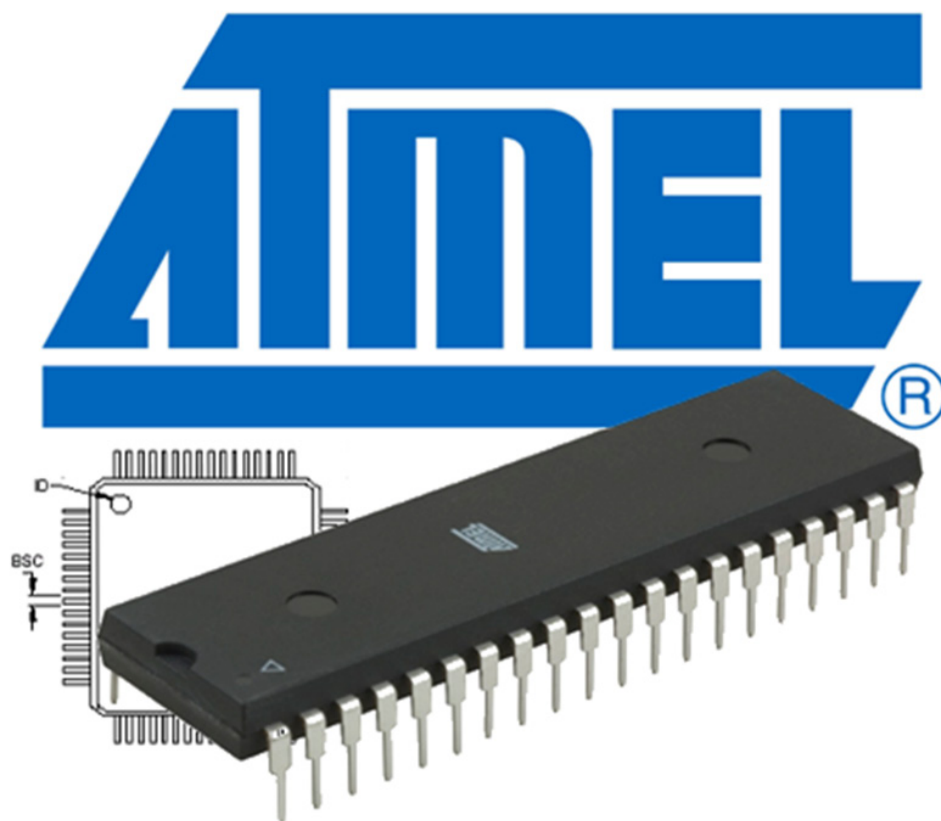


Mikroprocesory AVR Tiny

Skripta



Programování v assembleru (jazyk symbolických adres)

Paměti

Dělení mikropočítačů

Rodina mikropočítačů AVR

MCU ATtiny 13 a ATtiny 2313

Paměti MCU tiny 13/2313

Resetovací obvod MCU tiny13/2313

Časování MCU tiny13/2313

Blok čítače/časovače

Vstupně/výstupní (I/O) porty

Systém přerušení (Interrupt)

Převod analogového signálu

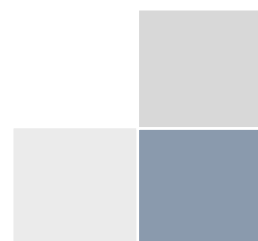
Komunikační rozhraní MCU

Řízení výkonu MCU (Power management)

Ing. Miroslav Duch

Střední průmyslová škola, Trutnov, Školní 101

2009



Mikroprocesory AVR Tiny

Skripta

Autor: Ing. Miroslav Duch

Copyright © 2009 Střední průmyslová škola, Trutnov, Školní 101. Všechna práva vyhrazena.

Tato publikace vznikla v rámci projektu „Moderní výuka mikroprocesorové techniky“ spolufinancovaného Královéhradeckým krajem.



Střední průmyslová škola, Trutnov, Školní 101

Školní 101

541 01 Trutnov 1

Tel.: 499 813 071, fax: 499 814 729

E-mail: skola@spstrutnov.cz

URL: <http://www.spstrutnov.cz>

VAŠE SPOJENÍ SE VZDĚLÁNÍM

Obsah

1. Programování v assembleru (jazyk symbolických adres)	- 1 -
2. Paměti	- 2 -
2.1. Energeticky nezávislé paměti (ROM).....	- 2 -
2.1.1. ROM.....	- 3 -
2.1.2. PROM.....	- 3 -
2.1.3. EPROM.....	- 3 -
2.1.4. EEPROM.....	- 4 -
2.1.5. FLASH.....	- 4 -
2.2. Energeticky závislé paměti (RAM)	- 4 -
2.2.1. DRAM (Dynamic RAM)	- 5 -
2.2.2. SRAM	- 5 -
3. Dělení mikropočítačů	- 6 -
3.1. Dle architektury	- 6 -
3.1.1. Von Neumannovo schéma – společná paměť programu a dat	- 6 -
3.1.2. Harwardské schéma – oddělená paměť programu a dat.....	- 6 -
3.2. Podle instrukčního souboru.....	- 7 -
3.2.1. CISC (Soubor s kompletní instrukční sadou)	- 7 -
3.2.2. RISC (Soubor s redukovanou instrukční sadou)	- 8 -
3.3. Sběrnice	- 8 -
3.4. Obecné schéma mikropočítače MCU	- 8 -
4. Rodina mikropočítačů AVR	- 9 -
5. MCU ATtiny 13 a ATtiny 2313	- 12 -
5.1. Primární parametry	- 12 -
5.1.1. ATtiny 13.....	- 12 -
5.1.2. ATtiny 2313	- 13 -
5.2. Architektura – blokové schéma	- 14 -
5.2.1. Blokové schéma tiny13 a tiny2313	- 15 -
6. Paměti MCU tiny 13/2313	- 18 -
6.1. Paměť programu MCU tiny 13/2313	- 18 -
6.2. Datová paměť MCU tiny 13/2313	- 18 -
6.2.1. EEPROM datová paměť	- 20 -
7. Resetovací obvod MCU tiny13/2313	- 20 -
8. Časování MCU tiny13/2313	- 21 -

8.1.	Interní zdroj hodinového signálu.....	- 21 -
8.2.	Taktování (časování) operandů	- 22 -
8.3.	Před-dělička systémového času	- 22 -
9.	Blok čítače/časovače.....	- 23 -
9.1.	8 bitový čítač/časovač.....	- 24 -
9.2.	16 bitový čítač/časovač	- 26 -
9.3.	PWM (Pulsně šířková modulace).....	- 27 -
9.4.	Funkce PWM u procesoru AVR.....	- 29 -
10.	Vstupně/výstupní (I/O) porty.....	- 31 -
10.1.	Přístup k portům (branám) MCU	- 32 -
10.2.	Alternativní funkce portů	- 33 -
11.	System přerušení (Interrupt).....	- 33 -
11.1.	Povolení přerušení.....	- 35 -
11.2.	Priorita přerušení.....	- 37 -
12.	Převod analogového signálu	- 37 -
12.1.	Analogový komparátor	- 37 -
12.2.	Analogově digitální (A/D) převodník	- 38 -
12.2.1.	Převod analogového signálu na digitální.....	- 39 -
12.2.2.	Přesnost a kvalita převodu signálu	- 40 -
12.2.3.	ADC ATtinny 13	- 40 -
13.	Komunikační rozhraní MCU	- 42 -
13.1.	Sériové periferní rozhraní - SPI (Seriál Peripheral Interface)	- 42 -
13.2.	USART (Univerzální synchronní asynchronní vysílač přijímač).....	- 43 -
13.3.	Registry rozhraní USART	- 44 -
14.	Řízení výkonu MCU (Power management)	- 45 -
15.	Použitá literatura.....	- 47 -

1. Programování v assembleru (jazyk symbolických adres)

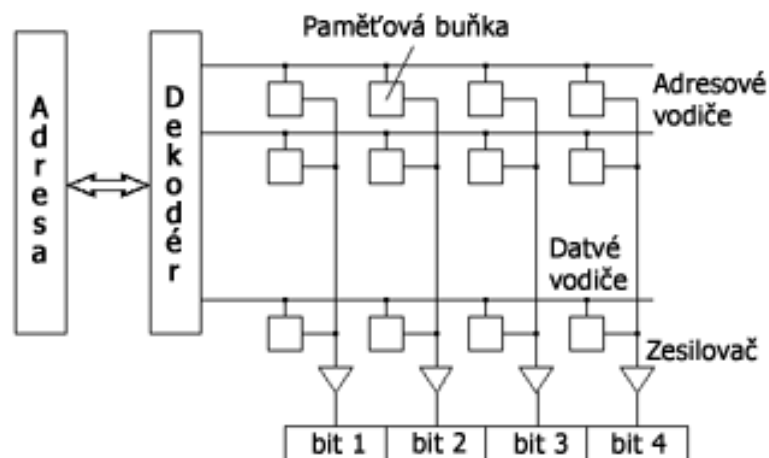
Mikroprocesory, dále jen MCU (Micro Controller Unit), mají velice široký způsob použití: počínaje osobním počítačem, kde zcela určitě sedí jeden mikroprocesor (micro controller) ve vaší klávesnici a zajišťuje posílání znaků z klávesnice do počítače nebo ve vašem automobilu, kde řídí vstřikování paliva nebo ovládá bezpečnostní systém vozu. Jednoduše MCU najdete všude tam, kde je nějaká elektronika. Hlavními faktory proč se mikropočítače tak hojně využívají, je jejich nízká cena a spotřeba (v klidovém režimu se může pohybovat v jednotkách mikroampérů, za chodu se dá dosahovat nejnižší spotřeby zhruba 500 mikroampérů). Dále jsou to rozměry procesoru s množstvím periférií (sériové a paralelní porty, A/D a D/A převodníky atd.). Jak je vidět z předchozího výčtu, dá se o MCU konstatovat, že jsou malé, ale šikovné.

U MCU hraje hlavní roli cena, které je podřízeno všechno. Jen pro zajímavost, v průmyslu platí, že cena jakéhokoliv samostatného prvku, který je součástí elektronické sekce musí být co nejlevnější, jinak by nebyla výroba efektivní. Jak je vidět, MCU tuto podmínku splňují bez problémů. Nízká cena MCU je dosahována také tím, že výrobci vyrábí mnoho variant lišících se perifériemi umístěných na čipu. Lze tak koupit například MCU s jedním, dvěma, nebo třemi paralelními porty. Díky tomu může člověk, který navrhuje zapojení s MCU, vybrat přesně obvod, který mu vyhovuje a využít jej na maximum. Nemusí platit za něco, co nepoužije.

MCU nebo, chcete-li micro controlery rodiny AVR využívající architekturu RISC (redukovaná instrukční sada) jsou velmi zdařilým výsledkem architektury mikropočítačů přizpůsobených jazyku C. I s omezeným počtem instrukcí se procesory AVR velmi přibližují MCU s architekturou CISC (procesory s kompletní instrukční sadou). Typickou vlastností RISC je zpracování instrukcí v jednom hodinovém cyklu. Kombinace zmínovaných architektur je základem moderního řešení architektury CPU (nízká velikost programu s velkou rychlostí zpracování). AVR míří tímto směrem. My se budeme během našeho studia seznamovat s mikroprocesory řady ATtiny a lehce se dotkneme složitější řady MEGA

2. Paměti

Polovodičové paměti se dělí do dvou základních skupin ROM (Read Only Memory) a RAM (Random Access Memory). Obě skupiny mají architekturu propojení jednotlivých paměťových buněk shodnou. Abychom mohli data do paměti ukládat nebo je číst, je nutné paměťové místa adresovat. Podle toho kolika bitová je adrese, tolik může paměť obsahovat adresovaných míst (např. 8bitová adresa může neadresovat $2^8\text{B} = 256\text{Bytu}$).



Obrázek 1 – obecná struktura paměti

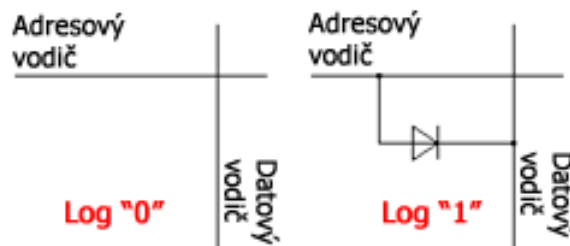
2.1. Energeticky nezávislé paměti (ROM)

Tyto paměti po odpojení elektrické energie svá data neztratí, zůstávají uložena na svých paměťových místech. Základní rozdělení energeticky nezávislých pamětí je následující:

- **ROM** (Read Only Memory)
- **PROM** (Programable Read Only Memory)
- **EPROM** (Eraseable Programable Read Only Memory)
- **EEPROM** (Electrically Eraseable Programable Read Only Memory)
- **Flash** (Flash memory)

2.1.1. ROM

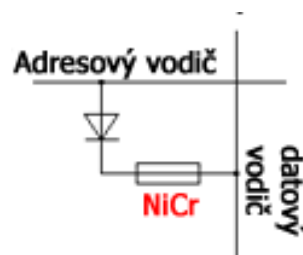
Paměti ROM jsou paměti, které jsou určeny pouze pro čtení informací. Informace jsou do těchto pamětí pevně zapsány při jejich výrobě a potom již není možné žádným způsobem jejich obsah změnit.



Obrázek 2 – Základní princip pamětné buňky ROM

2.1.2. PROM

Paměť PROM neobsahuje po vyrobení žádnou pevnou informaci a je až na uživateli, aby provedl příslušný zápis informace. Tento zápis je možné provést pouze jednou a poté již paměť slouží stejně jako paměť ROM.



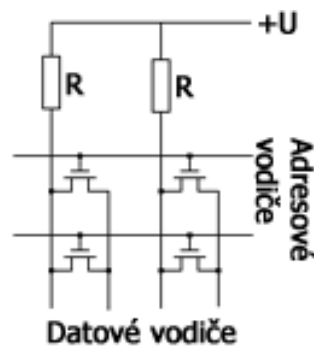
Obrázek 3 – Princip programování pamětí PROM

2.1.3. EPROM

Do paměti EPROM je možné uživatelsky zapisovat a mazat ji. Zapsané informace je možné vymazat působením ultrafialového záření. V současné době se již nepoužívá.

2.1.4. EEPROM

Paměť je možné naprogramovat a později z ní informace vymazat. Výhodou oproti EPROM pamětem je, že vymazání se provádí elektricky. Paměťové buňky jsou tvořeny pomocí MNOS tranzistorů.



Obrázek 4 – Paměťová buňka EEPROM (Tranzistory MNOS)

2.1.5. FLASH

Flash paměti jsou obdobou pamětí EEPROM. Princip zápisu a mazání jsou podobné. Jsou tvořeny hradly NAND a NOR. Struktura NAND Flash umožňuje rychlejší zápis a čtení z paměti a velikost paměťové buňky může být oproti NOR Flash menší. Z NAND flash pamětí jsou vyrobeny záznamová zařízení, jako jsou klíčenky, karty atd. Výhody těchto pamětí jsou nízká cena, velká rychlost, nízký příkon, vysoká spolehlivost.

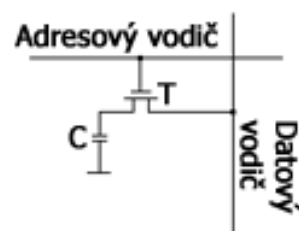
2.2. Energeticky závislé paměti (RAM)

Paměti RAM (Random Access Memory) jsou určeny pro zápis i pro čtení dat. Jedná se o paměti, které jsou energeticky závislé.

- **DRAM** (Dynamická paměť RAM)
- **SRAM** (Statická paměť RAM)

2.2.1. DRAM (Dynamic RAM)

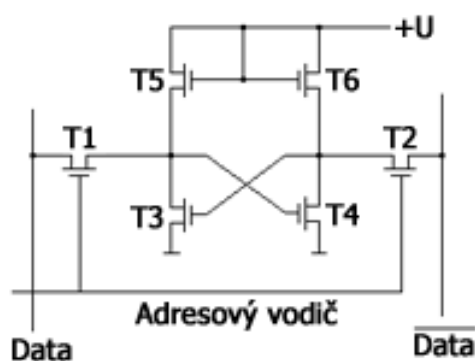
V paměti DRAM je informace uložena pomocí elektrického náboje na kondenzátoru. Tento náboj má však tendenci se vybití i v době, kdy je paměť připojena ke zdroji elektrického napájení. Aby nedošlo k vybití a tím i ke ztrátě uložené informace, je nutné periodicky provádět tzv. **refresh**, tj. obnovení logické hodnoty v paměťové buňce. Paměti DRAM mají jednoduchou strukturu a velkou kapacitu, z toho plyne nízká cena. Refresh snižuje rychlost paměti. S pamětí typu DRAM se můžete setkat např. v PC (operační paměť).



Obrázek 5 – Paměťová buňka typu DRAM

2.2.2. SRAM

Data v paměti SRAM jsou uchována po celou dobu, kdy je paměť připojena ke zdroji elektrického napájení. Paměťová buňka SRAM je realizována jako bistabilní klopný obvod, tj. obvod, který se může nacházet vždy v jednom ze dvou stavů (log „1“, log „0“). Je velmi rychlá a drahá. Vzhledem k složité struktuře paměťové buňky nedosahuje kapacit jako DRAM. Paměť SRAM nalezneme např. jako součást CPU (Cache paměti).



Obrázek 6 – paměťová buňka typu SRAM

3. Dělení mikropočítačů

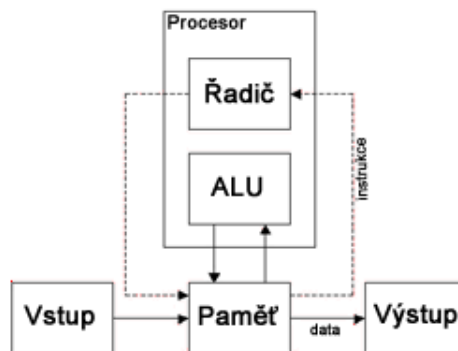
3.1. Dle architektury

Základní moduly počítače jsou procesor, řadič, operační paměť a vstupní a výstupní zařízení. Obě níže uvedené architektury tyto moduly využívají rozdílně.

3.1.1. Von Neumannovo schéma – společná paměť programu a dat

Výhodou pro programátora je, že si může paměť pro kód (program) a data určit sám. Dále platí, že řídicí jednotka procesoru přistupuje do paměti pro data i pro instrukce jednotným způsobem po jedné sběrnici (jednodušší výroba).

Nevýhodou se stává společné uložení dat a kódu v případě chyby, která má za následek přepsání vlastního programu. Jediná sběrnice tvoří úzké místo a snižuje tak přenosovou rychlost.



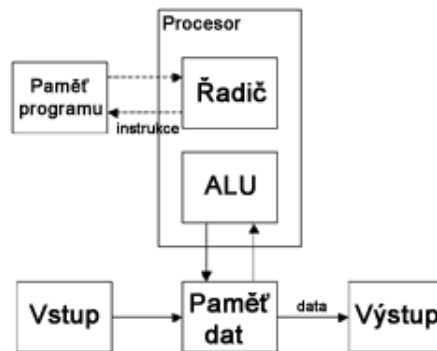
Obrázek 7 – Von Neumannovo schéma

3.1.2. Harwardské schéma – oddělená paměť programu a dat

Jedna z výhod této architektury je možnost použití různých typů (technologií) pro oddělené paměti. Během zápisu nemůže dojít k přepsání programu jako v předchozím případě a každá

paměť může mít svoji velikost v závislosti na adresovacích možnostech. Po dvou sběrnících lze k pamětem programu i dat přistupovat paralelně (současně).

Nevýhody jsou dvě. V první řadě nevyužitá paměť programu nemůže být použita pro data a naopak. Za druhé jsou vyšší nároky na řídicí jednotku při komunikaci po dvou sběrnících.



Obrázek 8 – Harwardské schéma

3.2. Podle instrukčního souboru

- **CISC** (Complex Instruction Set Computer)
- **RISC** (Reduce Instruction Set Computer)

3.2.1. CISC (Soubor s kompletní instrukční sadou)

Obsahuje instrukční soubor s takovými instrukcemi, které pod jedním operačním kódem vykonají složité operace s variabilitou různých adresovacích módů. Tato výhoda je vykoupena za cenu zpracování těchto instrukcí ve strojových cyklech. Strojový cyklus je tvořen několika takty oscilátoru, a tedy instrukce se vykonávají podstatně déle než u RISC. Řídicí obvody u CISC-architektury zabírají na čipu přibližně 60% místa, kdežto u RISC-architektury je to pouze 6-10%.

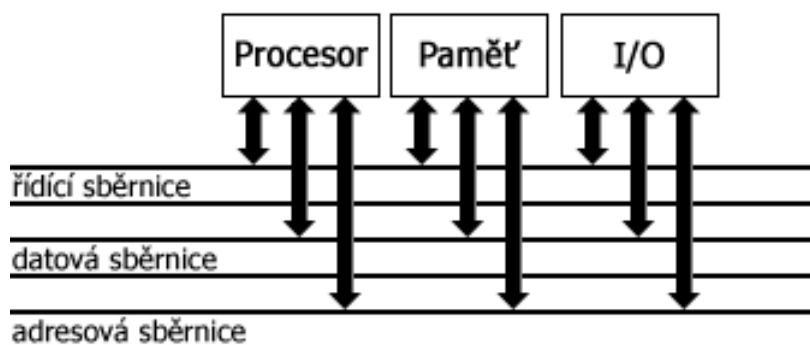
3.2.2. RISC (Soubor s redukovanou instrukční sadou)

Pokud si uvědomíme, že frekvence používání některých složitých instrukcí u CISC je tak malá, tedy nevyplatí se pro ně plýtvat plochou na čipu, pak je výhodnější nahradit je posloupností jednoduchých instrukcí z redukované instrukční sady. Instrukční sada obsahuje menší počet jednoduchých instrukcí, které se zpracovávají přímo v taktech oscilátoru. Díky tomu jsou řídicí obvody CPU jednodušší a zkracuje se doba zpracování instrukcí.

3.3. Sběrnice

Moderní počítače využívají tzv. sběrnice. Jejím úkolem je přenášet data a veškeré signály v rámci počítače mezi jeho jednotlivými částmi. Hlavními parametry sběrnice jsou:

- Šířka přenosu [bit] - počet bitů, které lze zároveň po sběrnici přenést
- Frekvence [Hz] - maximální frekvence, se kterou může sběrnice pracovat
- Rychlost (propustnost) [B/s] – počet Bytů přenesených za jednotku času.

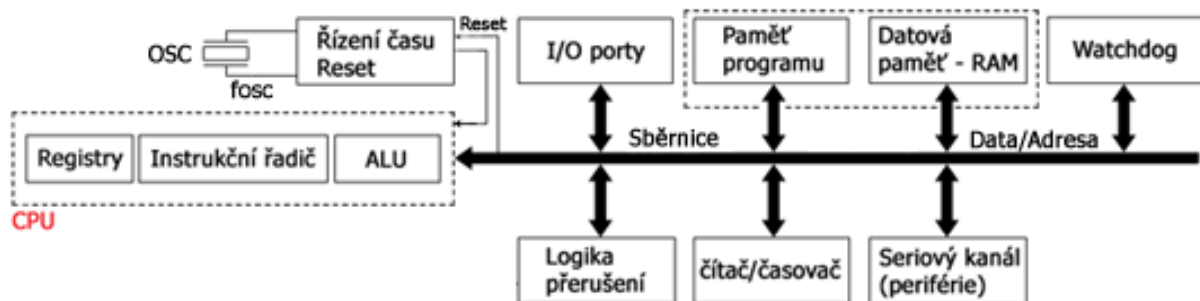


Obrázek 9 – základní typy sběrnice

3.4. Obecné schéma mikropočítače MCU

Obecné schéma mikropočítače popisuje základní bloky, které se nacházejí prakticky ve všech MCU. Všechny bloky jsou propojeny sběrnici, která přenáší data a adresy. Zdrojem frekvence

(f_{osc}) celého MCU je interní nebo externí oscilátor **OSC**. Blok **řízení času** zajistí úpravu frekvence f_{osc} nebo zajistí řasování pro komunikaci s externí pamětí. Další funkcí je zpracování signálu **reset**. **CPU**, centrální procesorová jednotka obsahuje aritmetickologickou jednotku **ALU**, uživatelské **registry** a **instrukční řadič** obsluhující instrukční soubor. ALU provádí aritmetickologické výpočty. Pro připojení ovládaného systému se využívají vstupně/výstupní porty – blok **I/O** (čidla, IO, atd.). Tento blok nedokáže ovládat výkonnostní prvky a pracuje pouze s digitálním signálem, pokud nejsou specializovány jinak. **Paměť programu a dat** slouží pro uložení zapsaného kódu a dílčích výpočtů. Velikost pamětí je dána strukturou MCU a šířkou sběrnice. **Watchdog** je blok zajišťující správný chod MCU, v případě chyby aktivuje signál reset. **Logika přerušeni** slouží k obsluze podprogramů, které využívají různé periférie. **Čítač a časovač** je velmi důležitá součást MCU, která zajišťuje generování časových intervalů nebo čítání interních či externích impulsů. Posledním neméně důležitým blokem je **sériový kanál**, přes který se připojuje MCU například k PC a dalším perifériím. Přes sériový kanál se programuje MCU.



4. Rodina mikropočítačů AVR

Novější procesory rodiny Atmel AVR se dají rozdělit do tří základních skupin:

- **ATtinyAVR** – 8 bitové MCU mají 1 až 8 KB paměti programu a základní instrukce (120).
- **ATmegaAVR** – 8/16 bitové MCU mají 4 až 256 KB paměti programu, 28.-100. pinová pouzdra, dále mají rozšířený instrukční soubor. Jsou podstatně vybavenější (připravené pro

složitější aplikace). Oproti ATtiny disponují JTAG rozhraním, které slouží pro ladění softwaru přímo v aplikaci

- **Xmega** – 8/ 16 bitové MCU s vyšším výkonem, nižší spotřebou a menšími rozměry. Typické aplikace pro využití tohoto MCU: audio systémy, zdravotnická zařízení, sítě, měření, řízení motorů a jakékoliv bateriově napájená zařízení.

Další podrobnější dělení lze provést dle aplikačního využití:

- AVR procesory pro automobilový průmysl
- AVR pro bateriové aplikace (řízené napájení)
- AVR s podporou CAN (průmyslová sběrnice)
- LCD AVR
- AVR řízení osvětlení
- AVR s podporou USB

Následující tabulka 1, zobrazuje stručný přehled některých typů MCU rodiny AVR se základními údaji. Tabulku se všemi parametry lze nalézt na stránkách firmy Atmel:

http://www.atmel.cz/dyn/products/devices.asp?family_id=607

Device	Flash (KB)	EEPROM (KB)	SRAM (KB)	I/O	F.max (MHz)	Vcc (V)	16 bit Timer	PWM ch.	Packages
ATxmega128A1	128	2	8	78	32	1.8 - 3.6	8	24	CBGA 100
ATxmega16A4	16	1	2	34	32	1.8 - 3.6	5	16	TQFP 44
ATxmega192A1	192	4	16	78	32	1.8 - 3.6	8	24	CBGA 100
ATxmega256A1	256	4	16	78	32	1.8 - 3.6	8	24	CBGA 100
ATxmega32A4	32	1	4	34	32	1.8 - 3.6	5	16	TQFP 44
ATxmega384A1	384	4	32	78	32	1.8 - 3.6	8	24	CBGA 100
ATxmega64A1	64	2	4	78	32	1.8 - 3.6	8	24	CBGA 100

Device	Flash (KB)	EEPROM (KB)	SRAM (KB)	I/O	F.max (MHz)	Vcc (V)	16 bit Timer	8 bit Timer	Watchdog	Packages
ATtiny13A	1	0.0625	64B+32r	6	20	1.8-5.5	--	1	Yes	SOIC (150mil) 8
ATtiny2313	2	0.125	128	18	20	1.8-5.5	1	1	Yes	SOIC (300mil) 20
ATtiny24	2	0.125	128	12	20	1.8-5.5	1	1	Yes	MLF (WQFN) 20
ATtiny25	2	0.125	128	6	20	1.8-5.5	--	2	Yes	SOIC (150mil) 8
ATtiny4	0.5	--	32	4	12	1.8-5-5V	1	--	Yes	
ATtiny43U	4	0.0625	256	16	8	0.7 - 1.8	--	2	Yes	SOIC (300mil) 20
ATtiny44A	4	0.25	256	12	20	1.8-5.5	1	1	Yes	VQFN 20
ATtiny45	4	0.25	256	6	20	1.8-5.5	--	2	Yes	SOIC (208mil) 8
ATtiny461A	4	0.25	256	16	20	1.8 - 5.5	1	2	Yes	SOIC (300mil) 20
ATtiny48	4	0.0625	256	24/28	12	1.8 - 5.5	1	1	Yes	TQFP 32
ATtiny5	0.5	--	32	4	12	1.8-5-5V	1	--	Yes	
ATtiny84	8	0.5	512	12	20	1.8-5.5	1	1	Yes	MLF (WQFN) 20
ATtiny85	8	0.5	512	6	20	1.8-5.5	--	2	Yes	SOIC (208mil) 8
ATtiny88	8	0.0625	512	24/28	12	1.8 - 5.5	1	1	Yes	TQFP 32

Device	Flash (KB)	EEPROM (KB)	SRAM (KB)	I/O	F.max (MHz)	Vcc (V)	16 bit Timer	8 bit Timer	Watchdog	Packages
ATmega1280	128	4	8192	86	16	1.8-5.5	4	2	Yes	CBGA100
ATmega128A	128	4	4096	53	16	2.7-5.5	2	2	Yes	TQFP 64
ATmega162	16	0.5	1024	35	16	1.8-5.5	2	2	Yes	TQFP 44
ATmega168PA	16	0.5	1024	23	20	1.8-5.5	1	2	Yes	TQFP 32
ATmega16A	16	0.5	1024	32	16	2.7-5.5	1	2	Yes	TQFP 44
ATmega325	32	1	2048	54	16	1.8-5.5	1	2	Yes	TQFP 64
ATmega3250	32	1	2048	69	16	1.8-5.5	1	2	Yes	TQFP 100
ATmega32A	32	1	2048	32	16	2.7-5.5	1	2	Yes	TQFP 44
ATmega48PA	4	0.25	512	23	20	1.8-5.5	1	2	Yes	TQFP 32
ATmega64	64	2	4096	54	16	2.7-5.5	2	2	Yes	TQFP 64
ATmega640	64	4	8192	86	16	1.8-5.5	4	2	Yes	CBGA 100
ATmega644	64	2	4096	32	20	1.8-5.5	1	2	Yes	TQFP 44
ATmega645	64	2	4096	54	16	1.8-5.5	1	2	Yes	TQFP 64
ATmega88PA	8	0.5	1024	23	20	1.8-5.5	1	2	Yes	TQFP 32
ATmega8A	8	0.5	1024	23	16	2.7-5.5	1	2	Yes	TQFP 32

Tabulka 1 – Stručný přehled rodiny mikroprocesorů AVR

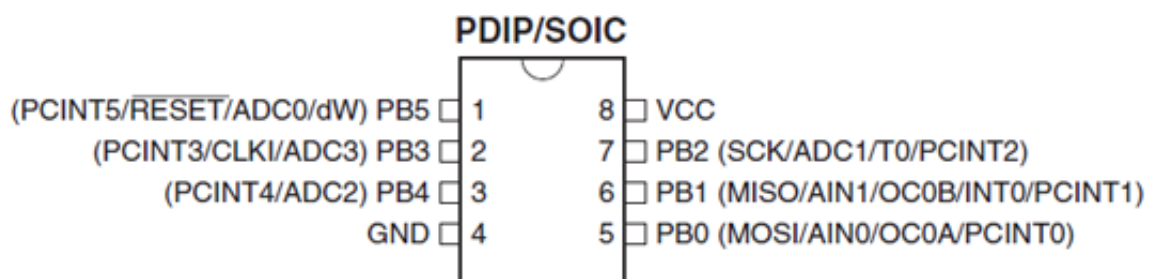
5. MCU ATtiny 13 a ATtiny 2313

5.1. Primární parametry

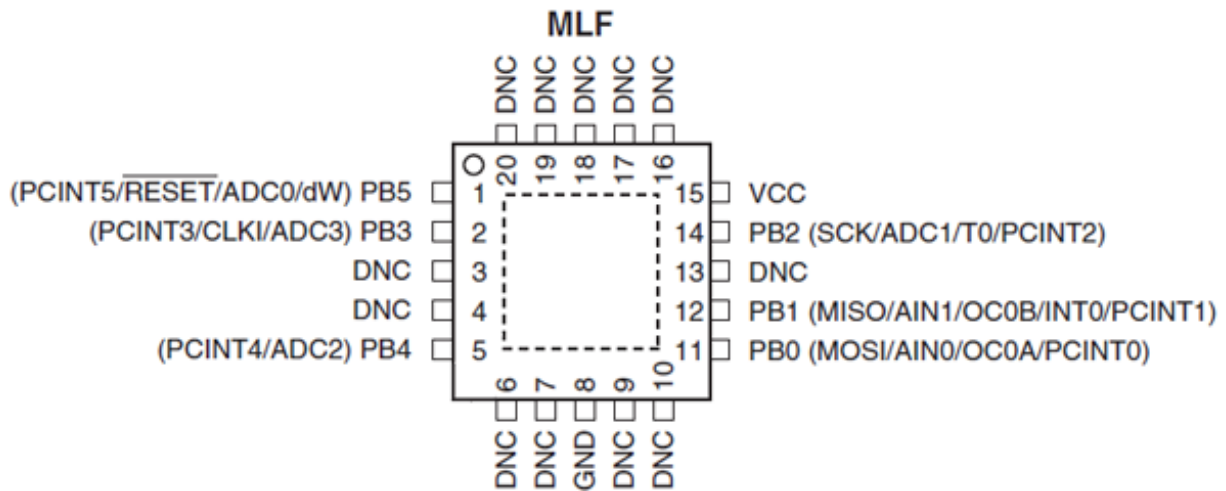
Hned na začátku se podíváme na základní parametry (vlastnosti) mikropočítačů ATtiny 13 a 2313, kterými se budeme po zbytek publikace zabývat. Na začátek je dobré připomenout, že pracujeme s procesory typu RISC s Harwardskou architekturou.

5.1.1. ATtiny 13

- 8 pinové pouzdro (obrázek 10) nebo 20 pinové pouzdro (obrázek 11)
- 8 bitová architektura, 120 instrukcí
- 9 zdrojů přerušení
- 1k FLASH (10 000 R/W cyklů, ISP), 64 bytů EEPROM, 64 bytů SRAM
- programování FLASH paměti.
- softwarové ovládání běhu programu, zásah do paměti
- 8 bitový čítač/časovač s možností porovnání se dvěma hodnotami, PWM
- analogový komparátor (možnost porovnávat s interní referencí 1,1V)
- 9,6MHz resp. 4,8MHz interní oscilátor - OSC
- 4 kanálový 10 bitový ADC +-2LSB
- napájení 1,8 - 5,5V



Obrázek 10 – 8 pinové pouzdro SOIC 8 mikropočítače tiny13

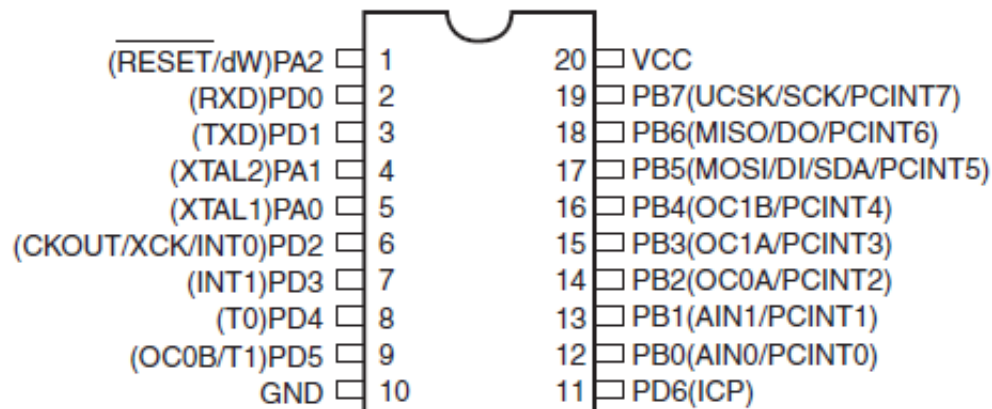


Obrázek 11 - 20 pinové pouzdro MLF 20 mikročítače tiny13

5.1.2. ATtiny 2313

- 20 pinové pouzdro (PDIP, SOIC, MLF)
- 8 bitová architektura se 120 instrukcemi
- 2K FLASH (10 000 R/W cyklů, ISP), 128 bytů EEPROM, 128 bytů SRAM
- 9 zdrojů přerušení
- 18 I/O pinů – vstupně / výstupní porty
- Zajištění komunikace: USI (universal serial interface). Full Duplex USART (The Universal Synchronous and Asynchronous serial Receiver and Transmitter)
- 1x 8 bitový čítač s PWM a porovnáváním s definovanou hodnotou
- 1x 16 bitový čítač s PWM, porovnáváním a možností zápisu hodnoty na výstup, což umožní měřit frekvence, či doby trvání akcí.
- 4x High Speed PWM
- analogový komparátor - má možnost srovnávat s referencí 1,1V
- programování - paralelní, sériové přes ISP
- napájení 1,8 - 5,5V
- zpětná kompatibilita s MCU ATtiny 13 je zajištěna – podporuje funkce

PDIP/SOIC

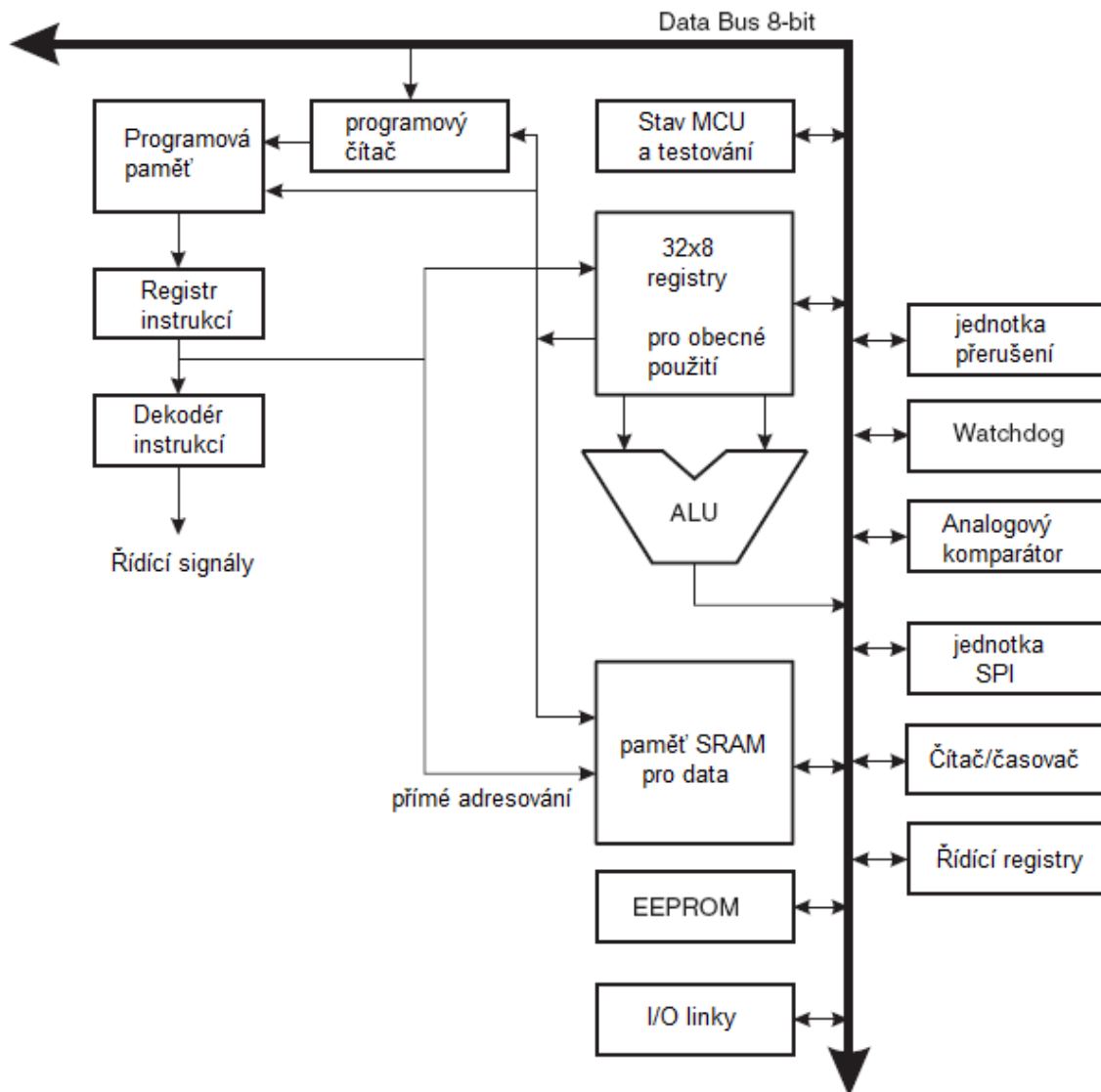


Obrázek 12 - 20 pinové pouzdro SOIC 20 mikropočítače tiny2313

5.2. Architektura – blokové schéma

V kapitole 3.4 jsme se seznámili s obecným schématem MCU. Nyní budeme téma konkretizovat a popíšeme si obecné schéma typického procesoru AVR (viz). Bloky které byly popsány v kapitole 3.4 dále nemá cenu rozebírat.

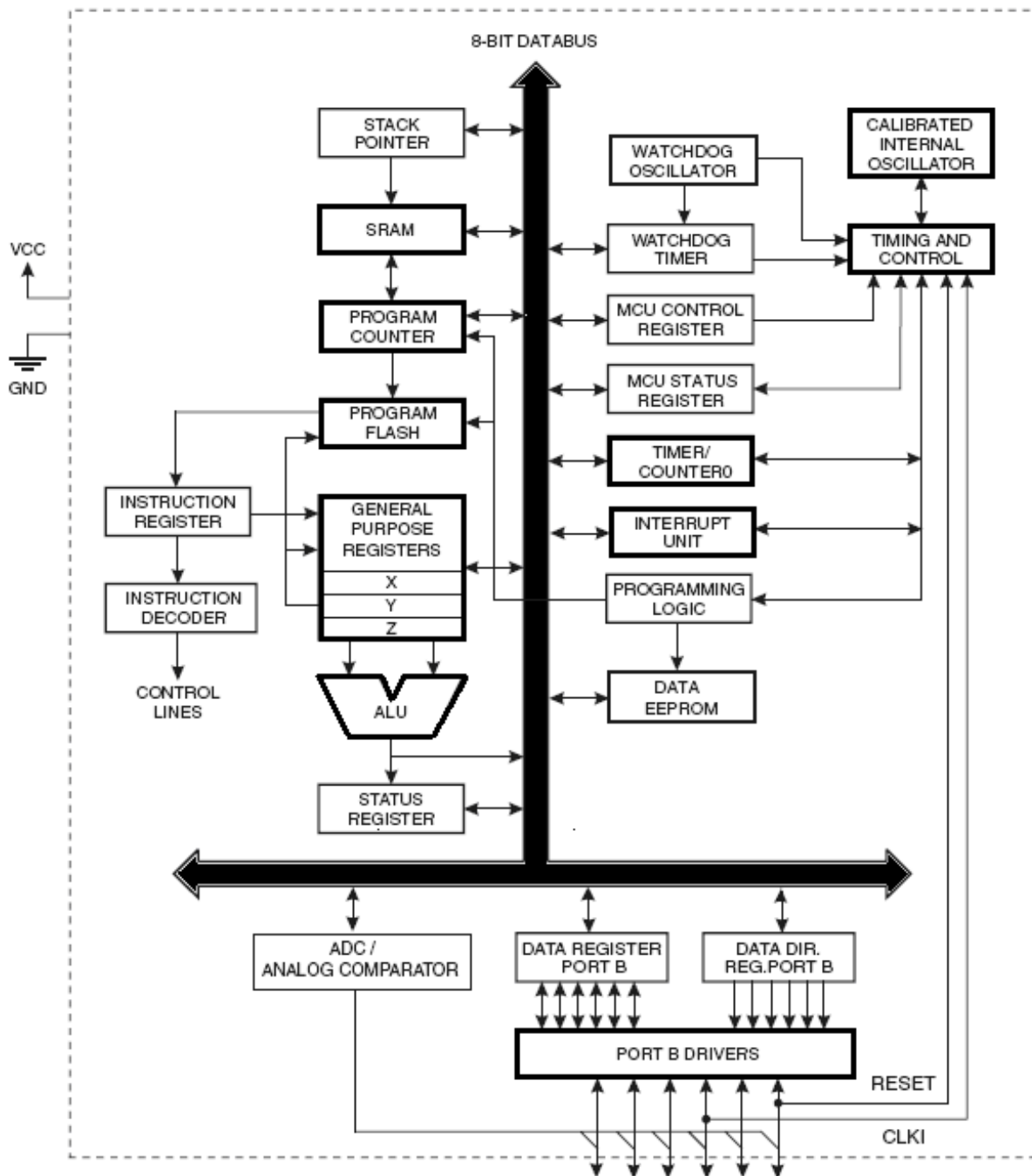
Základem je aritmetickologická jednotka, která je přímo propojena s 32 registry. Ty mohou obsahovat data i adresy a přístup k nim trvá jeden hodinový cyklus. Posledních šest registrů (X, Y, Z) lze využít ve dvojici jako ukazatelé adresy pro nepřímé adresování. Paměti dat a programu jsou doplněné energeticky nezávislou pamětí EEPROM (uchování dat, nastavení, atd.). Registr instrukcí a dekodér instrukcí se starají o zpracování instrukcí z instrukčního souboru, které jsou součástí kódu. Programový čítač je speciální registr, do kterého se zapisují adresy vykonávaných instrukcí, které tvoří uživatelem vytvořený program. Analogový komparátor poslouží jako jednoduchý digitálně analogový převodník. Porovnává hodnotu na vstupu komparátoru s referenční hodnotou MCU. Blok SPI, umožňuje programovat procesor, který je umístěný přímo v aplikaci (nemusí se vytahovat z patice a speciálně programovat). Poslední blok obsahuje řídicí registry, které obsahují nastavení všech zařízení MCU.



Obrázek 13 – Obecné schéma mikropočítače AVR

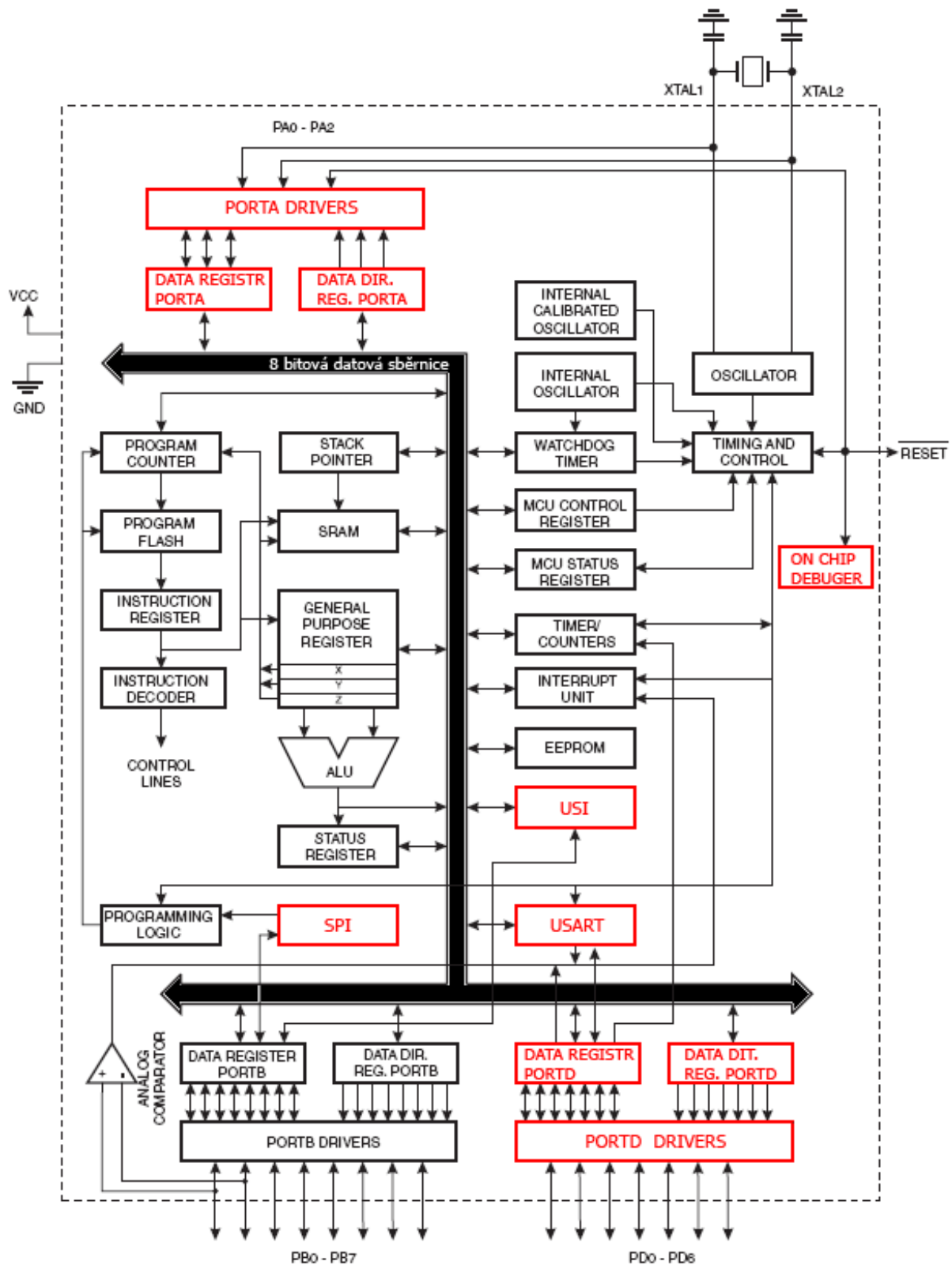
5.2.1. Blokové schéma tiny13 a tiny2313

V této kapitole si popíšeme konkrétní schéma obou MCU. Rozdíly mezi nimi jsou zvýrazněny červenou barvou. Obě schémata jsou si velmi podobná. ATtiny 2313 disponuje větším počtem vstupně výstupních portů (pinů) a větším počtem programovacích a komunikačních rozhraní (SPI, USI, USART). V následujících kapitolách si podrobně popíšeme jednotlivé bloky obou konkrétních blokových schémat (viz obrázek 14 a obrázek 15).



Obrázek 14 – blokové schéma MCU ATtiny 13

Důležitou částí obou procesorů je přímé propojení jednotky ALU a registrů pro obecné použití (32 registrů). To zajišťuje přímý přístup k registrům v jednom časovém cyklu interního oscilátoru (popřípadě externího).



Obrázek 15 – blokové schéma MCU ATtiny2313

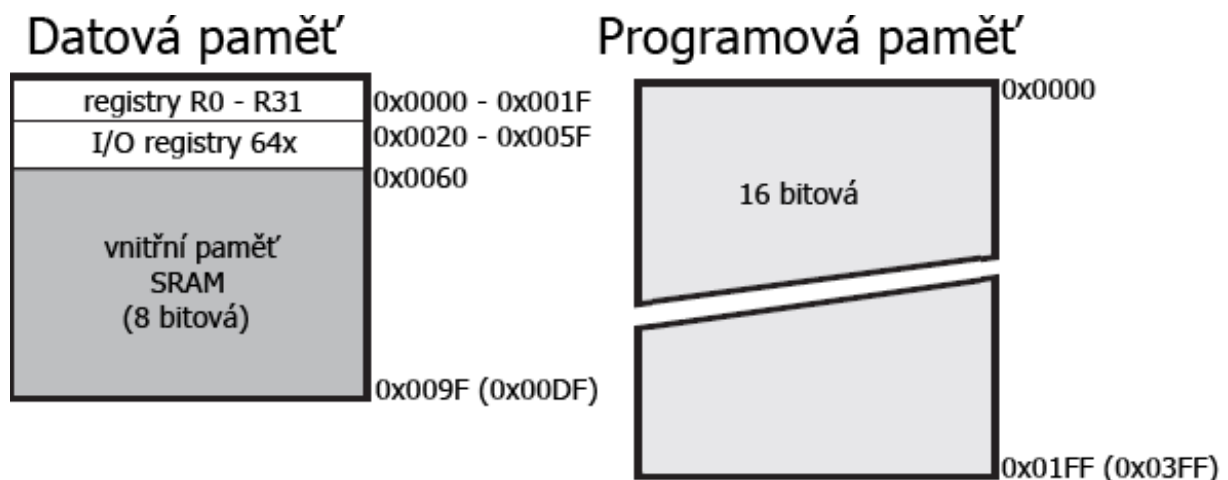
6. Paměti MCU tiny 13/2313

6.1. Paměť programu MCU tiny 13/2313

Jedná se o tzv. In-System Programmable (ISP) Flash paměť s kapacitou 1KB (2KB) (viz. 5.1). Základní funkcí paměti programu je uchovávat vytvořený program – instrukce, programový kód. Protože všechny instrukce jsou buď 16 bitové (čtyři instrukce jsou 32bitové) je paměť je organizována po 16 bitových slovech 512x16 (1024x16). Čtení instrukcí z paměti využívá tzv. překrývání instrukcí - dvoustupňové pipeline. Zatím co jedna instrukce se vykonává, další instrukce je přesouvána z programové paměti. Tento model umožňuje zpracování instrukcí v jednom hodinovém taktu. Paměť je schematicky zobrazena na obrázku 16. In-System Programmable (ISP) umožňuje programovat paměť sériově přímo v aplikaci.

6.2. Datová paměť MCU tiny 13/2313

Datová paměť FLASH je rozdělena do tří bloků (obrázek 16). První blok tvoří registry pro obecné použití, druhý registry vstupně výstupních portů a poslední blok je vnitřní paměť SRAM (přístup k paměti SRAM trvá dva strojové cykly).



Obrázek 16 – Programová a datová paměť ATtiny

32 registrů R0 až R31 jsou k dispozici pro všeobecné využití. Jak již bylo jednou řečeno, tyto registry jsou přímo propojeny s aritmeticko-logickou jednotkou ALU. Tím je zajištěno zpracování instrukcí během jednoho hodinového cyklu z interního oscilátoru (to MCU s úplnou instrukční sadou CISC neumí). Do registrů R0 až R15 nelze přímo zapsat konstantu pomocí instrukce LDI. Tento hendikep je dán odlišným adresováním těchto registrů. Posledních 6 registrů můžeme ve dvojici použít jako ukazatele 16ti bitové adresy pro nepřímé adresování paměti dat. Tyto registry označované písmeny X, Y a Z dovolují libovolné ukládací operace. AVR architektura má 5 adresovacích módů pro paměť dat:

- přímé adresování
- nepřímé adresování s posunem (6bitový posun v registrech Y a Z)
- nepřímé adresování (využívá pouze registry X, Y, Z)
- nepřímé adresování s dekrementací ukazatele adresy před zpracováním instrukce (X, Y, Z)
- nepřímé adresování s inkrementací ukazatele adresy po zpracování instrukce (X, Y, Z)

7bit		0bit	
	R1		0x00
	R2		0x01
	R3		0x02
	...		
	...		
	R14		0x0E
	R15		0x0F
	R16		0x10
	R17		0x11
	...		
	R26	X	0x1A X - registr horních 8 bitů (byte)
	R27		0x1B X - registr dolních 8 bitů (byte)
	R28	Y	0x1C Y - registr horních 8 bitů (byte)
	R29		0x1D Y - registr dolních 8 bitů (byte)
	R30	Z	0x1E Z - registr horních 8 bitů (byte)
	R31		0x1F Z - registr dolních 8 bitů (byte)

R0 - R15 nelze využít pro uložení konstanty přímo do registru (LDI)

Obrázek 17 – 32 registrů pro všeobecné použití

6.2.1. EEPROM datová paměť

Za zmínku určitě stojí paměť EEPROM, která poslouží k uložení dat v případě, že o ně nebudeme chtít přijít odpojení aplikace od napájení (např. uložení hesel, nastavení aplikace atd.). Do paměti lze zapisovat a číst s životností 100 000 cyklu (tento údaj udává životnost paměti nejenom v případě AVR ATtiny). Celková komunikace s EEPROM je definována pomocí tří registrů:

- **EECR** - Řídící registr (EEPROM Control Registr) – Obsahuje nastavení módu: čtení, zápis, čtení a zápis v jednom kroku, EEPROM Program Enable, přerušení od EEPROM atd..
- **EEARL** – Adresový registr (EEPROM Address Registr) – udává adresu pro uložit/čtení dat do/z paměti EEPROM (64Bytes/128Bytes).
- **EEDR** – Datový registr (EEPROM Data Registr) – Pro operaci čtení obsahuje registr data, která se vyčetla z paměti EEPROM z adresy dané registrem EEARL. Pro operaci zápis obsahuje data, která se mají zapsat do paměti na adresu danou registrem EEARL.

Příklady kódu v assembleru pro přístup k paměti EEPROM naleznete ve sbírce příkladů.

7. Resetovací obvod MCU tiny13/2313

Oba mikroprocesory mají čtyři zdroje, které mohou vygenerovat signál reset.

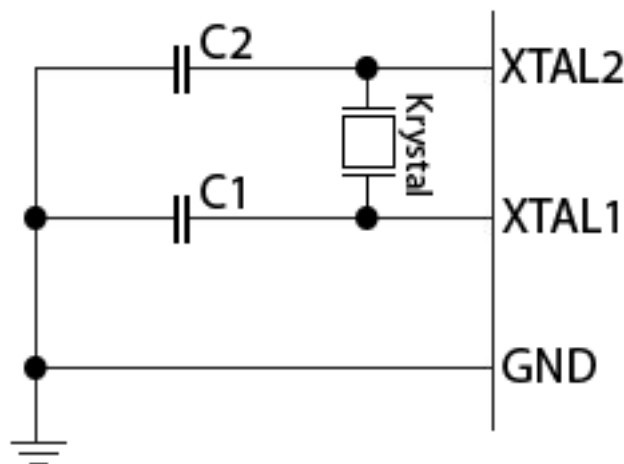
- **Watchdog** – pokud obvod watchdog zjistí chybu v chodu programu, pak provede reset MCU (program začne běžet znovu od začátku)
- **POR (Power On Reset)** – automatický reset po připojení k napájení
- **Externí reset** – resetuje MCU v případě, kdy na (invertovaný) resetovací vstup přivedeme "log 0" na dobu delší jak 50ns
- **Prahový reset** - při snížení napájecího napětí pod povolenou mez. Musí být spuštěno hlídání této úrovně.

8. Časování MCU tiny13/2313

Jednou ze základních vlastností MCU (CPU) je kmitočet. Jednotlivé instrukce, které MCU vykonává, potřebují pro své zpracování určitý čas. Ze známé rovnice $f = 1/T [Hz]$ vyplývá, že právě frekvence MCU určuje rychlost, kterou MCU zpracovává instrukce.

Zdroje hodinového signálu (frekvence) máme dva:

- **Interní RC oscilátor** – jedná se o integrovaný zdroj hodinového taktu (není součástí každého MCU)
- **Externí zdroj** – např. piezoelektrický krystal. Připojuje se na piny MCU označené XTAL1, XTAL2 (obrázek 18). Princip piezoelektrického krystalu je náplní předmětu elektronika. K MCU ATtiny 13 nelze externí oscilátor připojit.



Obrázek 18 – zapojení externího zdroje hodinového signálu

8.1. Interní zdroj hodinového signálu

Interní RC oscilátor - Kvůli minimalizaci počtu externích součástek nabízejí AVR MCU možnost generovat hodinový signál vestavěným laditelným RC oscilátorem. Protože oscilátor není kalibrován, ale pouze laditelný, je nutné provést přesné měření frekvence. Ladění interního oscilátoru se provádí pomocí hodnoty uložené registru OSCCAL. Během resetu se hodnota aktualizuje a RC oscilátor se automaticky kalibruje. Pokud je registr OSCCAL

(Oscilátor Calibration Registr) nulový pak se vybere nejnižší možná frekvence. Naproti tomu hodnota 0x7F zajistí výběr nejvyšší hodnoty frekvence pro oscilátor.

Frekvenci interního oscilátoru určují bity CKSEL. U MCU ATtinny13 je frekvence 4,8 MHz nebo 9,6 MHz. U ATtinny2313 je frekvence 4 MHz nebo 8 MHz. Reálně se frekvence MCU nastavuje při programování programové a paměti a Fuse bitů.

8.2. Taktování (časování) operandů

Operandy se zpracovávají tak, že během jednoho taktovacího cyklu se přivedou oba zdrojové operandy z pracovních registrů, uskuteční se potřebná operace a výsledek se uloží zpátky do registrů. Při úspěšném provedení je tedy potřebný čas jeden takt. Jedná se výhradně o operace typu registr - registr, na kterých je AVR-architektura založena. Složitější instrukce trvají déle a je potřeba k jejich vykonání větší počet taktovacích cyklů (viz instrukční soubor).

8.3. Před-dělička systémového času

Pokud dané aplikaci nevyhovuje frekvence interního RC oscilátoru, lze pro její snížení využít integrovanou před-děličku. Dělicí poměr lze nadefinovat v CLKPR řídicím registru a aplikuje se na všechny zdroje hodinového signálu. Kombinací bitů CLKPSx se nastavuje dělicí poměr s druhou mocninou viz tabulka 2.

	b7	b6	b5	b4	b3	b2	b1	b0
CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0

Obrázek 19 – řídicí registr CLKPR (Clock Pre-scale Register)

CLKPRS3	CLKPRS2	CLKPRS1	CLKPRS0	Dělicí poměr
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
⋮	⋮	⋮	⋮	⋮
1	0	0	0	256

Tabulka 2 – řídicí bity CLKPSx před-děličky hodinového signálu

9. Blok čítače/časovače

Blok čítače/časovače umožňuje čítání vnějších událostí, měření a generování časových intervalů nezávisle na MCU. Nezávislost na MCU znamená, že procesor vykonává hlavní program a současně s ním (nezávisle) se vykonává měření čas. Čítač/časovač může pracovat ve dvou módech:

- **čítač** – čítá impulzy (digitální signál) z I/O portů (např. impulzy snímače otáček jízdního kola). Frekvence čítaného signálu musí být menší nebo rovna f_{osc} . V opačném případě by se všechny impulzy nestihly zaznamenat. Impulzy jsou čítány z portu označeného T0 (pin PB0).
- **časovač** – čítání impulsů vnitřního RC oscilátoru – z přesně definovaného signálu (f_{osc}) lze lehce vyjádřit čas, který chceme využít například pro měření nebo k jeho generování.

$$T = 1/f_{osc}$$

Mikroprocesory AVR mají v sobě zabudované 8 bitové a 16 bitové čítače/časovače.

Samozřejmě každý AVR MCU nemá stejný počet a stejné typy čítačů/časovačů. 16 bitový Č/Č není pouhým rozšířením 8 bitové verze, navíc ještě obsahuje záchytný a komparační registr. Č/Č využívají pro korekci frekvence před-děličku času a jeho rychlost je tedy odvozena od f_{osc} .

9.1. 8 bitový čítač/časovač

Tento jednodušší blok obsahuje řídicí část, která koordinuje komunikaci mezi řídicím registrem (TCCR), registrem příznaku přerušení (TIFR), Registrem maskování (TIMSK) a registrem obsahujícím hodnotu Č/Č (TCNT).

- **TCCR0A** (Timer/Counter0 Control Register A) – řídicí registr nastavuje mód Č/Č (normal, PWM, fast PWM, CTC) a vlastnosti výstupních bitů Č/Č (OCRA,OCRB) viz obrázek 20.
- **TCCR0B** (Timer/Counter0 Control Register B) – definuje dělicí poměr zdroje času pro Č/Č (Č/Č stop, 1, 8, 64, 256) viz obrázek 21.
- **TIFR** (Timer/Counter Interrupt Flag) – tento registr obsahuje příznaky (Flags) přerušení. V případě, že dojde k přerušení od Č/Č nastaví se dle zdroje přerušení příznak v tomto registru. Pokud je přerušení pro daný příznak povoleno, dojde k jeho obsluze. Podrobný popis viz obrázek 22.
- **TIMSK** (Timer/Counter Interrupt Mask) – tento registr povoluje přerušení Č/Č v daném módu. Pokud by přerušení Č/Č nebylo povoleno, pak by i za předpokladu splnění daných podmínek, MCU na žádost o přerušení od Č/Č nereagoval, viz obrázek 23.
- **TCNT0** (Timer/Counter Register) – tento 8. bitový registr obsahuje hodnotu Č/Č. Registru TCNT0 umožňuje přímé operace zápisu a čtení. Modifikace tohoto registr v případě že Č/Č běží, může způsobit špatné porovnání hodnoty mezi registry TCNT0 a OCR0x.
- **OCR0x** (Output Compare Register A/B) – 8. bitový výstupní registr obsahuje data, která jsou nepřetržitě porovnávána s registrem Č/Č (TCNT0). V řídicím a stavovém registru je nastaveno jaká bude reakce v případě shody OCR0x a TCNT0.

	b7	b6	b5	b4	b3	b2	b1	b0
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

COM0Ax - tyto bity řídí chování výstupního porovnávacího pinu OC0A. Pokud nejsou oba bity nastaveny ("0"), pak je pin OC0A odpojený a port pracuje v normál. módu, pokud je pin OC0A připojen, pak význam bitů COM0A závisí na bitech WGMx.

COM0Bx - naprosto stejná funkce jakov COM0A1, ale pro pin OC0B.

WGM x - bity v kombinaci s bitem WGM 02 (v TCCR0B) definují jakým způsobem bude Č/Č rpcovat - Normal, PWM, c fast PWM, CTC.

Obrázek 20 – TCCR0A řídicí registr 8 bitového Č/Č

	b7	b6	b5	b4	b3	b2	b1	b0
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

FOC0x - byty jsou vždy nastaveny do log "0", využívají se pouze pokud bity WGMx definují jiný než PWM mód. Pokud jsou bity nastaveny do log "1", pak se CTC mód vynuceně přepne do PWM.

WGM02 - se využívá v kombinaci s WGM01 a WGM00 viz registr TCCR0A.

CSx - definují dělicí poměr zdroje času MCU. (No Clock ,1 ,8 ,64 ,256).

Obrázek 21 – TCCR0B řídicí registr 8 bitového Č/Č

	b7	b6	b5	b4	b3	b2	b1	b0
TIFR	TOV1	OCF1A	OCF1B	-	ICF1	OCF0B	TOV0	OCF0A

ICF1 - bit je nastaven pokud dojde na pinu ICP1 k zachycení události

TOV1(0) - bit je nastaven pokud dojde k přetečení čítače/časovače1(0)

OCFxA(xB) - bit je nastaven pokud dojde ke shodě mezi Č/Č1(0) a OCR1A (OCR0B)

Obrázek 22 – TIFR registr

	b7	b6	b5	b4	b3	b2	b1	b0
TIMSK	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A

OCIExA(xB) - povolení přerušení v případě CTC módu (Timer/Counter0(1) Output Compare Match A(B) Interrupt Enable)

TOIE0 - povolení přerušení od Č/Č v případě jejich přetečení (Timer/Counter0 Overflow Interrupt Enable)

ICIE1 - povolení přerušení, pokud dojde k zachycení události na pinu ICP1 (Input Capture Interrupt Enable)

Obrázek 23 – TIMSK registr určuje povolení přerušení módů Č/Č

Mode	WGM2	WGM1	WGM0	Mód Čítače/Časovače	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX = 0xFF
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM = 0x00
2	0	1	0	CTC	OCRA	Immediate	MAX = 0xFF
3	0	1	1	Fast PWM	0xFF	TOP	MAX = 0xFF
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM = 0x00
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	TOP	TOP

Tabulka 3 – Módy 8 bit. čítače/časovače 0 (PWM, CTC, Normal)

Módy čítače/časovače se nastavují pomocí bitů WGMx (v registrech TCCT0A, TCCR0B). Č/Č může pracovat v pěti módech. Jejich nastavení definuje tabulka 3.

9.2. 16 bitový čítač/časovač

Šestnácti bitový čítač/časovač je rozšířenou variantou výše uvedeného modulu. Jak již bylo uvedeno, obsahuje navíc záchytný registr a dva komparační registry. Počet módu, které může 16. bitový Č/Č využít je zobrazen v tabulce 4. Je označován jako Č/Č 1.

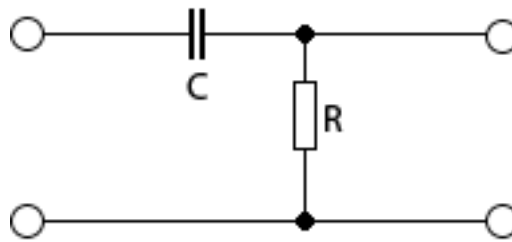
WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X at	TOV1 Flag Set on
0	0	0	0	Normal	0xFFFF	Immediate	MAX
0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
0	1	0	0	CTC	OCR1A	Immediate	MAX
0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
1	1	0	0	CTC	ICR1	Immediate	MAX
1	1	0	1	(Reserved)	–	–	–
1	1	1	0	Fast PWM	ICR1	TOP	TOP
1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Tabulka 4 - Módy 16 bit. čítače/časovače 1 (PWM, CTC, Normal)

9.3. PWM (Pulsně šířková modulace)

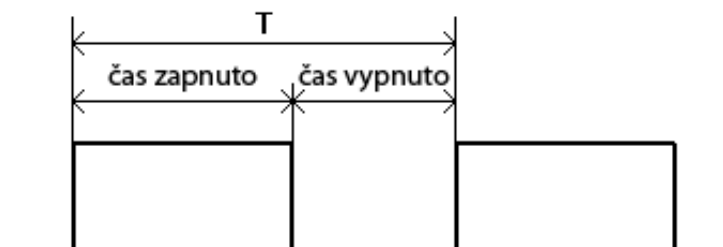
Pulsně šířková modulace, neboli PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového digitálního signálu. Jako dvouhodnotová veličina může být použito například napětí, proud, atd. Signál je přenášen pomocí **střídy**. Pro demodulaci takového signálu stačí dolnofrekvenční propust. Kombinace PWM modulátoru a dolnofrekvenční propusti bývá rovněž využívána jako levná náhrada D/A převodníku.

- **Dolní propust** – lineární filtr, který nepropouští signály vyšších frekvencí. Nejjednodušší zapojení zobrazuje obrázek 24 a maximální frekvence, kterou filtr propustí se spočítá podle rovnice $f_m = 1/2\pi RC$.



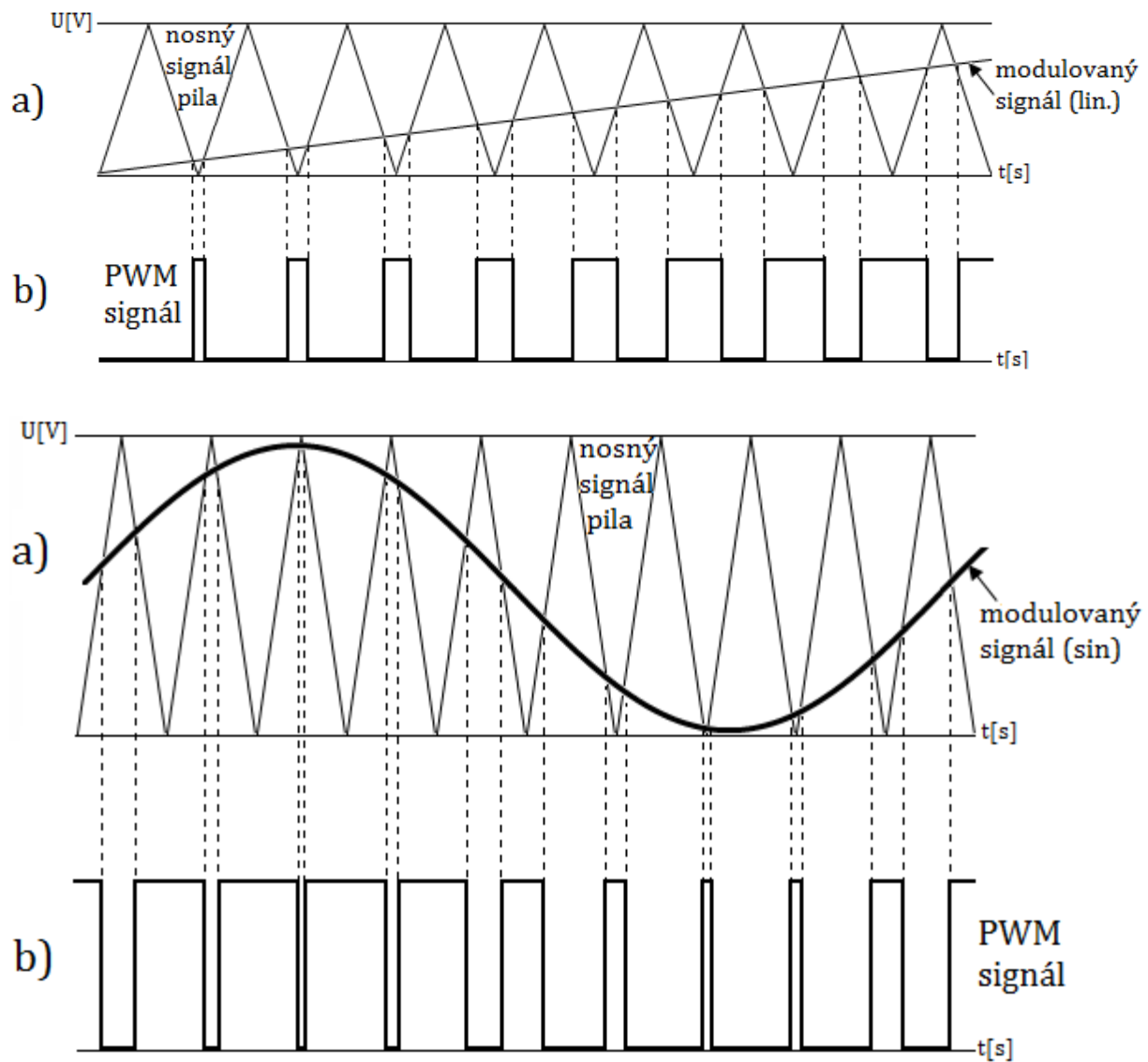
Obrázek 24 – Dolní propust (RC obvod)

- **Střída** – definuje se pro periodický signál. Periodický signál mění svou hodnotu mezi dvěma stavy (zapnuto, vypnuto). Poměr časů jednotlivých stavů definuje střidu signálu. Pokud se uvádí střída ve tvaru např 1:1, je tím myšleno, že obě dvě úrovně signálu trvají stejně dlouho. Pokud je střída udána v procentech, myslí se tím doba trvání úrovně „zapnuto“ vůči celkové periodě signálu. (0% až 100%, 50%).



Obrázek 25 – Střída periodického signálu

Popíšeme si nyní obrázek 26, na kterém je PWM modulace demonstrována. Představme si, že chceme s využitím PWM přenést analogové napětí (sinusový nebo jiný signál). Modulátor srovnává modulační signál (analogové napětí) s nosným signálem tvořeným střídavým napětím konstantní amplitudy (velkosti) a frekvence, jehož průběh je trojúhelníkový nebo pilový. Modulační signál musí mít podstatně nižší frekvenci než nosný signál. Pokud je okamžitá hodnota modulačního (modulovaného) signálu nižší (vyšší) než okamžitá hodnota nosného signálu, modulátor přepne do stavu zapnuto, v opačném případě do stavu vypnuto.



Obrázek 26 – PWM modulace, nosný signál trojúhelník (modulovaný signál lineární a sinusový)

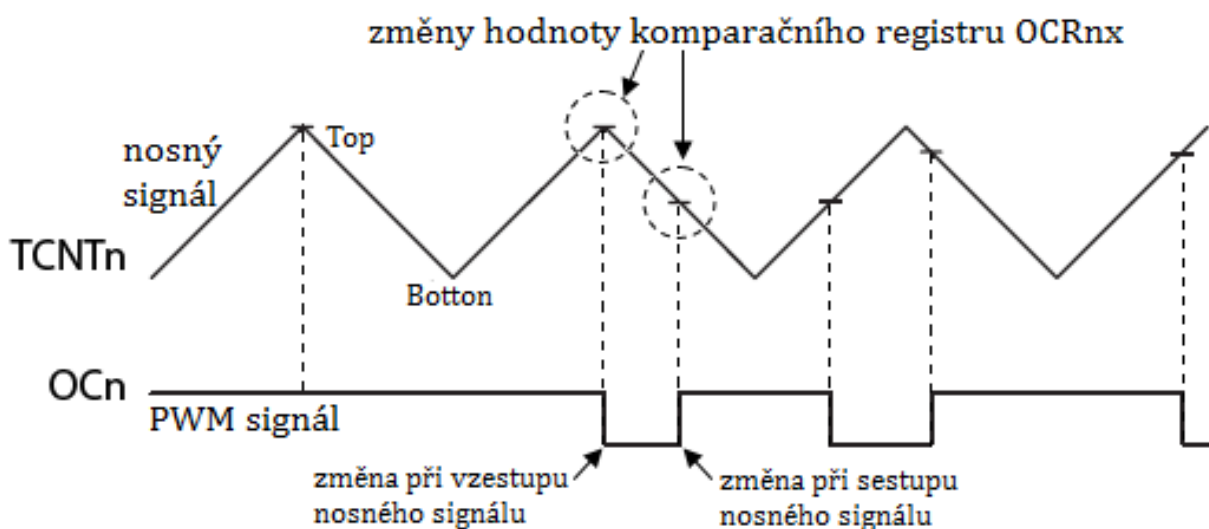
9.4. Funkce PWM u procesoru AVR

AVR procesory umí pracovat s několika módy pulsně šířkové modulace. Nejjednodušší variantou je **Fast PWM mód**. Tento režim je specifický tím, že čítá ode dna (Bottom) do maxima (Top) a poté se vrátí zpět ke dnu. Jedná se tedy trojúhelníkový nosný signál. Velikost maxima určuje přesnost PWM (8,9,10 bitový). Princip Fast PWM modulace je dobře patrný z obrázku 27.



Obrázek 27 – Fast PWM (Attiny2313, ATTiny13)

Druhý mód, který zmíníme je **fázově korigovaný mód**. Tento režim poskytuje oproti fast PWM 2x větší rozlišení. Je to dáno tím, že čítá nejenom nahoru, ale také ke dnu (Trojúhelníkový nosný signál). Velikost maxima opět určuje přesnost PWM. Při čítání nahoru je výstupní registr OCn vynulován pokud dojde ke shodě hodnot TCNTn a OCRnx. Při čítání směrem dolů (druhá fáze) je vývod nastaven opět při rovnosti registrů. To má za následek snížení pracovní frekvence (prodloužení periody), ale umožňuje to zase přesnější nastavení střídů. Grafické vyjádření tohoto módu PWM naleznete na obrázku 28.

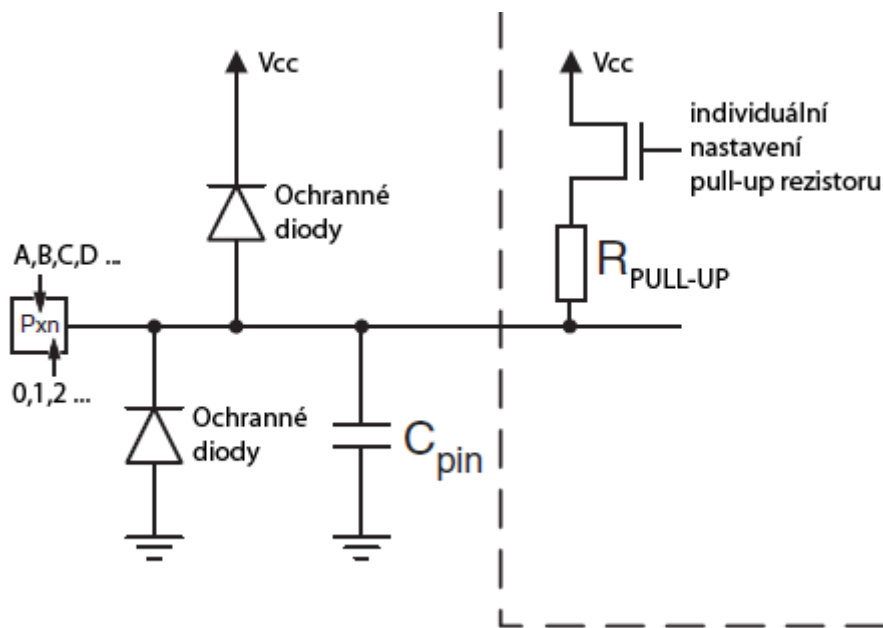


Obrázek 28 –Fázově korigovaný PWM mód (Attiny2313, ATTiny13)

10. Vstupně/výstupní (I/O) porty

Vstupně / výstupní porty (piny) slouží k připojení zařízení, která chceme řídit (jiné MCU, drivery zařízení, logické obvody, atd.), nebo z kterých chceme informace přijímat (např. čidla, jiné MCU). **MCU pracují s digitálním signálem**, pokud nezpracovávají definovaný analogový signál (analogový komparátor).

Všechny porty MCU AVR mohou pracovat jako vstupní nebo výstupní. Každému pinu je možné individuálně přiřadit pull-up rezistor s neměnnou hodnotou (často 10K Ω) a konstantním zdrojem napětí. Pull-up (vytáhnout) rezistor slouží k udržení vysoké úrovně na daném pinu. Dále všechny piny jsou chráněny diodami vůči napájecímu napětí a zemi. Zda je Pull-up defaultně přiřazen lze zjistit měřením na daném pinu (napěťová úroveň +5V).



Obrázek 29 – Struktura I/O AVR MCU (Hardware)

Piny portů jsou označovány symboly P xn (B, C, D, ...). Písmeno „ x “ je zástupný znak za označení portů (A,B,C,D), malé písmeno „ n “ je zástupný znak za pořadové číslo konkrétního pinu (bitu) daného portu (I/O). Jednodušší MCU, s kterými budeme pracovat, jsou v zásadě

vybaveny až třemi obousměrnými bránami (nejsou vždy 8bitové). Řada MCU ATmega může obsahovat až šest 8bitových obousměrných portů.

Porty mohou být v režimu výstupu zatíženy v „log 0“ proudem maximálně 20 mA, to umožní přímo napájet LED diodu.

Každý port je jednoznačně určen třemi adresami, které přesně definují funkci portu a obsahují jeho data. První z nich jsou určeny pro data registr PORTx (zápis/čtení), druhé pro směrový registr DDRx (zápis/čtení) a poslední adresy, jsou určeny pro vstupní piny portu PINx (pouze pro čtení). S adresami jsou propojeny bity DDxn, PORTxn a PINxn.

- **DDRx (DDxn)** – DDxn pin v registru DDRx definuje směr (vstupní pin/výstupní pin) tohoto pinu. Pokud je pin DDxn nastaven na hodnotu „log 1“, pak daný Pxn funguje jako výstupní pin. V opačném případě se bude Pxn chovat jako vstup.
- **PORTxn** – pokud je pin nastaven jako vstupní a bit PORTxn obsahuje logickou úroveň „1“ (high), pak je aktivován Pull-up rezistor. Deaktivace Pull-up rezistoru se provede po zapsání „log 0“ na pin PORTxn.
- **PINxn** – jednotlivé bity vstupního registru obsahují vstupní data, která je zapotřebí vyčítat v určitém řasovém harmonogramu (synchronizace s hodinovým signálem). Při změně vstupní hodnoty je potřeba počítat se zpožděním způsobeným synchronizací ($1/2$ až $1\frac{1}{2}$ periody vnitřních hodin). Proto je vhodné hned po změně signálu zařadit prázdnou instrukci NOP (získáme potřebný čas).

10.1. Přístup k portům (bránám) MCU

AVR a řada dalších MCU umožňuje bitový a bytový přístup k I/O portům.

- **Bitový** – Využívá dva registry PINx a PORTx. Pro zápis se využívají instrukce SBI, CBI (např. SBI PORTA, 0) pro čtení pak instrukce SBIS, SBIC (např. SBIS PINA, 2).
- **Bytový** – Pro bytový (bajtový) přístup se využívají instrukce IN, OUT (např. IN R16, PINA).

10.2. Alternativní funkce portů

Porty (piny) kromě funkcí uvedených výše mají i další využití, které souvisí s perifériemi daného MCU. Přehled základních funkcí a jejich stručný popis obsahuje tabulka 5.

Zkratka	Popis alternativní funkce
RESET	Signál reset se využívá při sériovém nebo paralelním programování MCU
XTAL1/XTAL2	Takto pojmenované piny slouží k připojení externího zdroje časování MCU
USCK/SCL/XCK	Univerzální sériové hodinové rozhraní/Sériové hodiny pro USI/USART hodiny
DO/DI	Univerzální sériové rozhraní pro datový výstup / datový vstup
SDA/ICP	Sériové datové rozhraní / vstupní záchytný pin pro Č/Č
OC1B/OC1A/OC0A	Porovnávací výstupní bity – viz funkce čítače/časovače
AIN1/ AINO	Pozitivní / negativní vstup analogového komparátoru
PCINT0 – PCINT7	Piny pro přivedení externího přerušení
TxD/RxD/CKOUT	UART vysílač / přijímač / výstupní port pro vnitřní systémové hodiny

Tabulka 5 – Alternativní funkce MCU

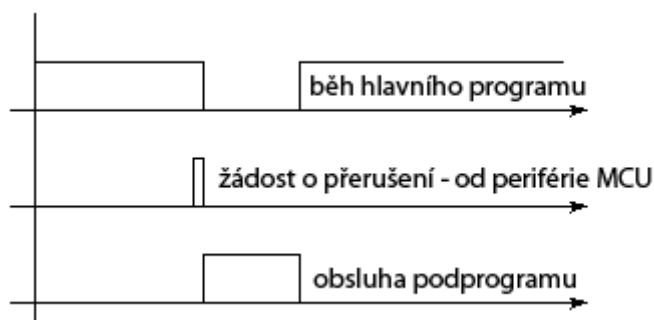
Popis a umístění registrů vstupně/výstupních portů jsou podrobně popsány v datasheetu AVR MCU.

11. Systém přerušení (Interrupt)

Přerušení je velice užitečná věc, bez které by se MCU (CPU) neobešly. Umožňuje kooperaci mezi běžícím hlavní programem a hardwarem MCU. Obecně lze princip přerušení MCU definovat takto:

Běží hlavní program MCU, v určitém okamžiku, přijde žádost o přerušení hlavního programu od zdroje přerušení, který chce být obsloužen (hardware periférie MCU). Pokud je přerušení povoleno, opustí se hlavní program (přestane se vykonávat) a začne se vykonávat obslužný podprogram dané periférie MCU. Po vykonání podprogramu, se procesor vrací zpět do hlavního programu na místo, kde ho opustil a pokračuje dál v jeho vykonávání.

Následující obrázek 25 graficky znázorňuje obecný princip přerušení.



Obrázek 30 – Systém přerušení (obecně)

MCU AVR mohou mít v základu až 16 zdrojů přerušení (zařízení, od kterých můžeme očekávat žádost o přerušení). Každý zdroj má nadefinovanou hardwarovou adresu – na tuto adresu se skočí právě při opuštění hlavního programu (konkrétní příklady viz tabulka 6, tabulka 7).

Adresy jednotlivých zdrojů přerušení jsou pochopitelně umístěny v paměti programu, protože sami uvozují nějaký podprogram. Dále je důležité si uvědomit, že jednotlivé adresy jsou umístěny hned za sebou, a proto na ně lze uložit pouze jednu skokovou instrukci (RJMP „návěští“). Ta nás odkáže dál do paměti programu, kde máme na uložení samotného podprogramu více místa.

Vektor	Adresa	Zkratka	Popis
1	0x0000	RESET	Inicializace systému, reset watchdog, spotřebu, programing
2	0x0001	INT0	Vnější zdroj přerušení číslo 0
3	0x0002	PCINT0	Kontrola změny úrovně na pinu
4	0x0003	TIM0_OVF	Přetečení čítače / časovače 0
5	0x0004	EE_RDY	Paměť EEPROM je připravena
6	0x0005	ANA_COMP	Analogový komparátor
7	0x0006	TIM0_COMPA	Shoda čítače / časovače (TCNT0) s registrem OCR0A
8	0x0007	TIM0_COMPB	Shoda čítače / časovače (TCNT0) s registrem OCR0B
9	0x0008	WDT	Přetečení čítače obvodu Watchdog
10	0x0009	ADC	ADC (Analog Digital Converter) převod dokončen

Tabulka 6 – Zdroje přerušení ATtiny 13

Vektor	Adresa	Zkratka	Popis
1	0x0000	RESET	Inicializace systému, reset watchdog, spotřebu, programing
2	0x0001	INT0	Vnější zdroj přerušení číslo 0
3	0x0002	INT1	Vnější zdroj přerušení číslo 1
4	0x0003	TIM1_CAPT	Vnější událost - zachycení obsahu čítače 1
5	0x0004	TIM1_COMPA	Shoda čítače s registrem OCR1A
6	0x0005	TIM1_OVF	Přetečení čítače / časovače 1
7	0x0006	TIM0_OVF	Přetečení čítače / časovače 0
8	0x0007	USART0, RX	USART - příjem dokončen (Rx)
9	0x0008	USART0, UDRE	USART – prázdný registr dat
10	0x0009	USART0, TX	USART – vysílání dokončeno (Tx)
11	0x000A	ANA_COMP	Analogový komparátor
12	0x000B	PCINT	Kontrola změny úrovně na pinu
13	0x000C	TIM1_COMPB	Shoda čítače s registrem OCR1B
14	0x000D	TIM0_COMPA	Shoda čítače s registrem OCR0A
15	0x000E	TIM0_COMPB	Shoda čítače s registrem OCROB
16	0x000F	USI_START	USI – počáteční podmínky nastaveny
17	0x0010	USI_OVF	USI – přetečení čítače
18	0x0011	EE_READY	Paměť EEPROM je připravena
19	0x0012	WDT_OVF	Přetečení čítače obvodu Watchdog

Tabulka 7 – Zdroje přerušení ATtiny 2313

11.1. Povolení přerušení

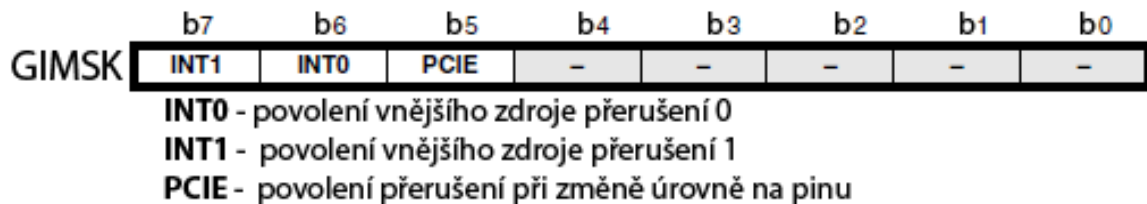
Pokud má být žádost o přerušení akceptována musí být nejdříve povolena v příslušném řídicím registru příslušné periférie (např. přerušení od čítače/časovače se nachází v řídicím registru TIMSK). Většina MCU využívá povolení přerušení na dvou úrovních.

- **Globální** – globální povolení přerušení je první ze dvou podmínek, aby byla žádost o přerušení od lokální periférie povolena. To znamená, že pokud je nastaven bit Global Interrupt Enable do stavu „log 1“, pak výše uvedení zdroje při žádosti o přerušení budou akceptovány (v závislosti na lokálním povolení přerušení). Tento bit lze s výhodou využít

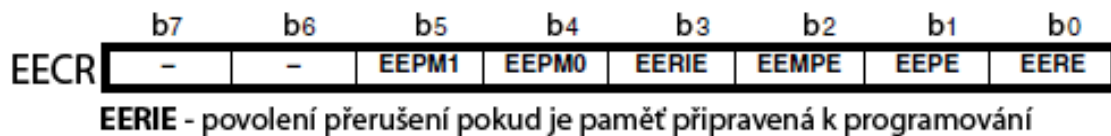
pro vypnutí všech přerušení MCU a nachází se v stavovém registru MCU (viz sbírka příkladů - Přílohy).

- **Lokální** – lokální povolení přerušení je druhou nutnou podmínkou k úplnému povolení přerušení. Týká se jednotlivých zdrojů přerušení, které jsou uvedené výše v tabulkách. Registry a bity jednotlivých zdrojů přerušení jsou popsány dále, kromě povolení všech přerušení od čítače/časovače, které se nacházejí v registru TIMSK (obrázek 23).

Pokud by jedna ze dvou podmínek nebyla splněna, nebude žádná žádost o přerušení mikroprocesorem akceptována.



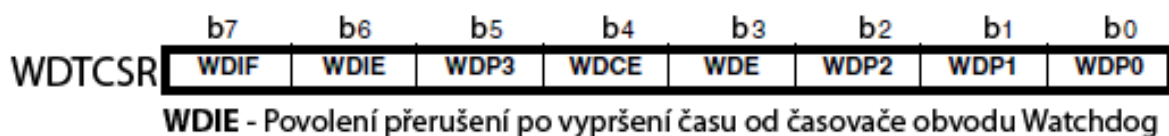
Obrázek 31 – (General Interrupt mask) Registr pro povolení externího přerušení



Obrázek 32 – (EEPROM Control register) Řídící registr paměti EEPROM



Obrázek 33 – (Analog Comparator Control Register) Řídící registr analogového komparátoru



Obrázek 34 – (Watchdog Timer Control Register) Řídící registr časovače obvodu watchdog

11.2. Priorita přerušení

Pokud by jednotlivé zdroje přerušení pracovali na stejné úrovni, nevěděl by MCU jak je rozsoudit v případě, že by dorazili dvě žádosti o přerušení od dvou zdrojů současně.

Priorita jednotlivých zdrojů přerušení je jednoznačně dána vektorem (pořadím) zdrojů přerušení v tabulkách 6 a 7. Nejvyšší prioritu má RESET přerušení a nejnižší poslední údaj v tabulce (poslední adresa zdrojů přerušení). Priority bohužel není možno měnit, ačkoliv by se nám to někdy hodilo.

12. Převod analogového signálu

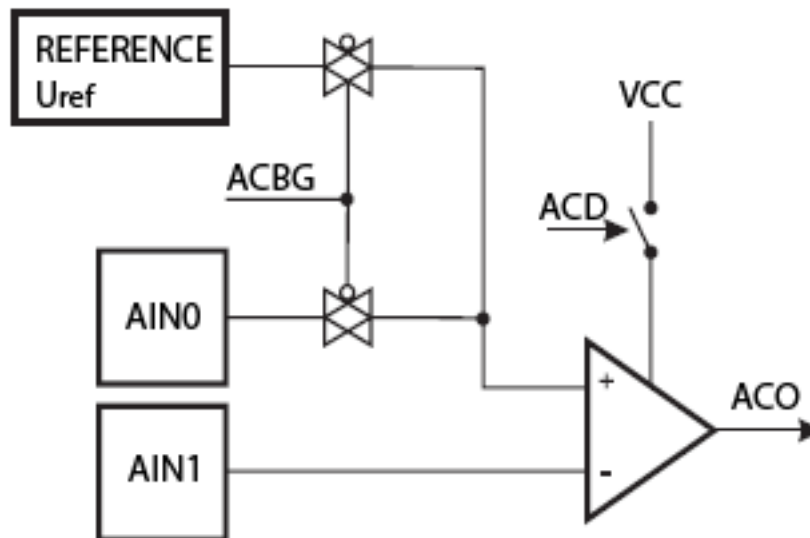
12.1. Analogový komparátor

Pokud chceme zpracovávat pomocí MCU analogový signál, je zapotřebí nejdříve ho převést do digitální formy. K tomuto účelu A/D převodníky. Pro jednodušší převod do digitálního tvaru lze využít levnější variantu a to analogový komparátor. Cena MCU tak dosáhne výrazného poklesu.

Vstupy analogového komparátoru jsou označeny AIN0, AIN1 (Analog Input). Úrovně napětí přivedených na vstupy lze mezi sebou porovnávat, nebo lze využít vnitřní referenci (U_{ref}) a porovnat s ní napětí na vstupu AIN1 (na vstup AIN0 je přivedeno U_{ref}). Velikost referenčního napětí je uvedena v základních vlastnostech MCU (viz kapitola 5.1).

Výstup analogového komparátoru ACO (Analog Comparator Output) obsahuje výsledek porovnání vstupů. Pokud je $AIN0 > AIN1$ pak bude $ACO = "1"$

Analogový komparátor umí vyvolat přerušení a to v případě překlopení ACO z "1" -> "0" nebo "0" <- "1" a lze ho od MCU odpojit v rámci snížení spotřeby. Jednoduché blokové schéma analogového komparátoru je zobrazeno na obrázku 31.



Obrázek 35 – Analogový komparátor (zjednodušené schéma)

	b7	b6	b5	b4	b3	b2	b1	b0
ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0

ACD - "log1" způsobí odpojení komparátoru od napájení (snížení spotřeby)
ACBG - "log1" určí na místo vstupu AIN0 napětovou referenci Uref
ACO - výstup komparátoru a zároveň příznak přerušení
ACI - pokud spustí výstup přerušení způsobené vlivem ACI0 a ACI1 je nastaven tento bit na "log 1"
ACIE - povolení přerušení pro analogový komparátor
ACISx - bity určují, kdy dojde k přerušení (při překlápění výstupu, setupná, vzestupná hran výstupu)

Obrázek 36 – Řídící registr analogového komparátoru

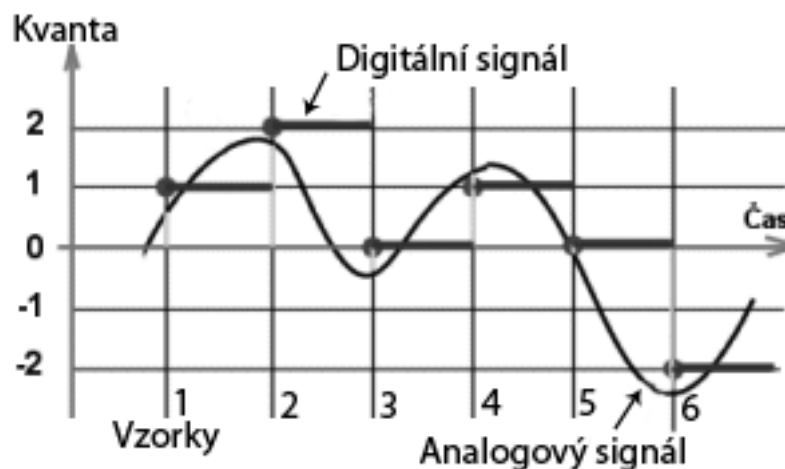
12.2. Analogově digitální (A/D) převodník

Analogově digitální (A/D, DAC) převodník, jak už je slyšet z jeho názvu, převádí analogový signál na digitální, ale s mnohem větší přesností než analogový komparátor. Jeho protějškem je digitálně analogový (D/A, ADC) převodník. Převodníky signálů mají v dnešní době velké využití v elektronice (MP3 přehrávače, fotoaparáty, rádia), v počítačích (zvukové karty) v telekomunikacích (GSM, ADSL, Wifi) a v dalších odvětvích. Abychom dobře pochopili princip převodu signálů, je důležité určit rozdíl mezi analogovým a digitálním signálem.

- **Analogový signál** – (spojitý signál) je dán s časem spojitou funkcí. Spojitý signál může být například sinusový signál, nebo audio signál budící reproduktory sluchátek.
- **Digitální signál** – (diskrétní signál) vyznačuje se určitým druhem nespojitosti (v hodnotě nebo v čase). Digitální signál může být binární signál, různé skokové signály čidel.

12.2.1. Převod analogového signálu na digitální

V této kapitole si stručně si představíme převod analogového signálu na digitální, s kterým umí MCU dále pracovat. Převod se skládá ze dvou fází. Nejdříve se provede vzorkování signálu a potom následuje kvantování.



Obrázek 37 – princip převodu analogového signálu na digitální

- **Vzorkování** – Vzorkování se provede tím způsobem, že rozdělíme vodorovnou osu signálu (v našem příkladu je na této ose čas) na rovnoměrné úseky a z každého úseku odebereme jeden vzorek (na obrázek 32 je to 6 vzorků). Je přitom zřejmé, že tak z původního signálu (f_{ANALOG_SIG}) ztratíme mnoho detailů, protože namísto spojitě čáry, kterou lze donekonečna zvětšovat dostáváme pouze množinu diskrétních bodů s

intervalem odpovídajícím použité vzorkovací frekvenci (f_{VZORK}). Aby bylo možné signál plnohodnotně obnovit, musí při vzorkování platit vztah: $2 \cdot f_{ANALOG_SIG} < f_{VZORK}$.

- **Kvantování** – po vzorkování je potřeba vzorkované hodnoty upravit i na svislé ose. Protože se hodnota vzorku dá vyjádřit pouze po určitých kvantech (hodnotách), nazýváme tuto fázi A/D převodu kvantování. Na obrázku 32 je patrné, že v našem případě jednotlivá kvanta nabývají celých čísel. Dále je patrné, že hodnoty kvant neodpovídají přesně průsečíku vzorku s analogovým signálem. Aby bylo možné určit, které hodnoty má po kvantování nabývat určitý vzorek, je třeba rozdělit prostor kolem jednotlivých hodnot (kvant) na toleranční pásy. Kterémukoliv vzorku, který padne do daného tolerančního pásu, je při kvantování přiřazena daná hodnota. Díky tomu se kvantované hodnoty ve většině případů liší od skutečných navzorkovaných hodnot. Této odchylce se říká kvantizační chyba.

12.2.2. Přesnost a kvalita převodu signálu

Protože se digitální signál zpravidla zpracovává na zařízeních pracujících ve dvojkové číselné soustavě, bývají počty kvantizačních úrovní A/D převodníků zpravidla rovny **N**-té mocnině čísla **2**, přičemž nakvantovaný signál pak lze vyjádřit v **N** bitech. Z krátké teorie je jasné, že větší počet bitů převodníku bude mít kladný vliv na kvalitu a věrohodnost převodu signálu.

12.2.3. ADC ATtinny 13

Přesto že MCU ATtinny 13 je jeden z nejmenších a nejjednodušších procesorů AVR, obsahuje 10bitový analogově digitální převodník. Přesnost převodníku je 2 LSB (LSB – poslední nejméně významný bit) a nelinearita $\pm 0,5\text{LSB}$ (nepřesnost vyjádření úrovně daného vzorku). Doba převodu se pohybuje mezi 13 a 260 μs . Rozsah amplitudy vstupního napětí je 0 až V_{cc} (napájecí napětí MCU), nebo 0 až U_{REF} (1,1V - hodnota vnitřního referenčního napětí).

A/D převodník pracuje ve dvou módech – v módu s jedním převodem a v módu volně běžícím. V prvním módu je každý převod řízen uživatelem. V druhém případě A/D převodník pravidelně vzorkuje vstupní signál a obnovuje navzorkovaná data v ADC (datový registr).

10 bitový výsledkem, který je uložený v datovém registr ADC, je rozdělen do dvou částí ADCH (High), ADCL(Low). Jsou dvě možnosti jak do registru ADCL a ADCH 10ti bitová data zapsat. Obě možnosti jsou dobře patrné z obrázku 33. Bit ADLAR, který definuje rozložení je umístěn v registru ADMUX. Při čtení dat z datových registrů je nejdříve nutné přečíst spodní ADCL registr a pak horní ADCH. Díky tomu nedojde ke ztrátě dat, protože při čtení ADCL je přístup k celému ADCx blokován. Po přečtení ADCH je opět možno do celého registr ADCx zapisovat nová data.

A/C převodník je vybaven vlastní předděličkou systémového času. Dělicí poměr se nastavuje v řídicím registr A/D převodníku pomocí bitů ADPSx v hodnotách druhých mocnin.

	b15	b14	b13	b12	b11	b10	b9	b8
ADLAR = 0								
ADCH	-	-	-	-	-	-	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
	b7	b6	b5	b4	b3	b2	b1	b0
ADLAR = 1								
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	-	-	-	-	-	-
	b7	b6	b5	b4	b3	b2	b1	b0

Obrázek 38 – datové registry ADCL a ADCH 10ti bitového A/D převodníku

	b7	b6	b5	b4	b3	b2	b1	b0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN - zapnutí ("1") / vypnutí ("0") A/D převodníku

ADSC - zapnutí převodu signálu

ADATE - automatické spuštění převodu s náběžnou hranou spoušť. signálu

ADIF - příznak přerušení od A/D převodníku

ADIE - povolení přerušení od A/D převodníku

ADPSx - nastavení dělicího poměru systémového času pro A/D převodník

Obrázek 39 – ADCSRA řídicí registr A/D převodníku

Analogové vstupy jsou v našem případě u ATtiny13 označeny ADC0 až ADC3. Tyto vstupy jsou přivedeny na multiplex. V závislosti na bitech MUX0 a MUX1, které jsou umístěny na začátku registru ADMUX, jsou vybírány vstupy, které se připojí k A/D převodníku.

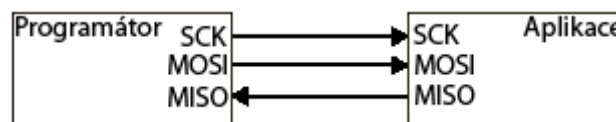
13. Komunikační rozhraní MCU

13.1. Sériové periferní rozhraní - SPI (Seriál Peripheral Interface)

Toto sériové rozhraní slouží pro programování MCU a dalších integrovaných obvodů přímo v aplikaci. Není tedy nutné MCU z aplikace vyndávat z patice a programovat ho v univerzálních programátorech (In system programming).

Programování se provádí přes speciální piny označené RESET, MISO, MOSI, SCK, VCC, GND a využívá se komunikace MASTER/SLAVE. V našem případě je MASTER programátor a SLAVE je MCU v aplikaci. Průběh komunikace vypadá následovně:

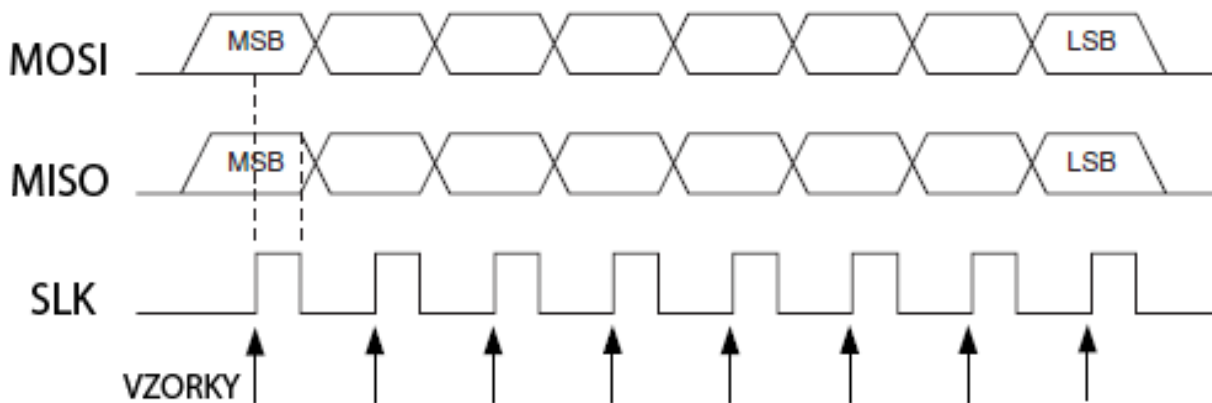
- Nejdříve se začne generovat hodinový signál na SCK a v té chvíli vyšlou obě zařízení svoje data, přičemž MOSI je vždy Master výstup, Slave vstup a MISO naopak.
- Jakmile jsou data vyslána může komunikace dále pokračovat, Master dále dodává hodinový signál, hodnota SS se nemění
- Ukončení komunikace zajistí Master tím, že přestane vysílat hodinový signál.
- Délka vyslaných dat je buď 8bit (Byte) a nebo 16bit (Word).



Obrázek 40 – SPI komunikace (In system programming)

Časový průběh komunikace zobrazuje obrázek 36 – jedná se o synchronní přenos dat se synchronizační složkou (SLK). Pokud zapisujeme sériová data do MCU, pak jsou data

synchronizována s náběžnou hranou hodinového signálu SLK. Pokud data z MCU čteme, jsou synchronizována na sestupnou hranu hodinového signálu.



Obrázek 41 – časový diagram komunikace SPI

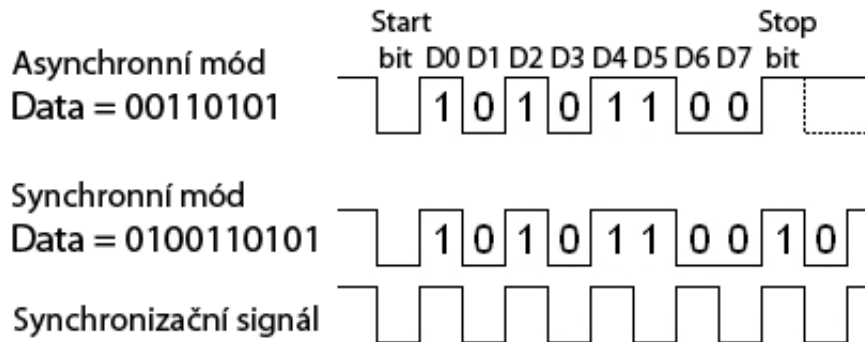
13.2. USART (Univerzální synchronní asynchronní vysílač přijímač)

Jedná se o sériové komunikační rozhraní, které dokáže obousměrně komunikovat s CP nebo jinými MCU synchronně nebo asynchronně s vysokou přenosovou rychlostí (Baud/s).

Rozhraní USART využívá dva datové vodiče Rx (přijímač), Tx (vysílač). S použitím obvodu MAX232 lze převést dvouvodičové rozhraní UART na sériové rozhraní RS232 (Cannon 9).

- **Synchronní komunikace** – halfduplexní přenos využívá k přenosu synchronizační složku. Tato složka je stále vysílána po jednom vodiči a udržuje tak přijímač a vysílač časově sladěný. Data se vysílají po druhém vodiči halfduplexně (obrázek 37).
- **Asynchronní komunikace** – synchronizační složka není při tomto přenosu přítomná. Aby přijímací strana dat mohla přesně definovat časové intervaly, kdy má vyhodnotit data. To znamená je potřeba přesně vyhodnotit časový interval, odpovídající přenosu jednoho bitu. K tomuto účelu je každá zpráva (byte) opatřena START bitem a STOP bitem (mohou být i dva). START bit – určuje začátek zprávy a zajišťuje zasynchronizování zprávy. STOP bit - určuje konec zprávy a umožní předzpracovat přijatou zprávu (obrázek 37).
- **Přenosová rychlost [Baud/s]** – přenos dat v Baudech udává počet změn signálu za sekundu, nikoli počet bitů za sekundu. Do jedné signálové změny lze zakódovat i více než

jeden bit. Pokud jedna signálová změna zahrnuje pouze jeden bit, pak se přenosová rychlost v bit/s rovna v Baud/s.



Obrázek 42 – Průběhy synchronního a asynchronního módu

13.3. Registry rozhraní USART

Pro vysílání a příjem dat se využívá registr UDR. Tento registr je fyzicky zastoupen dvěma registry pro příjem (UDR read - RXD) a pro vysílání (UDR write - TXD), ale oba mají stejnou adresu.

- Je-li jednotka USART korektně nastavena, provede se odeslání byte pouhým zapsáním do registru UDR a odtud je dále přeposlán na TX pin, přičemž je jim přiřazen na začátek start a na konec stop bit. Po úspěšném odvysílání je nastaven příznak TXC v registru UCSRA.
- Je-li jednotka USART korektně nastavena, pak po detekci start bitu na pinu RX je přijat byte do registru UDR. Po příjmu znaku je nastaven příznak RXC v registru UCSRA. Poté je možno přečíst přijatý znak opět z registru UDR. Důležité je, že znak může být čten pouze jednou.

	b7	b6	b5	b4	b3	b2	b1	b0
UCSRA	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM

RXC - příznak po úspěšném přijetí dat

TXC - příznak po úspěšném odeslání dat

UDRE - příznak vyprázdněného bufferu (datového registru USART) - indikuje, že je možné přimout nová data.

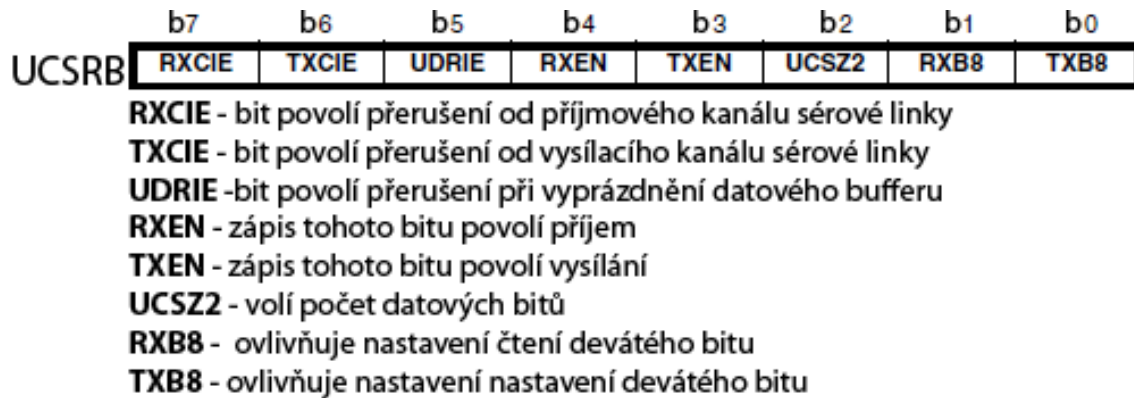
FE - indikuje chybu rámce

DOR - indikuje přeplnění bufferu

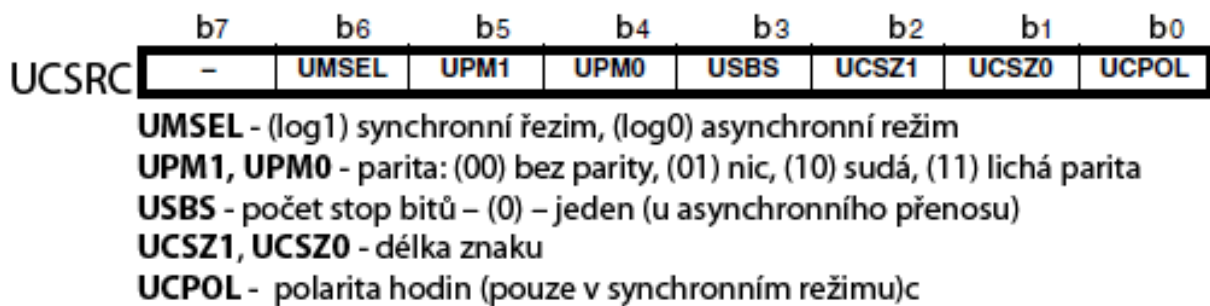
UPE - chyba parity

U2X - zdvojnásobení přenosové rychlosti

Obrázek 43 – UCSRA řídicí registr a registr příznaků obvodu USART



Obrázek 44 - UCSRB řídicí registr a registr příznaků obvodu USART



Obrázek 45 - UCSRC řídicí registr a registr příznaků obvodu USART

Přenosovou rychlost v jednotkách Baud/s je možné nastavovat pomocí 16bitového registru UBRRR. Tento registr je tvořen dolním a horním 8bitovým registrem.

14. Řízení výkonu MCU (Power management)

Aplikace, které nevyužijí všechny periférie MCU, mohou pracovat s malým výkonem. Toho dosáhneme vypnutím nepotřebných modulů MCU, např. D/A převodníky, analogový komparátor, USI, čítač/časovač, watchdog, blok přerušení, atd. Snížení spotřeby se ve většině případů řídí pomocí řídicích registrů jednotlivých periférií. Odpojení některých

z periférií se také využívá z důvodů menšího rušení dalších bloků MCU. Podrobnější informace naleznete v datasheet konkrétního obvodu (MCU).

15. Použitá literatura

Seznam použité literatury, internetových adres a zajímavé odkazy na tematiku AVR.

- [1] Vladimír Váňa: „Mikrokontroléry ATMEL AVR popis procesorů a instrukční soubor“
BEN technická literatura, Praha 2003
- [2] <http://www.atel.cz>
- [3] <http://www.hw.cz>
- [4] <http://avr-forum.com>
- [5] <http://www.avr-feaks.net>
- [6] <http://www.wikipedia.com>
- [7] AVR Studio 4.15 – Help F1