CS4 budou adresové prostory 0000h÷DFFFh představovat druhou, třetí a čtvrtou stránku programové paměti, adresové prostory E000h÷FFFFh nebudou využívány (nebudou nikdy aktivovány). Ve společném prostoru pak bude umístěn tzv. ***přepínací program***, který podle vstupních parametrů zajistí přechod na definovanou adresu v definované stránce programové paměti, kde bude vykonávána požadovaná část programu. V dnešní době se situace komplikuje díky velkému množství vnějších i vnitřních přerušení procesoru. Obslužné programy přerušení musí být umístěny ve společné oblasti proto, aby mohly být vykonány bez ohledu na to, v které programové stránce se právě vykonává program. Většina velkých programů je realizována v jazyce C, což situaci příliš neusnadňuje. Z tohoto důvodu se daleko častěji přistupuje k programové tvorbě společné oblasti tím, že ve všech programových stránkách v určité oblasti je uložen stejný obsah. Výhodou tohoto řešení je, že velikost společné oblasti nemusí být přesně obvodově vymezena. Její velikost bude dána rozsahem všech přerušení, všech obecně přístupných podprogramů, programových konstant a řetězců, inicializační a přepínací části programu. Tuto strukturu přepínání programových stránek podporují nejnovější verze jazyka C V5.0 a vyšší od firem Keil Electronic a Archimedes pro procesory 80C51. Tyto překladače jsou navíc vybaveny programem BL51, který linkuje programu podle výše uvedeného stránkování programové paměti přímo pro paměti EPROM.

Druhým problémem, který se vyskytuje u vývojových systémů a je podobný tomu předcházejícímu, je rekonfigurace programové paměti. Řada vývojových systémů vyžaduje nejen společný programový a datový prostor, který získáme logickým součinem signálů $\overline{PSEN}$, $\overline{RD}$ a $\overline{WR}$ pro aktivaci (CS) paměti RAM a signálů $\overline{PSEN}$ a $\overline{RD}$ pro aktivaci (OE) jejího třístavového budiče, ale i jeho umístění od adresy 0000h. V okamžiku po vynulování procesoru to však není možné, protože z nulové adresy a adres následujících budou čteny počáteční instrukce programu vývojového systému. Ty musí být čteny z paměti typu ROM (obvykle EPROM) a nikoliv z nezálohované paměti RAM. Tím stojíme před problémem rekonfigurace programové paměti. Paměť EPROM musí být po vynulování systému mapována od adresy 0000h a teprve později po jedné nebo více instrukcích přemapována do jiného adresového prostoru. Vlastní přepnutí může být řízeno pouze obvodově nebo programově a obvodově. Z obou řešení, která si ukážeme pro případ procesoru 80C51, se nejprve zmíníme o klasickém obvodovém řešení. V tomto případě se obvykle přečte z paměti EPROM instrukce dlouhého skoku (3 byty) na adresu v horní části adresového prostoru. Po přečtení této instrukce tj. po třech impulzech $\overline{PSEN}$ je automaticky obvodově paměť EPROM přemapována do horního adresového prostoru a to tak, aby na adrese na kterou byl realizován skok následoval program vývojového systému. Paměť EPROM obsahuje na prvních třech bytech instrukci skoku následovanou úplnou 16bitovou adresou, na dalších bytech se nachází program vývojového systému adresovaný pro paměť umístěnou v horní části adresového prostoru. Systém potom musí být

doplněn sekvenčním obvodem, který spočítá tři čtení z paměti EPROM (signály $\overline{PSEN}$) a potom změní vlastnosti adresového dekodéru a paměť přemapuje. Jako příklad si ukážeme přemapování 8 kB paměti EPROM z adresového prostoru 0000h÷1FFFh do prostoru E000h÷FFFFh. Vlastní přemapování zajistí adresový dekodér, který bude realizovat tuto logickou rovnici

$$\overline{CS} = (A15 + A14 + A13 + RID) * (\overline{RID * A15 * A14 * A13})$$



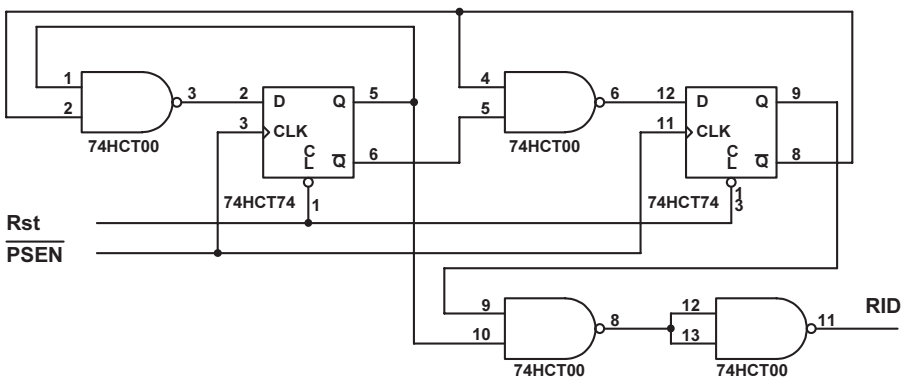
Obr. 70

Přepínací signál RID je výstupem sekvenčního obvodu, který bude ovládán signály nulování Rst a $\overline{PSEN}$ podle stavového diagramu z *obr. 70*. Zakódováním stavů A = 00, B = 01, C = 10 a D = 11 získáme tyto rovnice přechodů

$$Q_1^{i+1} = \overline{Q_1^i} + Q_2^i \qquad Q_2^{i+1} = Q_1^i + Q_2^i$$

$$RID = Q_1^i * Q_2^i$$

Na *obr. 71* je potom uvedena výsledná realizace navrženého obvodu z klasických obvodů.



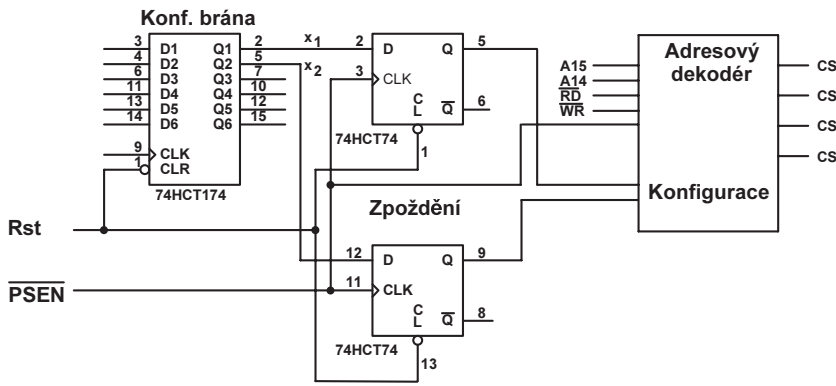
Obr. 71 Zapojení řídicího obvodu pro rekonfiguraci programové paměti

Druhé řešení téhož problému si ukážeme na obvodu, který využívá autor knihy ve vývojových systémech MIK552 a MIK537. Při tomto řešení se využívá programového řešení ve spolupráci s obvodovými prostředky. Hlavní výhoda řešení spočívá v možnosti opakované rekonfigurace pamětí celého systému s možností skoku na libovolnou adresu. Na výstupní bráně procesoru, která je mapována v oblasti vnější datové paměti (je ovládána instrukcí MOVX), je generován jeden nebo více konfiguračních signálů x_1, x_2 atd. Aby se vliv změny konfiguračních signálů neuplatnil ihned po instrukci zápisu prochází přes sekvenční obvod *obr. 72*, který je zpozdí o jednu instrukci tj. o dobu do příští náběžné hrany signálu $\overline{PSEN}$.

Díky tomuto obvodu se změna konfiguračních signálů uplatní až po provedení další instrukce (ve skutečnosti se projeví již při čtení 2 bytu, který však instrukce nevyužívá). Rekonfigurační program se potom skládá z následující posloupnosti instrukcí:

```
MOV A, # LOW  AdrPokr     ;Spodní část pokračovací adresy
PUSH ACC                  ;Ulož ji do zásobníku
MOV A, # HIGH  AdrPokr    ;Horní část pokračovací adresy
PUSH ACC                  ;Pokračovací adresa v zásobníku
MOV DPTR, # Adrkonfig     ;Do DPTR adresa konfigurační brány
MOV A, # Konfig_slovo     ;Konfigurační slovo = nové konfig.signály
MOV @DPTR, A              ;Změna konfiguračních signálů
RET                       ;Skok na pokračovací adresu, po instrukci
                          ;se konfigurační slovo uplatní v dekodéru
                          ;adres
```

Je-li použit jako adresový dekodér programovatelný obvod GAL, potom lze snadno realizovat i složité rekonfigurace. Budeme-li požadovat paměť EPROM s kapacitou 8 kB adresovanou v prostoru 0000h÷1FFFh pro x_1 = 0, E000h÷FFFFh pro x_1 = 1 a 32 kB paměť RAM konfigurovatelnou jako datovou paměť v adreso-



Obr. 72 Obvodové řešení rekonfiguračního obvodu

vém prostoru 8000h÷FFFFh pro $x_1 = 0$ a jako programovou i datovou paměť v prostoru 0000h÷7FFFh pro $x_1 = 1$, potom bude dekodér realizovat tyto rovnice:

$$\overline{CS_{EPROM}} = (x_1 + A15 + A14 + A13) * (\overline{x_1} + \overline{A15} + \overline{A14} + \overline{A13})$$
$$\overline{OE_{EPROM}} = \overline{PSEN}$$
$$\overline{CS_{RAM}} = (x_1 + \overline{A15} + \overline{RD} + \overline{WR}) * (\overline{x_1} + A15 + \overline{RD} * \overline{PSEN} * \overline{WR})$$
$$\overline{OE_{RAM}} = (x_1 + \overline{RD} + \overline{A15}) * (\overline{x_1} + A15 + \overline{RD} * \overline{PSEN})$$
$$\overline{WR_{RAM}} = \overline{WR}$$

Uvedený příklad není jediným programově-obvodovým řešením rekonfigurace pamětí EPROM a RAM v systémech 80C51, ale patří k těm nejvýkonnějším a nejjednodušším.

Ve vývojových systémech umožňujících krokování programu je třeba v místě zastavení (break point) přepsat původní program skokem do tzv. teplého místa monitoru, které umožní přečtení všech příznaků, registrů atd. Pokud se nehodí přepisovat všechny tři byty programu k realizaci dlouhého skoku, potom se využívá takováto „finta“. Přepisují se pouze dva byty instrukcí skoku a horní částí adresy vstupu do monitoru, jako spodní část skoku se využívá stávající byte testovaného programu. Vstup do monitoru pak začíná na adrese horního bytu doplněného nulami následovanou 256bytovou oblastí vyplněnou nulami (instrukcí NOP), do které se instrukce skoku s náhodným obsahem spodní části adresy vždy „trefí“.

Na závěr bych věnoval několik řádek zabezpečení vytvořeného programu. V mnoha aplikacích je v programu uchováno mnoho hodin vývoje a testování, a proto je více než žádoucí jeho obsah uchránit proti zcizení a kopírování. Je-li program jednoduchý řádově do 1 kB, potom lze jeho strojový kód pomocí zpětného překladače přeložit do jazyka symbolických adres a pokusit se jej analyzovat a pochopit jeho algoritmus. Je-li program mnohem větší, potom je tento způsob analýzy skoro nemožný. Používáme-li procesory s vnitřní pamětí FLASH, je program uzavřený v procesoru dostatečně ochráněn proti přečtení. Složitější situace je v zařízeních, kde program je natolik velký, že musí být uložen v pamětech EPROM. Tyto paměti nelze ochránit proti přečtení a proto někteří výrobci se snaží jeho obsah uchránit jednodušším nebo složitějším kódováním proti případným chytrákům kopírujícím výrobky. Kódování se obvykle provádí pomocí programovatelných obvodů, jejichž obsah lze chránit proti přečtení. Ochrana spočívá v přeházení datových nebo adresových vodičů, v lineární operaci (EX-OR, EX-NOR) mezi adresovými a datovými vodiči. Možné je dekódování části nebo celého zakódovaného programu v paměti EPROM do paměti RAM následované rekonfigurací celého systému. Většina těchto ochran však ochrání program pouze proti amatérskému kopírování. V současnosti lze proto jako nejspolehlivější ochranu považovat uložení a uzamčení alespoň části programu do vnitřní paměti FLASH moderního procesoru.

7. Dodatky

7.1. Instrukční soubor 8051 - přehled

Aritmetické instrukce

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
ADD A, Rr	$(A) \leftarrow (A) + (Rr)$			C,AC,OV,P
ADD A, @Rr	$(A) \leftarrow (A) + ((Rr))$	1	1	C,AC,OV,P
ADD A, adr	$(A) \leftarrow (A) + (adr)$			C,AC,OV,P
ADD A, #data	$(A) \leftarrow (A) + data$	2	1	C,AC,OV,P
ADDC A, Rr	$(A) \leftarrow (A) + (Rr) + (C)$	1	1	C,AC,OV,P
ADDC A, @Rr	$(A) \leftarrow (A) + ((Rr)) + (C)$	1	1	C,AC,OV,P
ADDC A, adr	$(A) \leftarrow (A) + (adresa) + (C)$	2	1	C,AC,OV,P
ADDC A, #data	$(A) \leftarrow (A) + data + (C)$	2	1	C,AC,OV,P
CLR A	$(A) \leftarrow 0$	1	1	P
CPL A	$(A) \leftarrow \overline{(A)} = {}^{1}(A)$	1	1	
DA A	*Dekadická korekce střadače*	1	1	C,AC,P
DEC A	$(A) \leftarrow (A) - 1$	1	1	P
DEC Rr	$(Rr) \leftarrow (Rr) - 1$	1	1	
DEC adr	$(adr) \leftarrow (adr) - 1$	2	1	
DEC @Rr	$((Rr)) \leftarrow ((Rr)) - 1$	2	1	
DIV AB	*(A) ← celá část (A) / (B)* *(B) ← zbytek (A) / (B)*	1	4	C=0,OV,P
INC A	$(A) \leftarrow (A) + 1$	1	1	P
INC Rr	$(Rr) \leftarrow (Rr) + 1$,	1	1	
INC adr	$(adr) \leftarrow (adr) + 1$	2	1	
INC @Rr	$((Rr)) \leftarrow ((Rr)) + 1$	2	1	
INC DPTR	$(DPTR) \leftarrow (DPTR) + 1$	1	2	
MUL AB	*(A)←nižší byte (A)*(B),* *(B)←vyšší byte (A)*(B)*	1	4	C=0,OV,P
NOP	*Prázdná operace*	1	1	
SUBB A, Rr	$(A) \leftarrow (A) - (Rr) - (C)$	1	1	C,AC,OV,P
SUBB A, @Rr	$(A) \leftarrow (A) - ((Rr)) - (C)$	1	1	C,AC,OV,P
SUBB A, adr	$(A) \leftarrow (A) - (adr) - (C)$	2	1	C,AC,OV,P
SUBB A, #data	$(A) \leftarrow (A) - data - (C)$	2	1	C,AC,OV,P

Logické operace

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
ANL A, Rr	$(A) \leftarrow (A)$ **AND** (Rr)			P
ANL A, @Rr	$(A) \leftarrow (A)$ **AND** $((Rr))$	1	1	P

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
ANL A, #data	$(A) \leftarrow (A)$ **AND** *data*			P
ANL adr, A	$(adr) \leftarrow (adr)$ **AND** (A)	2	1	P
ANL adr, #data	$(adr) \leftarrow (adr)$ **AND** *data*	3	2	
ORL A, Rr	$(A) \leftarrow (A)$ **OR** (Rr)	1	1	P
ORL A, @Rr	$(A) \leftarrow (A)$ **OR** $((Rr))$	1	1	P
ORL A, adr	$(A) \leftarrow (A)$ **OR** (adr)	1	1	P
ORL A, #data	$(A) \leftarrow (A)$ **OR** *data*	1	1	P
ORL adr, A	$(adr) \leftarrow (adr)$ **OR** (A)	2	1	
ORL adr, #data	$(adr) \leftarrow (adr)$ **OR** *data*	3	2	
RL A	$(A(n+1)) \leftarrow (A(n))$, n = 0 ÷ 6 $(A(0)) \leftarrow (A(7))$	1	1	
RLC A	$(A(n+1)) \leftarrow (A(n))$, n = 0 ÷ 6 $(A(0)) \leftarrow (C)$, $(C) \leftarrow (A(7))$	1	1	C,P
RR A	$(A(n)) \leftarrow (A(n+1))$, n = 0 ÷ 6 $(A(7)) \leftarrow (A(0))$	1	1	
RRC A	$(A(n)) \leftarrow (A(n+1))$, n = 0 ÷ 6 $(A(7)) \leftarrow (C)$, $(C) \leftarrow (A(0))$	1	1	C,P
XRL A, Rr	$(A) \leftarrow (A) \oplus (Rr)$	1	1	P
XRL A, @Rr	$(A) \leftarrow (A) \oplus ((Rr))$	1	1	P
XRL A, adr	$(A) \leftarrow (A) \oplus (adr)$	2	1	P
XRL A, #data	$(A) \leftarrow (A) \oplus data$	2	1	P
XRL adr, A	$(adr) \leftarrow (adr) \oplus (A)$	2	1	
XRL adr, #data	$(adr) \leftarrow (adr) \oplus data$	3	2	
SWAP A	$(A) \leftarrow (A(3 \div 0) * 16 - A(7 \div 4))$	1	1	

Přesunové instrukce

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
MOV A, Rr	$(A) \leftarrow (Rr)$	1	1	P
MOV A, @Rr	$(A) \leftarrow ((Rr))$	1	1	P
MOV Rr, A	$(Rr) \leftarrow (A)$	1	1	
MOV @Rr, A	$((Rr)) \leftarrow (A)$	1	1	P
MOV A, adr	$(A) \leftarrow (adr)$	2	1	P
MOV A, #data	$(A) \leftarrow data$	2	1	P
MOV Rr, #data	(Rr) ← data	2	1	
MOV @Rr, #data	$((Rr)) \leftarrow data$	2	1	
MOV adr, A	$(adr) \leftarrow (A)$	2	1	
MOV Rr, adr	$(Rr) \leftarrow (adr)$	2	1	
MOV adr, Rr	$(adr) \leftarrow (Rr)$	2	1	
MOV @Rr, adr	$((Rr)) \leftarrow (adr)$	2	1	
MOV adr, @Rr	$(adr) \leftarrow ((Rr))$	2	1	
MOV adr1, adr2	$(adr1) \leftarrow (adr2)$	3	2	
MOV adr, #data	$(adr) \leftarrow data$	3	2	
MOV DPTR, #data16	$(DPTR) \leftarrow data16$	3	2	

MOVC A, @A+DPTR	$(A) \leftarrow ((A)+(DPTR))$	1	2	P
MOVC A, @A+PC	$(A) \leftarrow ((A)+(PC))$	1	2	P
MOVX A, @DPTR	$(A) \leftarrow ((DPTR))$	1	2	P
MOVX @DPTR, A	$((DPTR)) \leftarrow (A)$	1	2	
MOVX A, @Rr	$(A) \leftarrow ((Rr))$	1	2	P
MOVX @Rr, A	$((Rr)) \leftarrow (A)$	1	2	
POP adr	$(adr) \leftarrow ((SP)),\ (SP) \leftarrow (SP)-1$	2	2	POP PSW
PUSH adr	$(SP) \leftarrow (SP)+1,\ ((SP)) \leftarrow (adr)$	2	2	
XCH A, Rr	$(A) \leftrightarrow (Rr)$	1	1	P
XCH A, @Rr	$(A) \leftrightarrow ((Rr))$	1	1	P
XCH A, adr	$(A) \leftrightarrow (adr)$	2	1	P
XCHD A,@Rr	$A(3 \div 0) \leftrightarrow (Rr(3 \div 0))$	1	1	P

Bitové operace

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
ANL C, bit	$(C) \leftarrow (C)$ **AND** (bit)	2	2	C
ANL C, /bit	$(C) \leftarrow (C)$ **AND** $\overline{(bit)}$	2	2	C
CLR C	$(C) \leftarrow 0$	1	1	C
CLR bit	$(bit) \leftarrow 0$	2	1	Příslušný bit
CPL C	$(C) \leftarrow$ **NOT** $(C) = \overline{(C)}$	1	1	C
CPL bit	$(bit) \leftarrow$ **NOT** $(bit) = \overline{(bit)}$	2	1	Příslušný bit
MOV C, bit	$(C) \leftarrow (bit)$	2	2	C
MOV bit, C	$(bit) \leftarrow (C)$	2	2	
ORL C, bit	$(C) \leftarrow (C)$ **OR** (bit)	2	2	C
ORL C, /bit	$(C) \leftarrow (C)$ **OR** $\overline{(bit)}$	2	2	C
SETB C	$(C) \leftarrow 1$	1	1	C
SETB bit	$(bit) \leftarrow 1$	2	1	Příslušný bit

Instrukce větvení programu

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
ACALL adr11	$PC(10 \div 0) \leftarrow adr\,11$	2	2	
AJMP adr11	$PC(10 \div 0) \leftarrow adr\,11$	2	2	
JB bit, rel. adr	*Je-li bit=1, potom skoč*	3	2	
JBC bit, rel. adr	*Je-li bit=1, skoč a (bit←0)*	3	2	
JC rel. adr	*Je-li C=1, potom skoč*	2	2	
JMP @A+DPTR	$(PC) \leftarrow (A)+(DPTR)$	1	2	
JNB bit, rel. adr	*Je-li bit=0, potom* $(PC) \leftarrow (PC)+rel.adr$	3	2	
JNC rel. adr	*Je-li C=0, potom* $(PC) \leftarrow (PC)+rel.adr$	2	2	
JNZ rel. adr	*Je-li* $(A) \neq 0$, *potom* $(PC) \leftarrow (PC)+rel.adr$	2	2	

JZ rel. adr	*Je-li* $(A)=0$, *potom* $(PC)\leftarrow(PC)+rel.adr$	2	2	
LCALL adr16	$PC(15\div 0)\leftarrow adr\,16$	3	2	
LJMP adr16	$PC(15\div 0)\leftarrow adr\,16$	3	2	
RET	$(PC(15\div 8))\leftarrow((SP)),\ (SP)\leftarrow(SP)-1$ $(PC(7\div 0))\leftarrow((SP)),\ (SP)\leftarrow(SP)-1$	1	2	
RETI	$(PC(15\div 8))\leftarrow((SP)),\ (SP)\leftarrow(SP)-1$ $(PC(7\div 0))\leftarrow((SP)),\ (SP)\leftarrow(SP)-1$	1	2	
SJMP rel. adr	$(PC)\leftarrow(PC)+2,\ (PC)\leftarrow(PC)+rel.adr$	2	2	

Sdružené instrukce

Syntaxe instrukce	POPIS INSTRUKCE	B	C	Příznaky
CJNE A, adr, rel. adr	$(PC)\leftarrow(PC)+3$ $Je-li\ (A)\neq(adr)$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	3	2	C
CJNE A, #data, rel. adr	$(PC)\leftarrow(PC)+3$ $Je-li\ (A)\neq data$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	3	2	C
CJNE Rr, #data, rel. adr	$(PC)\leftarrow(PC)+3$ $Je-li\ (Rr)\neq data$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	3	2	C
CJNE @Rr, #data, rel. adr	$(PC)\leftarrow(PC)+3$ $Je-li\ ((Rr))\neq data$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	3	2	C
DJNZ Rr, rel. adr	$(Rr)\leftarrow(Rr)-1$ $Je-li\ (Rr)\neq 0,$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	2	2	
DJNZ adr, rel. adr	$(adresa)\leftarrow(adresa)-1,$ $Je-li\ (adresa)\neq 0,$ $pak\ (PC)\leftarrow(PC)+rel.adr$ $jinak\ (PC)\leftarrow(PC)+2$	3	2	

Vysvětlivky

Rr - registr aktivní banky (R0 ÷ R7)
@Rr - nepřímý přístup do datového prostoru, adresa je obsažena v R0, R1
adr - osmibitová přímá adresa vnitřního datového prostoru
#data - osmibitová konstanta
#data16 - šestnáctibitová konstanta
adr11 - jedenáctibitová adresa programové paměti
adr16 - šestnáctibitová adresa programové paměti
@DPTR - nepřímý přístup do vnější datové paměti
bit - osmibitová adresa bitu

rel.adr - osmibitová hodnota pro relativní adresaci (od -128 do 127)
symbol - přímá adresa registru nebo paměťového místa (adresa střadače ≡ ACC)
(symbol) - je-li symbol uzavřen v závorce jedná se o obsah příslušného registru nebo paměťového místa
((symbol)) - obsah adresy adresované obsahem příslušného registru nebo paměťového místa

7.2. Speciální registry procesoru 8051, 8052, 80C251, 80C51XA

Speciální registry procesoru 8051									
Symbol Přímá adr.	Popis Stav po nulování	Označení bitu Bitová adresa							
ACC E0H	Střadač 00H	**ACC.7** E7H	**ACC.6** E6H	**ACC.5** E5H	**ACC.4** E4H	**ACC.3** E3H	**ACC.2** E2H	**ACC.1** E1H	**ACC.0** E0H
B F0H	Registr B 00H	**B.7** F7H	**B.6** F6H	**B.5** F5H	**B.4** F4H	**B.3** F3H	**B.2** F2H	**B.1** F1H	**B.0** F0H
IE A8H	Povolení přeruš. 00H	**EA** AFH			**ES** ACH	**ET1** ABH	**EX1** AAH	**ET0** A9H	**EX0** A8H
IP B8H	Úroveň přerušení 00H				**PS** BCH	**PT1** BBH	**PX1** BAH	**PT0** B9H	**PX0** B8H
TCON 88H	Řízení časovačů 00H	**TF1** 8FH	**TR1** 8EH	**TF0** 8DH	**TR0** 8CH	**IE1** 8BH	**IT1** 8AH	**IE0** 89H	**IT0** 88H
SCON 98H	Řízení seriov.kan. 00H	**SM0** 9FH	**SM1** 9EH	**SM2** 9DH	**REN** 9CH	**TB8** 9BH	**RB8** 9AH	**TI** 99H	**RI** 98H
P0 80H	Brána 0 FFH	**P0.7** 87H	**P0.6** 86H	**P0.5** 85H	**P0.4** 84H	**P0.3** 83H	**P0.2** 82H	**P0.1** 81H	**P0.0** 80H
P1 90H	Brána 1 FFH	**P1.7** 97H	**P1.6** 96H	**P1.5** 95H	**P1.4** 94H	**P1.3** 93H	**P1.2** 92H	**P1.1** 91H	**P1.0** 90H
P2 A0H	Brána 2 FFH	**P2.7** A7H	**P2.6** A6H	**P2.5** A5H	**P2.4** A4H	**P2.3** A3H	**P2.2** A2H	**P2.1** A1H	**P2.0** A0H
P3 B0H	Brána 3 FFH	**P3.7** **RD** B7H	**P3.6** **WR** B6H	**P3.5** **T1** B5H	**P3.4** **T0** B4H	**P3.3** **INT1** B3H	**P3.2** **INT0** B2H	**P3.1** **TxD** B1H	**P3.0** **RxD** B0H
PSW D0H	Stavové slovo 00H	**CY** D7H	**AC** D6H	**F0** D5H	**RS1** D4H	**RS0** D3H	**OV** D2H		**P** D0H
PCON 87H	Řízení odběru 00xx0000B	**SMOD**						**PD**	**IDL**
TMOD 89H	Řízení časovačů 00H	**GATE**	**C/T**	**M1**	**M0**	**GATE**	**C/T**	**M1**	**M0**

Registry bez bitového označení a přístupu							
Registr	Popis	Adresa	Po nulování	**Registr**	Popis	Adresa	Po nulování
DPH	Vyšší část DPTR	83H	00H	**DPL**	Nižší část DPTR	82H	00H
SBUF	Sériový kanál	99H	xxxx xxxxB	**SP**	Ukazat. zásobníku	81H	07H
TH0	Časovač 0 - vyšší	8CH	00H	**TL0**	Časovač 0 - nižší	8AH	00H
TH1	Časovač 1 - vyšší	8DH	00H	**TL1**	Časovač 1 - nižší	8BH	00H

Tabulka D1

Speciální registry procesoru 8052									
IE A8H	Povolení přeruš. 00H	**EA** AFH		**ET2** ADH	**ES** ACH	**ET1** ABH	**EX1** AAH	**ET0** A9H	**EX0** A8H
IP B8H	Úroveň přerušení 00H			**PT2** F5H	**PS** F4H	**PT1** F3H	**PX1** F2H	**PT0** F1H	**PX0** F0H
T2CON C8H	Řízení časovače 2 00H	**TF2** CFH	**EXF2** CEH	**RCLK** CDH	**TCLK** CCH	**EXEN2** DBH	**TR2** CAH	**C/T2** C9H	**CP/RL2** C8H

Tabulka D2

Adresy obslužných podprogramů přerušení 8051					
Zdroj přerušení	Adresa	Priorita	Zdroj přerušení	Adresa	Priorita
Reset (nulování)	0000H	nejvyšší	Vnější přerušení INT0	0003H	0
Přetečení časovače 0	000BH	1	Vnější přerušení INT1	0013H	2
Přetečení časovače 1	001BH	3	Sériový kanál	0023H	4

Tabulka D3

Speciální registry procesoru 8x251SB									
Symbol Přímá adr.	Popis Stav po nulování	Označení bitu							
ACC S:0E0H	Střadač 00H	**ACC.7**	**ACC.6**	**ACC.5**	**ACC.4**	**ACC.3**	**ACC.2**	**ACC.1**	**ACC.0**
B S:0F0H	Registr B 00H	**B.7**	**B.6**	**B5**	**B.4**	**B.3**	**B.2**	**B.1**	**B.0**
CCON S:0D8H	Řídicí reg. PCA 00x00000B	**CF**	CR	---	**CCF4**	**CCF3**	**CCF2**	**CCF1**	**CCF0**
CH S:0F9H	Horní byte čít. PCA 00H	**CH.7**	CH.6	CH.5	**CH.4**	**CH.3**	**CH.2**	**CH.1**	**CH.0**
CL S:0E9H	Dolní byte čít. PCA 00H	**CL.7**	CL.6	CL.5	**CL.4**	**CL.3**	**CL.2**	**CL.1**	**CL.0**
CMOD S:0D9H	Mod čítače PCA 00xxx000B	**CIDL**	WDTE	---	---	---	**CPS1**	**CPS0**	**ECF**
IE0 S:0A8H	Povolení přeruš. 00H	**EA**	EC	ET2	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
IPH0 S:0B7H	Úroveň přerušení H x0000000B	---	IPH0.6	IPH0:5	IPH0.4	**IPH0.3**	IPH0.2	IPH0.1	IPH0.0
IPL0 S:0B8H	Úroveň přerušení L x0000000B	---	IPL0.6	IPL0:5	IPL0.4	**IPL0.3**	IPL0.2	IPL0.1	IPL0.0
P0 S:080H	Brána 0 FFH	**P0.7**	**P0.6**	**P0.5**	**P0.4**	**P0.3**	**P0.2**	**P0.1**	**P0.0**
P1 S:090H	Brána 1 FFH	**P1.7**	**P1.6**	**P1.5**	**P1.4**	**P1.3**	**P1.2**	**P1.1**	**P1.0**
P2 S:0A0H	Brána 2 FFH	**P2.7**	**P2.6**	**P2.5**	**P2.4**	**P2.3**	**P2.2**	**P2.1**	**P2.0**
P3 S:0B0H	Brána 3 FFH	**P3.7**	**P3.6**	**P3.5**	**P3.4**	**P3.3**	**P3.2**	**P3.1**	**P3.0**
PCON S:087H	Řízení odběru 00xx0000B	**SMOD1**	SMOD0	---	POF	GF1	GF0	**PD**	**IDL**

Tabulka D4

PSW S:0D0H	Stavové slovo 00H	**CY**	**AC**	**F0**	**RS1**	**RS0**	**OV**	UD	P
PSW1 S:0D1H	Stavové slovo 00H	**CY**	**AC**	**N**	**RS1**	**RS0**	**OV**	Z	—
SCON S:098H	Řízení seriov.kan. 00H	**FE/SM0**	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**
T2CON S:0C8H	Řízení časovače2 00H	**TF2**	**EXF2**	**RCLK**	**TCLK**	**EXEN2**	**TR2**	**C/T2#**	**CP/RL2**
T2MOD S:0C9H	Řízení časovače2 xxxxxxx00H	—	---	---	---	---	---	**T2OE**	**DCEN**
TCON S:088H	Řízení časovačů 00H	**TF1**	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**
TMOD S:089H	Řízení časovačů 00H	**GATE1**	**C/T1#**	**M11**	**M01**	**GATE0**	**C/T0#**	**M10**	**M00**

Registry jejichž bitové označení se skládá ze jména a čísla bitu jako u ACC (např. jmeno.5)

Registr	Popis	Adresa	Stav po resetu	Registr	Popis	Adresa	Stav po resetu
CCAP0H	Zách/komp reg.H0	S:0FAH	xxxx xxxxB	**CCAP1H**	Zách/komp reg.H1	S:0FBH	xxxx xxxxB
CCAP2H	Zách/komp reg.H2	S:0FCH	xxxx xxxxB	**CCAP3H**	Zách/komp reg.H3	S:0FDH	xxxx xxxxB
CCAP4H	Zách/komp reg.H4	S:0FEH	xxxx xxxxB	**CCAP0L**	Zách/komp reg.L0	S:0EAH	xxxx xxxxB
CCAP1L	Zách/komp reg.L1	S:0EBH	xxxx xxxxB	**CCAP2L**	Zách/komp reg.L2	S:0ECH	xxxx xxxxB
CCAP3L	Zách/komp reg.L3	S:0EDH	xxxx xxxxB	**CCAP4L**	Zách/komp reg.L4	S:0EEH	xxxx xxxxB
DPH	Vyšší část DPTR	S:083H	00H	**DPL**	Nižší část DPTR	S:082H	00H
DPXL	Rozšíř. část DPX	S:084H	01H	**RCAP2H**	Časov.2 horní část	S:0CBH	00H
RCAP2L	Časov.2 dolní část	S:0CAH	00H	**SADDR**	Registr adresy	S:0A9H	00H
SADEN	Maska adresy	S:0B9H	00H	**SBUF**	Reg. sériov.kanálu	S:099H	xxxx xxxxB
SP	Ukazat. zásobníku	S:081H	07H	**SPH**	Rozšíř.ukaz.zásob.	S:0BDH	00H
TH0	Horní část časov.0	S:08CH	00H	**TL0**	Dolní část časov.0	S:08AH	00H
TH1	Horní část časov.1	S:08DH	00H	**TL1**	Dolní část časov.1	S:08BH	00H
TH2	Horní část časov.2	S:0CDH	00H	**TL2**	Dolní část časov.2	S:0CCH	00H
WDTRST	Nulování watchdog	S:0A6H	xxxx xxxxB				

Tabulka D4 dokončení

Speciální registry procesoru XA - G3									
Symbol Přímá adr.	Popis Stav po nulování	Označení bitu							
BRC 46AH	Konfigurace BUS viz.text	---	---	---	**WAITD**	**BUSD**	**BC2**	**BC1**	**BC0**
BTRH 469H	Časování BUS high FFH	**DW1**	**DW0**	**DWA1**	**DWA0**	**DR1**	**DR0**	**DRA1**	**DRA0**
BTRL 468H	Časování BUS low EFH	**WM1**	WM0	ALEW	—	**CR1**	**CR0**	**CRA1**	**CRA0**
IEH 427H	Povol přeruš. high 00H	---	---	---	---	**ETI1**	**ERI1**	**ETI0**	**ERI0**
IEL 426H	Povol přeruš. low 00H	EA	---	---	ET2	**ET1**	**EX1**	**ET0**	**EX0**
IPA0 4A0H	Priorita přerušení 0 00H	---	PT0			---	PX0		

IPA1 4A1H	Priorita přerušení 1 00H	---	PT1			---	PX1		
IPA2 4A2H	Priorita přerušení 2 00H	---	---			---	PT2		
IPA3 4A3H	Priorita přerušení 3 00H	---	PTI0			---	PRI0		
IPA4 4A4H	Priorita přerušení 4 00H	---	PTI1			---	PRI1		
P0 430H	Brána 0 FFH	**AD7**	**AD6**	**AD5**	**AD4**	**AD3**	**AD2**	**AD1**	**AD0**
P1 431H	Brána 1 FFH	**T2EX**	**T2**	**TxD1**	**RxD1**	**A3**	**A2**	**A1**	**WRH**
P2 432H	Brána 2 FFH	**P2.7**	**P2.6**	**P2.5**	**P2.4**	**P2.3**	**P2.2**	**P2.1**	**P2.0**
P3 433H	Brána 3 FFH	**RD**	**WR**	**T1**	**T0**	**INT1**	**INT0**	**TxD0**	**RxD0**
PCON 404H	Řízení odběru 00H	---	---	---	---	---	---	**PD**	**IDL**
PSWH 401H	Stavové slovo high viz.text	**SM**	**TM**	**RS1**	**RS0**	**IM3**	**IM2**	IM1	**IM0**
PSWL 400H	Stavové slovo low viz.text	**C**	**AC**	---	---	---	**V**	N	**Z**
PSW51 402H	Stavové slovo low viz.text	**C**	**AC**	**F0**	**RS1**	**RS0**	**V**	F1	**P**
S0CON 420H	Řízení seriov.kan.0 00H	**SM0_0**	**SM1_0**	**SM2_0**	**REN_0**	**TB8_0**	**RB8_0**	**TI_0**	**RI_0**
S0STAT 421H	Řízení seriov.kan.0 00H	---	---	---	---	**FE0**	**BR0**	**OE0**	**STINT0**
S1CON 424H	Řízení seriov.kan.1 00H	**SM0_1**	**SM1_1**	**SM2_1**	**REN_1**	**TB8_1**	**RB8_1**	**TI_1**	**RI_1**
S1STAT 425H	Řízení seriov.kan.1 00H	---	---	---	---	**FE1**	**BR1**	**OE1**	**STINT1**
SCR 440H	Systémová konfig. 00H	---	---	---	---	**PT1**	**PT0**	**CM**	**PZ**
SSEL 403H	Výběr seg. registru 00H	ESwen	R6SEG	R5SEG	R4SEG	R3SEG	R2SEG	R1SEG	R0SEG
SWE 47AH	Povol. soft.přeruš. 00H	---	SWE7	SWE6	SWE5	SWE4	SWE3	SWE2	SWE1
SWR 42AH	Žádost soft.přeruš. 00H	---	SWR7	SWR6	SWR5	SWR4	SWR3	SWR2	SWR1
T2CON 418H	Řízení časovače2 00H	**TF2**	**EXF2**	**RCLK0**	**TCLK0**	**EXEN2**	**TR2**	**C/T2**	**CP/RL2**
T2MOD 419H	Řízení časovače2 xxxxxxx00H	---	---	RCLK1	RCLK1	---	---	**T2OE**	**DCEN**
TCON 410H	Řízení časovačů 00H	**TF1**	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**
TMOD 45CH	Řízení časovačů 00H	**GATE1**	**C/T1**	**M1**	**M0**	**GATE0**	**C/T0**	**M1**	**M0**
TSTAT 411H	Rozšíř. status 0 a 1 00H	---	---	---	---	---	**T1OE**	---	T0OE
WDCON 41FH	Řízení WD časov. viz.text	**PRE2**	**PRE1**	**PRE0**	---	---	**WDRun**	**WDTof**	---

Registry bez bližší identifikace bitů

Registr	Popis	Adresa	Stav po resetu	**Registr**	Popis	Adresa	Stav po resetu
CS	Kódový segment	443H	00H	**DS**	Datový segment	441H	00H
ES	Zvláštní segment	442H	00H				
P0CFGA	Konfigurace P0	470H	Viz.text	**P0CFGB**	Konfigurace P0	4F0H	Viz.text

P1CFGA	Konfigurace P1	471H	Viz.text	**P1CFGB**	Konfigurace P1	4F1H	Viz.text
P2CFGA	Konfigurace P2	472H	Viz.text	**P2CFGB**	Konfigurace P2	4F2H	Viz.text
P3CFGA	Konfigurace P3	473H	Viz.text	**P3CFGB**	Konfigurace P3	4F3H	Viz.text
RTH0	Přednast. T0 high	455H	00H	**RTH1**	Přednast. T1 high	457H	00H
RTL0	Přednast. T0 low	454H	00H	**RTL1**	Přednast. T1 low	456H	00H
S0BUF	Reg.sériov.kan. 0	460H	xxH	**S1BUF**	Reg.sériov.kan. 1	464H	xxH
S0ADDR	Adr.reg serial 0	461H	00H	**S1ADDR**	Adr.reg serial 1	465H	00H
S0ADEN	Povol.reg serial 0	462H	00H	**S1ADEN**	Povol.reg serial 1	466H	00H
TH0	Horní část časov.0	451H	00H	**TL0**	Dolní část časov.0	450H	00H
TH1	Horní část časov.1	453H	00H	**TL1**	Dolní část časov.1	452H	00H
TH2	Horní část časov.2	459H	00H	**TL2**	Dolní část časov.2	458H	00H
T2CAPH	Zácht. reg T2 high	45BH	00H	**T2CAPL**	Zácht. reg T2 low	45AH	00H
WDL	Přednastavení WD	45FH	00H	**WFEED1**	Přístupový reg.1	45DH	xxH
WFEED2	Přístupový reg.2	45EH	xxH				

Tabulka D5

Literatura

[1] Intel: Microcontroller Handbook, Santa Clara 1986
[2] Intel: MCS-51 Macro Assembler User's Guide, Santa Clara 1988
[3] Babák, M., Chládek. L.: Architektura a technické vlastnosti jednočipových mikrořadičů 8051, Tesla Eltos, Praha 1987
[4] Zděnek, J.: Mikropočítače 51, ČSVTS-ČVUT, Praha 1991
[5] Intel: Embedded microcontroller 8xC251, Santa Clara 1995
[6] Dallas: Short-Form Catalog, duben 1996
[7] Atmel: Microcontroller Handbook, San Jose 1996
[8] Philips: Semiconductors 80C51-Based 8-bit Microcontrollers, 1995
[9] Siemens: Microcontroller SAB80515/535, SAB8080517/537, 1993
[10] Skalický, P.: Digitální filtrace a signálové procesory, Skripta ČVUT, Praha 1995
[11] Philips: 16-bit 80C51XA Microcontrollers, CD-ROM 1997

STŘEDNÍ PRŮMYSLOVÁ ŠKOLA ELEKTROTECHNICKÁ V ÚŽLABINĚ 320 PRAHA 10

PRAHA 10, V ÚŽLABINĚ 320 SPŠE je státní střední odborná škola, specializující se na výpočetní techniku, automatizační techniku, informační technologie a elektrotechniku. Čtyřleté denní studium je určeno absolventům základních škol a je zakončeno maturitní zkouškou. Absolventi školy získávají úplné střední vzdělání.

Škola si během své téměř dvacetileté existence vybudovala dobré jméno mezi pražskými průmyslovými školami. Rozšířila nabídku studijních oborů, vybudovala nové učebny pro výuku odborných předmětů a cizích jazyků, snaží se pravidelně obnovovat vybavení odborných učeben i vybavení pro mimoškolní činnost žáků, sleduje moderní trendy v oblasti programového vybavení a samozřejmě pečuje o zvyšování odborné kvalifikace pedagogů.

Školu navštěvuje přibližně 720 žáků, převažují chlapci, obor technické lyceum je vhodný i pro dívky. Každoročně otevíráme šest tříd po třiceti žácích.

Studijní obory a jejich stručná charakteristika

- Automatizační technika

Žáci se seznámí s možnostmi využití výpočetní techniky v oblasti automatizace, získají přehled o automatizačních prostředcích používaných v současné době, o principech snímacích prvků, o způsobech řízení a regulace. Naučí se programovat moderní logické automaty, programovat v assembleru a v jazyce Pascal včetně prostředí Delphi, zároveň se naučí základům speciálních programovacích jazyků využívaných pro nastavení funkcí řídících automatů.

- Elektronické počítačové systémy

Žáci se seznámí s architekturou počítačů typu PC a s vývojovými trendy v této oblasti. Pochopí principy činnosti jednotlivých částí počítače a zákonitosti jejich vzájemné provázanosti. Žáci získají poznatky o návrzích logických obvodů, o operačních systémech (např. Linux), naučí se programovat v assembleru, v jazyce Pascal včetně prostředí Delphi, seznámí se s problematikou počítačových sítí.

- Informační technologie

Žáci se seznámí s architekturou počítačů, s jejich zapojením do sítě a s principy ochrany dat při sdílení a přenosu po síti. Naučí se využívat vhodné programové vybavení pro kompletní návrh a sestavení sítě a pro její zabezpečení. Naučí se základům programování v assembleru, v jazyce Pascal včetně prostředí Delphi, konfigurovat síťové operační systémy (např. Linux).

- Technické lyceum

Žáci získají rozsáhlé všeobecné vzdělání, které odpovídá úrovni gymnázia, a kvalitní odborné vzdělání, které odpovídá úrovni SOŠ se zaměřením na výpočetní techniku. Důraz se klade na matematiku a fyziku. V odborných předmětech si žáci osvojí základy elektrotechniky, naučí se pracovat s CAD systémy, důraz se klade na grafické systémy. Žáci se naučí tvořit webové stránky a využívat prezentačních programů.

Uplatnění absolventů

Absolventi školy jsou velmi žádáni, nacházejí uplatnění ve výrobních, obchodních či komerčních oblastech. Pracují podle oborů například jako programátoři, správci počítačových sítí, pracovníci v DTP studiích, jako servisní technici nebo jako prodejci elektroniky a počítačové techniky.

Je potěšitelné, že většina absolventů si rozšiřuje své vzdělání na vysokých školách technického zaměření. Nejžádanější je ČVUT Fakulta elektrotechnická, Vysoká škola ekonomická a Česká zemědělská univerzita.

A kde nás najdete?

Škola leží v Malešicích, v klidné pražské čtvrti plné zeleně, a je dostupná MHD ze stanic metra Želivského, Skalka nebo Florenc.

Pokud byste nás rádi navštívili, přijďte na Dny otevřených dveří, které pořádáme každoročně druhou středu v prosinci od 13.00 do 18.00 hodin.

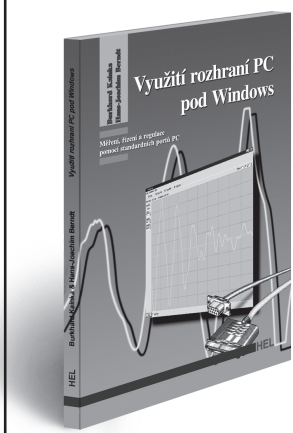
Telefony: 274 016 111, 274 774 210, 274 016 225
Internet: http://www.uzlabina.cz
E-mail: info@uzlabina.cz

Za 200 Kč
osciloskop z vašeho PC

Nebo byste rádi z vašeho počítače udělali „za pár babek“ síťový časový spínač, ovládali krokový motor, měřili čtyřkanálově napětí nebo kmitočet, navrhli si čítač či zapisovač, měřili teplotu a vlhkost vzduchu, automaticky testovali integrované obvody, postavili generátor funkcí, programátor pamětí EPROM, paměťový osciloskop, …

Řešení je poměrně jednoduché: kupte si knihu:

Využití rozhraní PC pod Windows



ve které najdete kromě výše zmíněných návodů i to, jak ze zvukové karty „udělat“ ve Windows kompletní osciloskop pomocí programu **SSCANP.EXE**. Ten je umístěn na doprovodném CD ROM. Na něm jsou všechny programové příklady ve VB5, Delphi 3 nebo Delphi 4 se všemi zdrojovými texty ve spustitelné formě, včetně **PORT.DLL**. CD obsahuje ještě jednu perličku – uživatelský program k univerzálnímu rozhraní **COMPUNI.EXE**.

Celý komplet stojí neuvěřitelných 198 Kč (tedy o 2 Kč méně než hlásá titulek), obj. číslo je 121040. **DOPORUČUJEME!**

Dále je v knize popsána stavba a provoz:

- ovládacích obvodů
- měřicích přístrojů
- analogově-číslicových převodníků
- měření se zvukovou kartou
- měření se záchytnou (capture) videokartou
- asynchronní sériový přenos
- a další aplikace PC.

Vedle přesného, srozumitelného a praktického popisu klasických rozhraní PC (s čím a jakým způsobem můžeme prostřednictvím software komunikovat) nás autoři seznamují se svépomocnou stavbou a programováním několika velmi zajímavých obvodů, které mohou být připojeny k portu počítače, aniž bychom z něj museli sundat kryt.

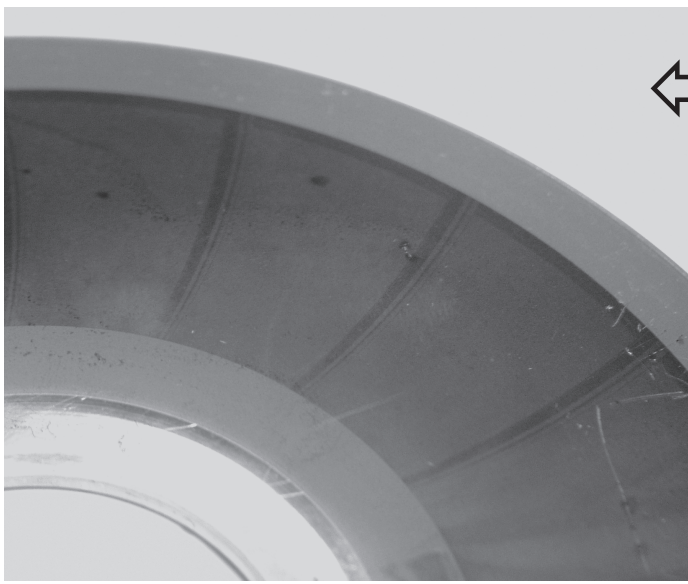
Základem aplikací popisovaných v knize jsou 32bitové systémy, tedy Windows 95/98 a NT. Přednost byla dána programovacím jazykům Visual Basic a Delphi.

Rozeberte si PC

šílená kniha nakladatelství BEN – technická literatura

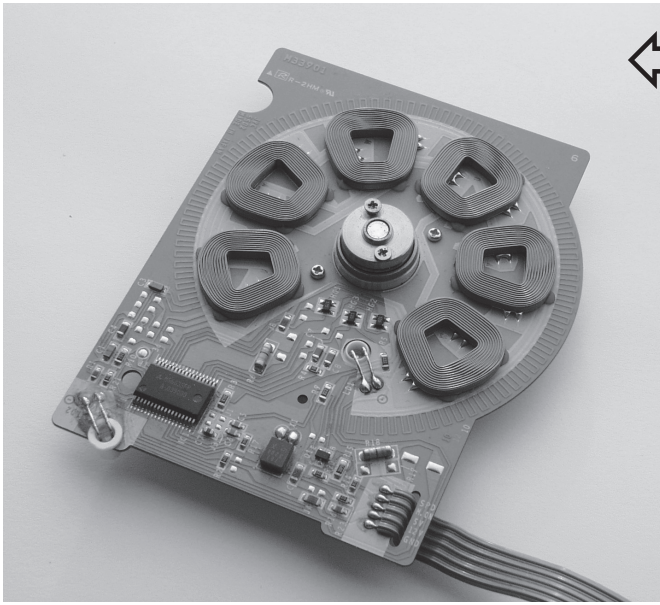
Příručka má podnadpis **„Kniha pro kutily nového tisíciletí“**, mohlo by se též dodat **„… a pro totální bastlíře“**, neboť to, co v knize najdete, **nemá ve světě obdoby**. Je tedy určena všem, kteří se chtějí snadno, rychle a bez horentních finančních nákladů dozvědět něco o elektronice, avšak **bez zbytečných a složitých teorií**. A aby pak to, co se dozví, mohli nějak prakticky využít. Kniha je psána „populárně technicky“ s humorně komentovanými příklady z praxe, **které se skutečně udály**. Je určena především pro začínající elektroniky, **badatele** a kutily.

Nevěříte? Kniha vyšla již počátkem března 2001 a stále je aktuální. Rozsah 208 stran formátu B5, objednací číslo 121051, MC 199 Kč.



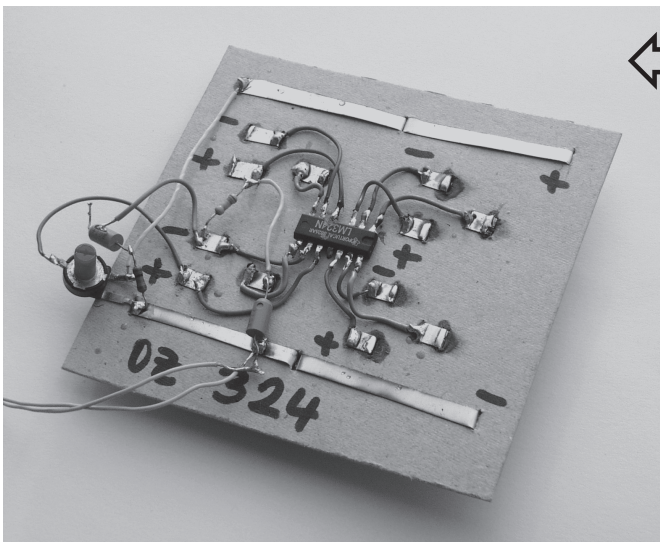
Již jste někdy viděli sektory vašeho pevného disku „živě“? Jen račte! Stačí si rozmíchat náš **„magnetickej lógr“** a vidíte data **na vlastní oči!** Jen si prosím nesrkněte jako jedna osoba zmíněná v knize …

426410



Schválně jestli poznáte, co je to na fotografii. Není to jen taková ledajaká deska. Její předností je, že je **„za pár šupů“**, že ji dostanete téměř všude a hlavně – že na ní jsou tři Hallovy generátory. Nevíte co s nimi? Dal by se s nimi postavit třeba **digitální kompas** … Nevíte jak obvody odpájíte? Stačí se jen začíst …

Diskuse a polemiky o tomto úsměvném knižním titulu můžete sledovat na internetové adrese knihy ***http://shop.ben.cz/default.asp?kam=detail.asp?id=121051***.



Co říkáte naší nové pokusné desce? Chce to pouze tvrdý papír, nůžky, fixu a nějaké děrovátko. Myslíte, že by mohlo časem navlhnout? Tak to zalejte parafínem nebo nějakou jinou **„sračkou“**. Konec s nepájivými poli a dalšími jinými pokusnými deskami. Dokud si to sami nevyzkoušíte, neuvěříte, že je něco takového vůbec možné! S chutí do toho …

V nakladatelství BEN – technická literatura dosud vyšlo:

Mikrokontroléry PIC

Komunikace mikrokontroléru s okolím 1.

V první a druhé části této publikace je podrobně rozebrána komunikace s různými druhy klávesnic a zobrazovacích jednotek. Zájemce zde nalezne i velké množství konkrétních řešení od nejjednodušších až po využití inteligentních obvodů včetně způsobu jejich programové obsluhy.

Třetí část je věnována nejčastěji používaným způsobům komunikace mikrokontrolérů s okolními systémy, ať jsou těmito systémy jiné mikrokontroléry, počítače PC, nebo jiné více či méně inteligentní elektronické systémy. Jde o komunikace SCI (RS232C), SPI, I^2C, jednovodičová DALLAS a komunikaci s IR systémy. Opět i zde čtenář nalezne příklady jejich programové obsluhy.

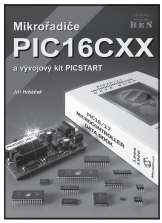
Autor Jiří Hrbáček, 160 stran B5 + disketa, obj. číslo 120921.

Komunikace mikrokontroléru s okolím 2.

Volně navazuje na první díl. Doplňuje uvedené informace a klade si za cíl, seznámit podrobně čtenáře se zajímavými obvody používanými ve spolupráci s mikrokontroléry. Z Obsahu: Obvody automatické identifikace, adresovatelné spínače, digitální teploměry, dotykové paměti, bezdrátová komunikace

Probíraná témata jsou vysvětlována na příkladech, konkrétní řešení jsou pak ukázána s použitím mikrokontrolérů PIC. Uvedené informace a postupy jsou však velice užitečné i pro ty, kteří používají jiné typy mikrokontrolérů.

Autor Jiří Hrbáček, 152 stran B5 + disketa, obj. číslo 120983.



MIKROŘADIČE PIC16CXX

Kniha poskytuje čtenáři základní informace o mikrořadičích řady PIC16CXX, jejich vlastnostech a použití tak, aby je mohl využít při vlastních konstrukcích zařízení. Začátek textu je věnován popisu mikrořadičů PIC16C54, 55, 56, 57, 71 a 84. Další kapitoly podávají základní informace o assembleru MPALC, simulátoru MPSIM a programování pomocí programátoru PICSTART. Informace jsou zde předkládány ve formě návodů. Výklad není doplňován teoretickými odvozeními a důkazy.

Autor Jiří Hrbáček, 144 stran B5, obj. č. 180029.

PROGRAMOVÁNÍ MIKROKONTROLÉRŮ PIC16CXX

V úvodu této publikace je vysvětlení základních pojmů, používaných ve výpočetní technice a způsob převodu čísel mezi dvojkovou, šestnáctkovou a desítkovou soustavou. Výuka programování je nemyslitelná bez praktických zkoušek programovaných aplikací. Proto je zde postupně ukázán soubor tréninkových desek (bastldesek, jak se říká mezi amatéry). Výuka programování je vedena na příkladech od nejjednodušších až po složité systémy. Na nich je ukázána funkce jednotlivých instrukcí mikrokontrolérů, způsob psaní programů, praktické rady k usnadnění programování (pomocné stránky, poznámky na konci každého příkladu).

Autor Jiří Hrbáček, 112 stran B5, obj. číslo 180036.

V nakladatelství BEN – technická literatura dosud vyšlo:

Mikrokontroléry MOTOROLA

Začínáme pracovat s mikrokontroléry Motorola HC08 NITRON

– příručka pro naprosté začátečníky

Kniha je "kuchařkou" pro první pokusy s mikrokontroléry. Obsahuje vše podstatné co začátečník potřebuje, neodrazuje svoji "tloušťkou".

Hlavním důvodem k napsání knihy, byla soutěž české pobočky MOTOROLA o nejlepší konstrukci s novými mikrokontroléry HC08 NITRON nebo jiným modelem rodiny HC08. O soutěž je stále velký zájem, neboť v krátké době rozeslala MOTOROLA několik stovek stavebnic vývojového kitu JANUS s mikrokontrolérem NITRON.

Výhodou doprovodného CD je i to, že kromě části věnované kitu JANUS (kompletní stavební návod včetně klišé plošného spoje a vývojového prostředí) obsahuje celou řadu materiálů o rodině mikrokontrolérů HC08, včetně nejmenšího modelu NITRON. Zvláštní pozornost je vhodné věnovat speciální výukové prezentaci. Ta totiž přehledně vysvětluje jak funkci jádra procesoru, tak jeho periferních subsystémů. A jako bonus si můžete zvolit ozvučenou variantu a pocvičit se současně v náslechu technické angličtiny.

Na CD jsou rovněž obsaženy zdrojové i přeložené formy všech příkladů realizovaných v knize.

Obsah:

Autor Vladimír Váňa, 96 stran B5 + CD ROM,
obj. číslo 121170.

DOPORUČUJEME

426410

Knihy nakladatelství BEN – technická literatura:

USB

Autor David Matoušek, 272 stran B5 + CD ROM, obj. č. 121136.

USB prakticky s obvody FTDI – 1. díl
edice **PC & elektronika**

Kniha se věnuje popisu a praktickému použití obvodu FT232BM (cena asi 180 Kč) vyráběného firmou FTDI Chip.

Tento obvod pracuje jako konvertor signálů sběrnice USB na signály asynchronního sériového kanálu včetně linek modemu. Také je schopen pracovat v paralelním režimu (Bit Bang). Pomocí vnější E^2PROM lze stanovit VID, PID, sériové číslo a popis, takže je možno vytvářet nejen amatérské, ale i plně profesionální konstrukce.

Po úvodním popisu sběrnice USB a obvodu FT232BM jsou uvedeny a vysvětleny funkce ovládacího rozhraní, které umožňuje vytvářet aplikace pro operační systémy Windows 98/2000/Me/XP (je zde i podpora pro operační systémy Linux).

Dále jsou uvedeny vlastnosti modulů firmy ASIX (použití modulů osazených obvodem FT232BM zjednodušuje a urychluje proces vývoje aplikací).

Následuje kapitola představující základy práce s obvodem FT232BM včetně tvorby jednoduché testovací desky (instalace ovladačů, programování konfigurační E^2PROM přímo v aplikaci, zjištění připojených zařízení, přímé řízení linek modemu, řízení linek v režimu Bit Bang).

Další dvě kapitoly představují programátory mikrokontrolérů AT89C2051 a AT90S2313. Napájení je získáno přímo ze sběrnice USB.

Následuje kapitola s příklady použití mikrokontrolérů AT89C2051 a AT90S2313 pro asynchronní sériovou komunikaci s obvodem FT232BM.

Dále jsou publikovány napájecí zdroj a měřicí deska (obojí s mikrokontrolérem AT90S2313). Měřicí deska je navíc napájena přímo ze sběrnice USB.

Nakonec jsou zařazeny přílohy (programátor konfiguračních E^2PROM, konvertory USB<=>RS-232C atd.).

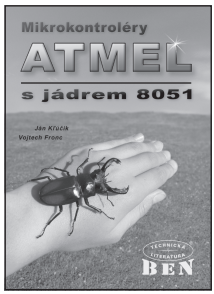
Kniha rovněž obsahuje popis konstrukce přípravků (včetně desek plošných spojů) pro všechny publikované příklady. Přiložené CD-ROM obsahuje klišé plošných spojů přípravků a zdrojové kódy všech publikovaných příkladů.

Stručný obsah: Co najdete na doprovodném CD-ROM; O knize; 1. Základní pojmy USB; 2. Popis obvodu FT232BM; 3. Ovládací rozhraní; 4. Zařízení USB od firmy ASIX; 5. Univerzální modul s FT232BM; 6. ATPROG 3.0; 7. USBAVR – vývojové kity pro mikrokontrolér AT90S2313; 8. Příklady komunikace vybraných mikrokontrolérů s obvodem FT232BM; 9. Napájecí zdroj s regulovatelnou proudovou pojistkou; 10. USBMC – Univerzální měřicí deska; 11. EFSProg – Programátor konfiguračních E^2PROM; 12. Konvertory sběrnice USB<=>RS-232C a použití modulu UMS2; 13. Výpis souborů EFS.INF a FTD2XXUN.INI; 14. Dodavatelé součástek, modulů a hotových přípravků uvedených v této knize; Přehled přípravků; Plošné spoje; Literatura.

426410

Knihy nakladatelství BEN – technická literatura:

Mikrokontroléry ATMEL



Mikrokontroléry ATMEL s jádrem 8051

Kniha představuje základní popis procesorů ATMEL (architektura, přehled typové řady, instrukční soubor). Kniha se zabývá typy AT89C1051/2051/4051, AT89C51/52/53/55, AT89S8252/4D12. V knize nejsou obsaženy procesory typu AVR.

Autoři Ján Kľúčik a Vojtech Fronc, 128 stran B5, obj. číslo 180046.

Učebnice programování ATMEL s jádrem 8051

Cílem této publikace je nastínit metodiku a popsat několik algoritmů programování těchto mikroprocesorů, protože dostatečně podrobný popis programovacích metod v jazyku symbolických instrukcí Assembler na trhu doposud chyběl. První část se zabývá podrobným popisem instrukcí jazyka Assembler. Součástí popisu každé instrukce je podrobné vysvětlení její funkce včetně popisu změn stavového slova procesoru a její typické použití ve zdrojovém kódu programu, tj. včetně příkladu.

Tato kniha je určena pro všechny skupiny zájemců o programování procesorů ATMEL s jádrem 8051, především však začátečníkům.

Autor Václav Vacek, 144 stran B5, obj. č. 121072.

Učebnice programování PIC

Tato učebnice je obdobou "Učebnice programování ATMEL s jádrem 8051", ale pro procesory PIC. Na rozdíl od výše uvedené knihy, je navíc v poslední části popsána konstrukce jednoduchého programátoru procesoru PIC16F84, který byl vybrán jako vzorový procesor pro tuto učebnici. Tento typ je jediný, který má na svém čipu kódovou paměť typu FLASH, a je možné ho snadno elektronicky přeprogramovávat. Programátor je řešen jednoduchým způsobem přes sériový port počítače PC.

Součástí knihy je i disketa s příslušným software pro obsluhu zmíněného programátoru. Jde o dosovský program, a proto je nutné ho provozovat v operačním systému MS-DOS. Dosovské okno ve WINDOWS '98 a vyšší neemuluje operační systém MS-DOS zcela korektně a programování někdy selhává. Řešením může být restartování takového počítače v režimu MS-DOS a spuštění programu „picprog.exe“ striktně v prostředí MS-DOSu.

Autor Václav Vacek, 144 stran B5, obj. č. 121007.

Číslicová technika

Kniha velice podrobně popisuje druhy číslicových obvodů a jejich použití. Je určena nejen začátečníkům, protože i pokročilí „bastlíři“ zde naleznou řadu dosud nepublikovaných konstrukcí.

Zvláště poslední kapitola ukazuje různé užitečné konstrukce vytvořené na bázi číslicových obvodů. Jedná se o zajímavá použití ovládacích tlačítek, časovací obvody, řízení výkonových obvodů včetně použití PWM.

Autor David Matoušek, 208 stran B5, obj. č. 121060.

Knihy nakladatelství BEN – technická literatura:

EDICE PC & elektronika

Udělejte si z PC – 1. díl

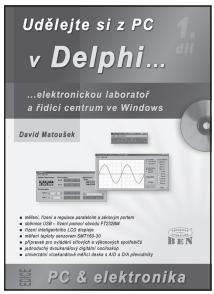
– generátor, čítač, převodník, programátor…
Měření, řízení a regulace pomocí sériového portu PC a sběrnice I²C

Autor Ing. David Matoušek, 176 stran B5 + CD ROM, obj. číslo 121069.

Udělejte si z PC – 2. díl

Měření, řízení a regulace pomocí portů PC prostřednictvím několika jednoduchých přípravků
Komunikace PC s aplikacemi mikrořadičů řady AT89C2051, stavba jednoduchého programátoru mikrořadiče AT89C2051

Autor Ing. David Matoušek, rozsah 224 stran B5 + CD ROM, obj. číslo 121114.



Udělejte si z PC v Delphi – 1. díl

elektronickou laboratoř a řídicí centrum ve Windows
Kniha uvádí konstrukce několika zařízení, která lze používat v amatérské praxi ale i mnohem obecněji: LPTLCD (ovládání LCD displeje), použití teplotního čidla SMT160-30, COM4021 (8bitový vstupní port), COM1320 (levný 8bitový D/A převodník se sběrnicí I²C), LPTUNI – univerzální deska pro paralelní port (triaky nebo relé pro ovládání síťových spotřebičů, tranzistory pro spínání stejnosměrných obvodů, jeden digitální vstup), COMOSC – dvoukanálový osciloskop s rozlišením 8 bitů (maximální vzorkovací rychlost 100 kSPS), USBMC – univerzální měřicí karta pro USB (D/A převodníky, A/D převodníky, digitální vstupy a výstupy, čítač a časovač). Další konstrukce budou uvedeny v následujícím díle.

Autor Ing. David Matoušek, rozsah 272 stran B5 + CD ROM, vázané, obj. číslo 121161.

USB – měření, řízení a regulace pomocí sběrnice USB

Autor B. Kainka, 248 stran B5 + CD ROM, obj. č. 121116.

Měření, řízení a regulace pomocí PC

Autor B. Kainka, 272 stran B5 + CD, obj. č. 121128.

Knihy nakladatelství BEN – technická literatura:

EDICE µC & praxe

Práce s mikrokontroléry Atmel AT89C2051 – 1. díl

Autor Ing. David Matoušek, 248 stran B5 + CD ROM,
obj. číslo 121093.

Práce s mikrokontroléry Atmel AT89S8252 – 2. díl

Autor Ing. David Matoušek, rozsah 304 stran B5 + CD ROM,
obj. číslo 121112.

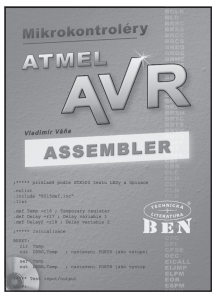
Práce s mikrokontroléry Atmel AVR – 3. díl

Autor Ing. David Matoušek, rozsah 376 stran B5 + CD ROM,
obj. číslo 121130.

C pro mikrokontroléry

Autor Mann Burkhard, rozsah 280 stran B5 + CD ROM, vázané,
obj. číslo 121120.

EDICE mikrokontroléry ATMEL AVR



Mikrokontroléry Atmel AVR
– popis procesoru a instrukční soubor

Autor Vladimír Váňa, 336 stran B5, obj. č. 121125.

Mikrokontroléry Atmel AVR – Assembler

Autor Vladimír Váňa, 144 stran B5 + CD ROM, obj. č. 121135.

Mikrokontroléry Atmel AVR
– Programování v jazyce C

Autor Vladimír Váňa, 216 stran B5 + CD, obj. č. 121139.

Mikrokontroléry Atmel AVR – vývojové prostředí

Autor Vladimír Šubrt, 96 stran B5 + CD ROM, obj. č. 121099.

Mikrokontroléry Atmel AVR – Bascom

Mikrokontroléry Atmel AVR – Pascal

426410

Knihy nakladatelství BEN – technická literatura:

DALŠÍ TITULY

Moderní učebnice programování mikrokontrolérů PIC

– 1. díl

Mikrokontroléry zaujímají nesmírně důležité místo v moderních elektronických systémech. Použití mikrokontrolérů velmi zjednodušuje tyto systémy a zvyšuje jejich schopnosti. Dá se bez nadsázky říct, že návrh a realizace elektronických systémů s použitím mikrokontrolérů patří mezi základní znalosti a dovednosti současných elektroniků. Učebnice je psána tak, že umožňuje velmi efektivní způsob samostudia. Pro úspěšné zvládnutí výuky postačují pouze nejzákladnější znalosti elektroniky a základní uživatelskou znalost práce s počítačem. Učebnici lze s výhodou využívat i v prezenční výuce.

Autor Jiří Hrbáček, 96 stran A5 + CD ROM, obj. číslo 121109.

Programátor PIC16F84 - stavební návod s ovládacím programem

Autor David Matoušek, 24 stran A5 + disketa, obj. číslo 121143.

Vývojový kit Vývojový kit USB51KIT pro AT89S51 a AT89S52

– kompletní stavební návod s ovládacím programem pro Windows

Autor David Matoušek, 24 stran A5 + CD, obj. číslo 121235.

Vývojový kit USBmegaKIT pro AVR ATmega16

– kompletní stavební návod s ovládacím programem pro Windows

Autor David Matoušek, 28 stran A5 + CD, obj. číslo 121236.

Programátor ATPROG 4.0 - univerzální programátor ATMEL na USB

– kompletní stavební návod s ovládacím programem pro Windows

Autor David Matoušek, 32 stran A5 + CD, obj. číslo 121204.

http://www.ben.cz

426410

426410

Pár slov o nakladatelství

*Nakladatelství **BEN – technická literatura** se věnuje vydávání převážně počítačové a elektrotechnické literatury. Nakladatelství je součástí stejnojmenné firmy, která se zabývá prodejem a distribucí technické a počítačové literatury, jež v poslední době vyšla.*

Adresa této knihy na Internetu:
http://shop.ben.cz/180035

Petr Skalický
MIKROPROCESORY ŘADY 8051
Vydala firma MC COMPLETE s.r.o.
v nakladatelství BEN – technická literatura, Praha 2012
6. dotisk 2. rozšířeného vydání

Vedoucí nakladatelství Libor Kubica
Vedoucí redakce a DTP Hana Züglerová
Odpovědný a technický redaktor Libor Kubica

Obálka	layout Libor Kubica, foto AFIS
Sazba	Petr Skalický
	finální layout Martin Havlák
Tisk	Typos

objednací číslo	180035
EAN	9788086056395
ISBN	80-86056-39-2 (tištěná kniha)
ISBN	978-80-7300-452-1 (elektronická kniha v PDF)

426410