Vstupy a výstupy jednočipového mikroprocesoru

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

První systémy

- **Výstupy – latch, registry** připojené ke společné sběrnici. Mapované do I/O nebo datového prostoru.
- **Vstupy – obvody s třístavovým výstupem (**budiče**,** latch, registry nebo multiplexery) připojené ke společné sběrnici. Mapované do I/O, datového prostoru případně i programového prostoru.

Nevýhoda – procesorový systém pro zařízení s jiným počtem vstupů a výstupů musel být obvodově upravován.

⇒ **Vznik programovatelných vstupně-výstupních obvodů** I8255 (I8080), PIO (Z80).

Koncepce převzata k realizaci I/O bran jednočipových procesorů pro maximální přizpůsobení k ovládání realizované aplikace. **Každý vývod je konfigurovatelný** do

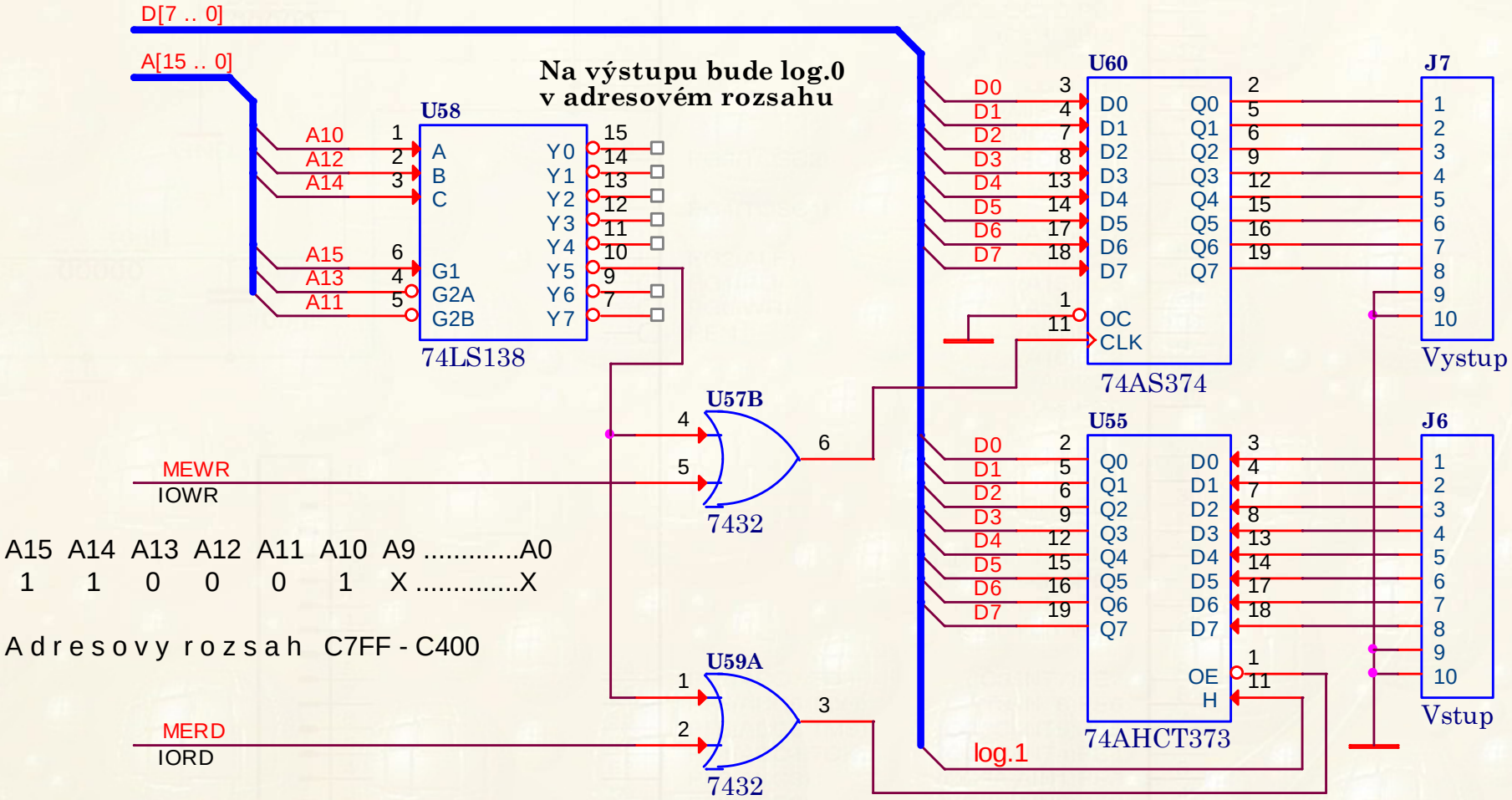
- Vstupního nebo výstupního režimu
- Stavu pro měření analogové veličiny (je-li vybaven A/D)
- U některých µP upnutí vývodu přes odpor k napájení nebo k zemi.
- Řízení doby náběžné a sestupné hrany (přechod 0→1 a obráceně).

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

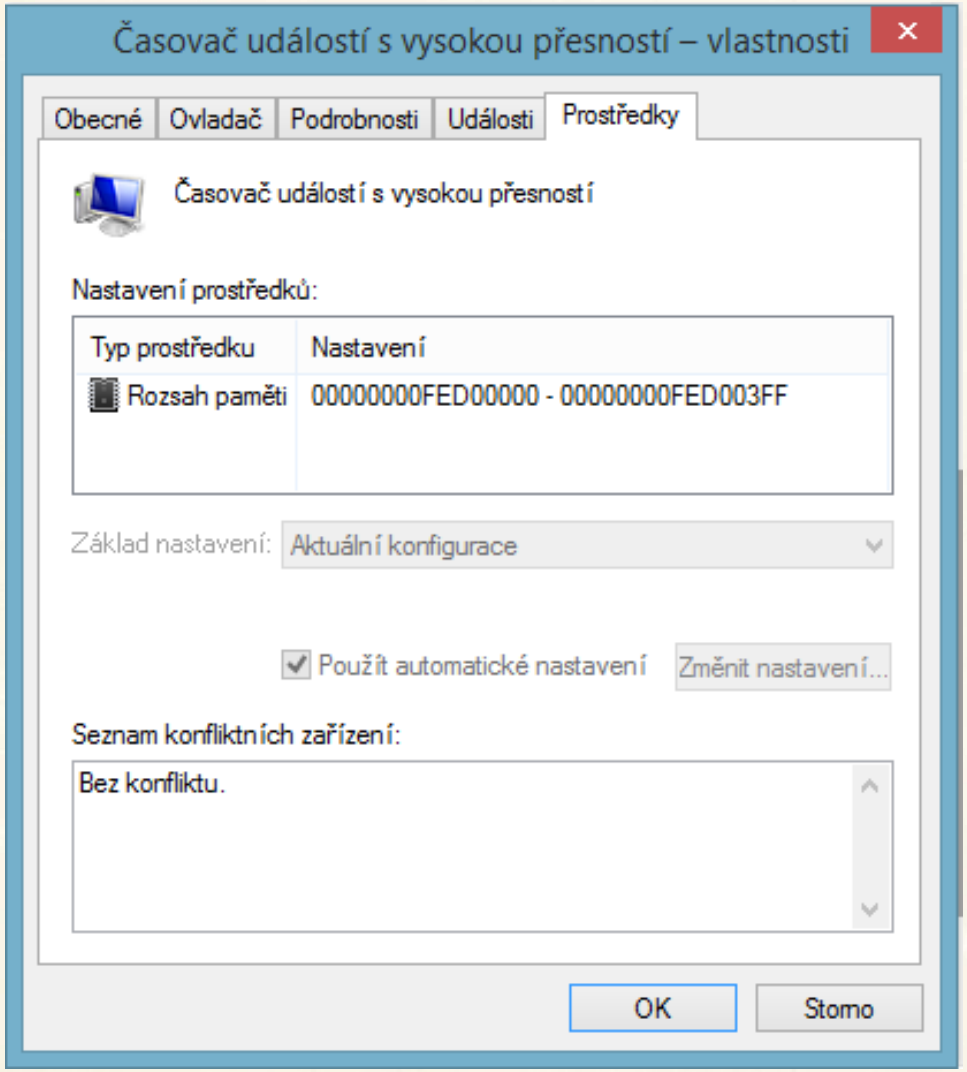
Připojení vstupu a výstupu do adresového prostoru C7FF÷C400.

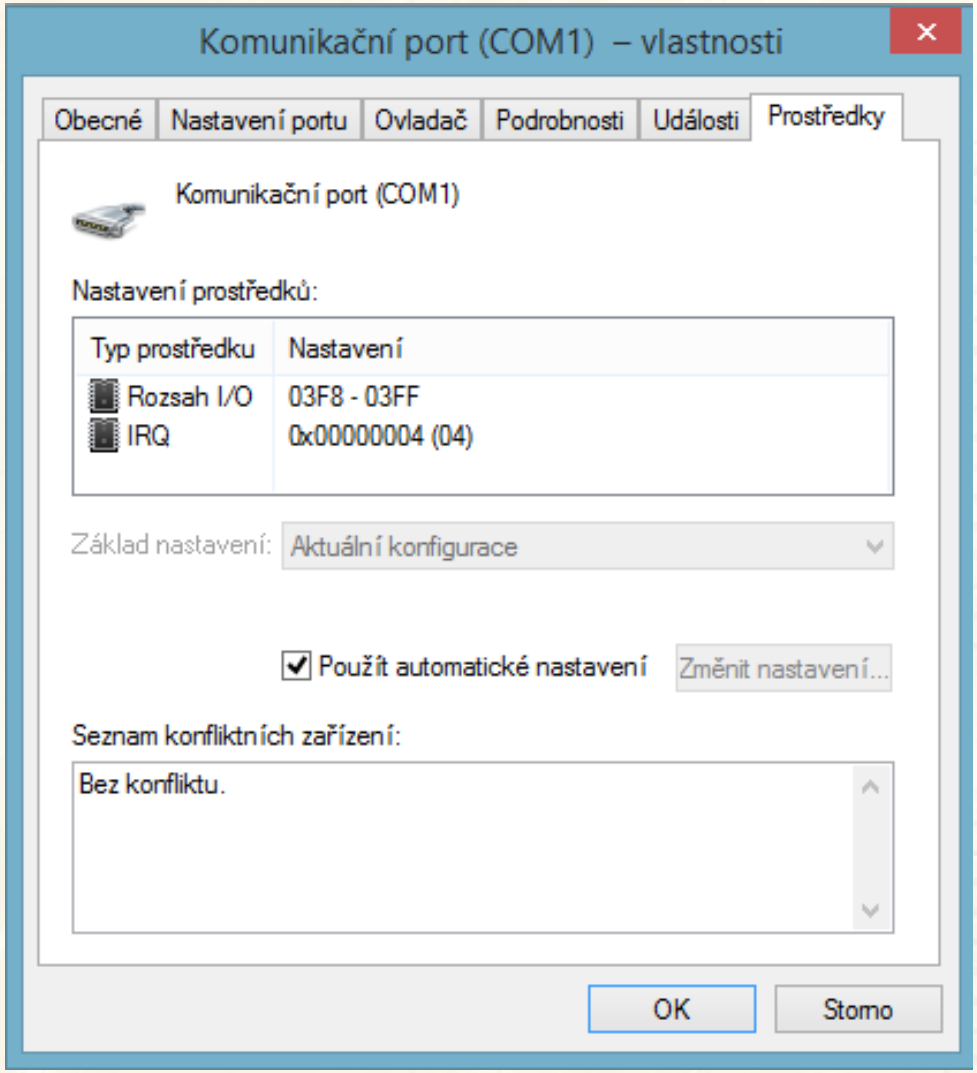
Pro signály MEWR a MERD – **paměťově mapovaná** periferie

Pro signály IOWR a IORD – periferie **ve vstupně-výstupním** prostoru



MAPOVÁNÍ PERIFERIÍ V POČÍTAČI PC

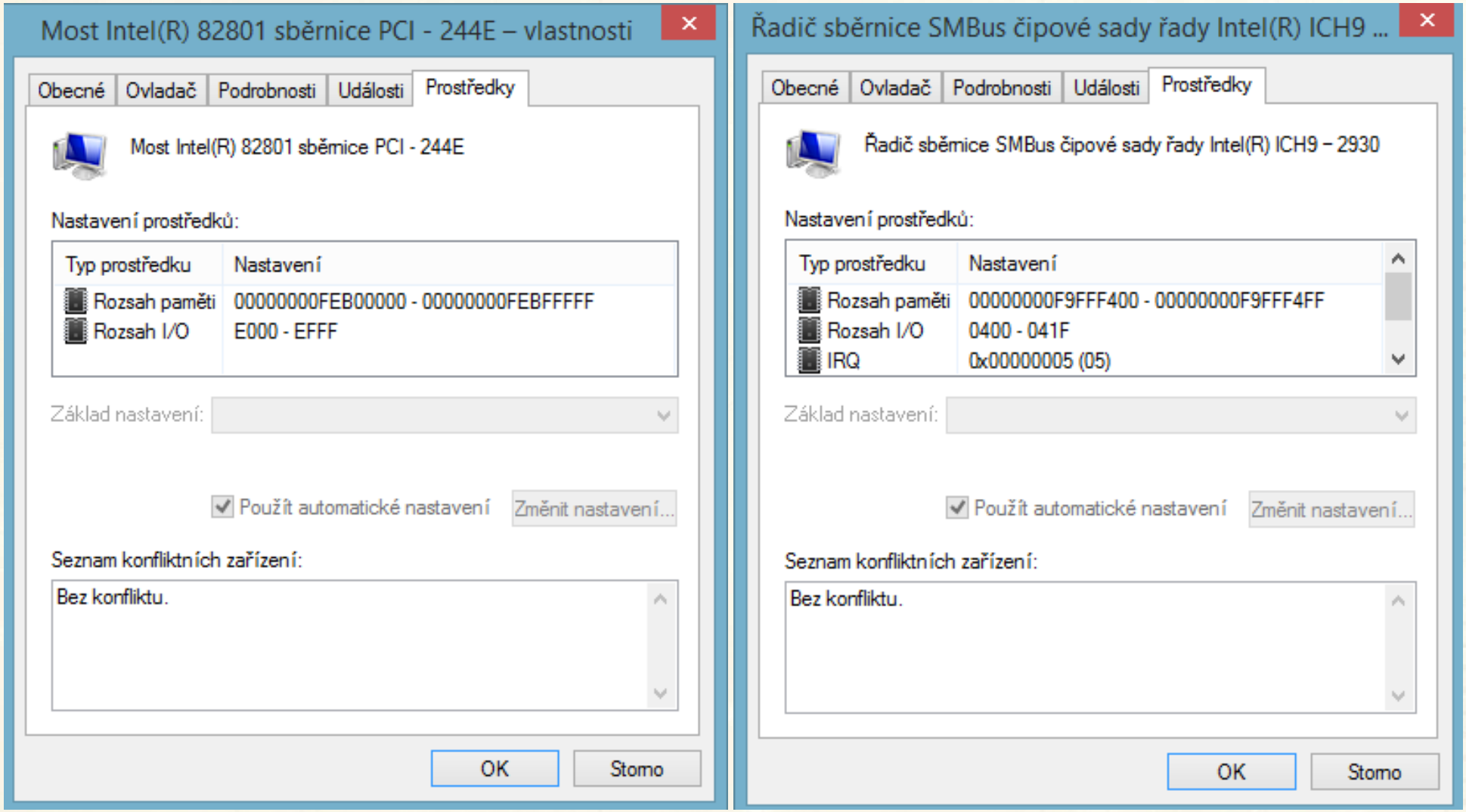




Paměťově mapovaná periferie

periferie **ve vstupně-výstupním** prostoru

MAPOVÁNÍ PERIFERIÍ V POČÍTAČI PC



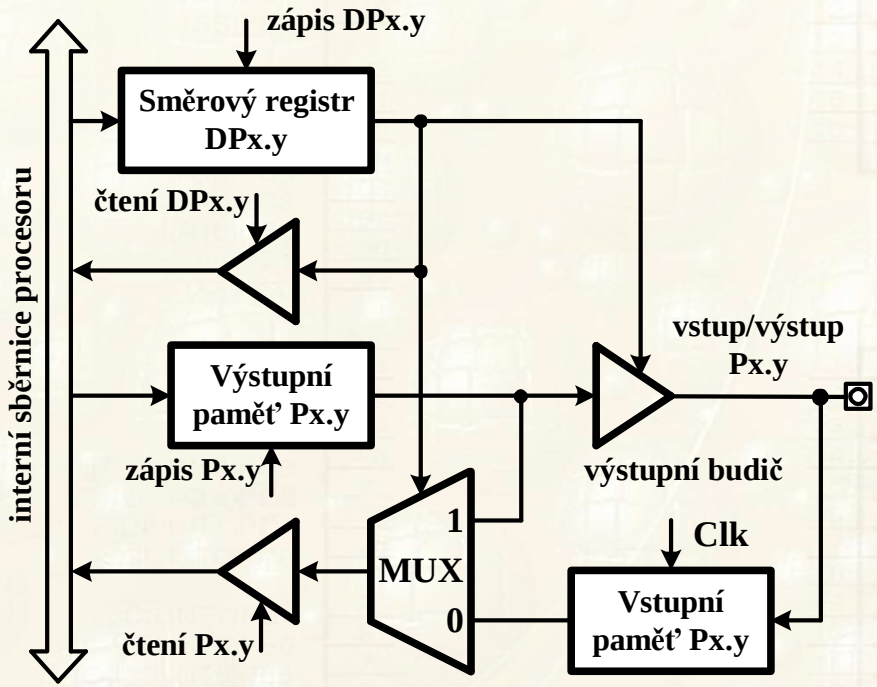
Paměťově mapované periferie pravděpodobně se stránkováním a pravá spojená s přerušovacím systémem.

VSTUPY A VÝSTUPY JEDNOČIPOVÝCH MIKROPROCESORŮ

Každý vývod **vstupně-výstupní brány** typu **Push-Pull** se skládá:

- **Paměťového členu** uchovávajícího výstupní hodnotu
- **Multiplexeru** pro zpětné čtení zapsané hodnoty do výstupu nebo hodnoty přivedené na vývod brány.
- **Směrového registru** určujícího zda vývod brány bude logickým vstupem nebo výstupem.
- **Třístavového výstupního budiče** a třístavových budičů připojených k datové sběrnici.
- **Synchronizaci** asynchronního vstupního signálu zajišťuje **dvojnásobný** nebo **vícenásobný** synchronizátor.

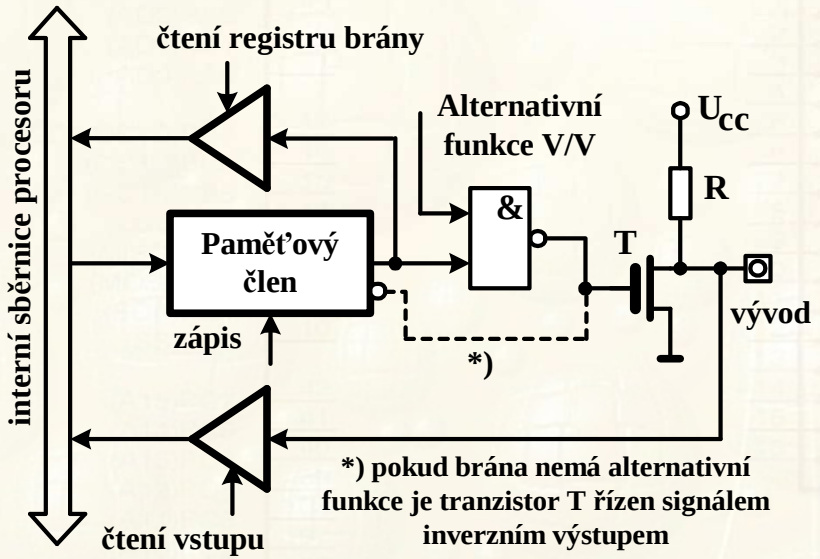
Při **nulování procesoru** – směrový registr je nastaven (po připojení napájení) do **vstupního stavu** nebo do **stavu vysoké impedance**.

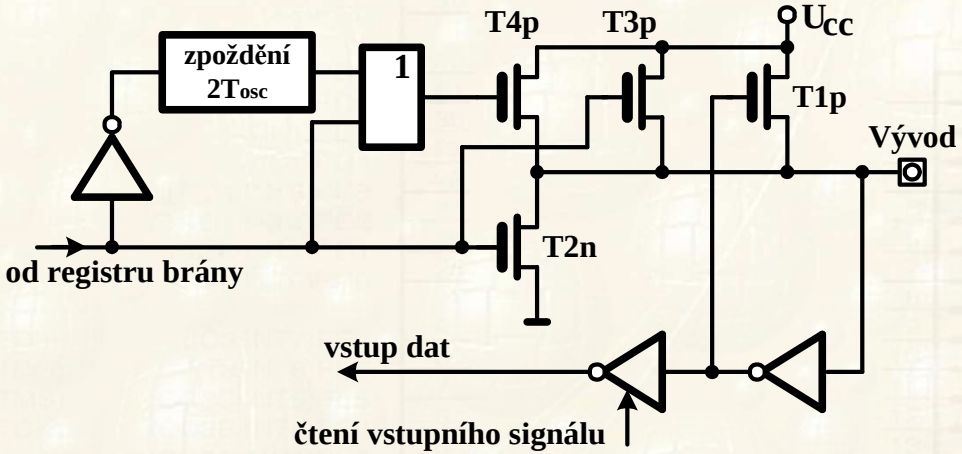


Pseudoobousměrná brána (obvodu s otevřeným kolektorem) se skládá z:

- **Paměťového členu**
- **Slučovače** výstupu s alternativním výstupem (periferií)
- **Výstupního tranzistoru** (obvykle s upínacím odporem)
- **Oddělovače** pro čtení vstupu
- **Oddělovače** pro čtení zapsané hodnoty v paměťovém členu.

Zpětné čtení stavu výstupu zavedeno pro připojení přechodu **báze-emitor** tranzistoru na výstupní vývod. Upínací odpor (T1p) má odpor **30÷ 50kΩ,** při přechodu 0→1 snížen na dobu dvou period hodinového signálu na hodnotu (T4p) **500 Ω**.





VARIANTY VSTUPŮ A VÝSTUPŮ JEDNOČIPOVÝCH MIKROPROCESORŮ

8051 - Výstup s otevřeným kolektorem (OK)
log.0 – pozor na připojení výkonného zdroje
log.1 – výstup je schopen poskytnout proud cca 140μA
Vstupní režim (výstup=log.1, u P0 chybí odpory**)**

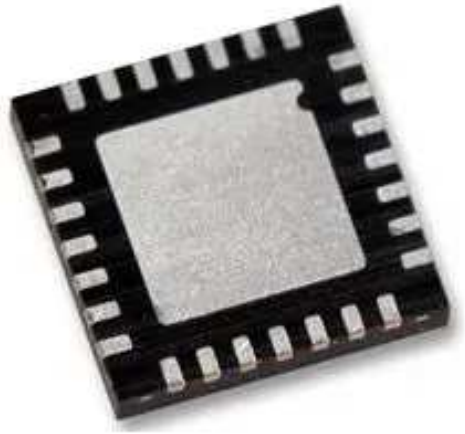
AVR - Výstup - Push-Pull (log.1 ve zkratu až 100mA)
Vstup - Plovoucí
- s upnutím přes odpor k napájení (Pull-UP)

ARM - Výstup - Push-Pull
- Otevřený kolektor
Vstup - Plovoucí nebo s upnutím
Analogový vstup multiplexoru k A/D převodníku
Alternativní funkce dle konfigurace registrů AFR
(viz. manuál daného procesoru)
U vývodů je volitelná rychlost přeběhu, upnutí přes odpor k napájení (Pull-UP) nebo k zemi (Pull-DOWN) a nastavení stavu po připojení napájení.

JEDNOČIPOVÝ MIKROPROCESOR S PAMĚTÍ EPROM, OTP A FLASH

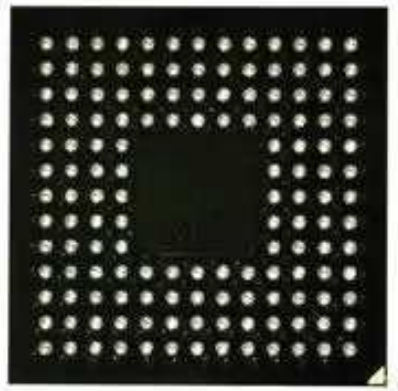
DIL

PLCC



UFQFN

TFBGA



TSSOP

TQFP

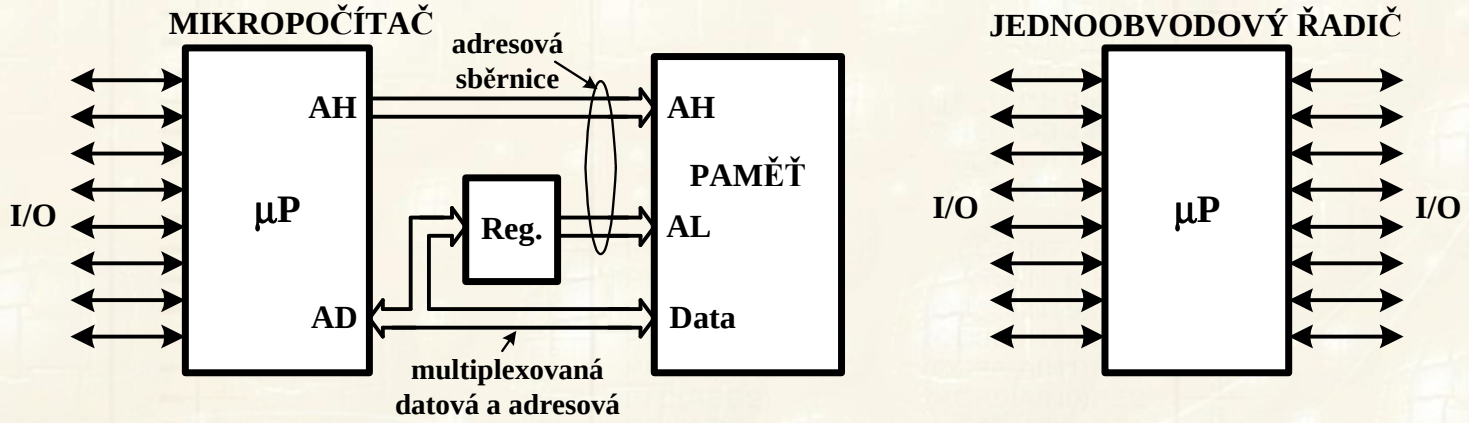
JEDNOOBVODOVÉ MIKROPROCESORY A ŘADIČE

Historicky se ustálilo rozdělení jednočipových procesorů na

- **Mikroprocesory -** V/V vývody je možné využívat k
  - **připojení periferií**
  - **vytvoření společné sběrnice pro připojení vnějších pamětí**, V/V bran, převodníků, displejů, atd.

Periferie připojené ke společné sběrnici jsou **mapovány v programovém, datovém nebo vstupně/výstupním prostoru** a ovládány **jedinou instrukcí.**

- **Mikrokontroléry (řadiče)** - u těchto obvodů na V/V vývodech
  - **nelze** vytvořit **společnou sběrnici** ovládanou **jedinou instrukcí**.
  - Lze vytvořit **sběrnici programově ovládanou,** (programem o několika instrukcích pro práci s V/V vývody mikrořadiče).



Přerušovací systém mikroprocesoru

PŘERUŠOVACÍ SYSTÉM PROCESORŮ - OBECNĚ

Přerušovací systémem (**Interrupt systém**) podstatně zjednodušuje a zefektivňuje styk s vnitřními i vnějšími periferiemi. Přerušení je takový **„zvonek“ .**

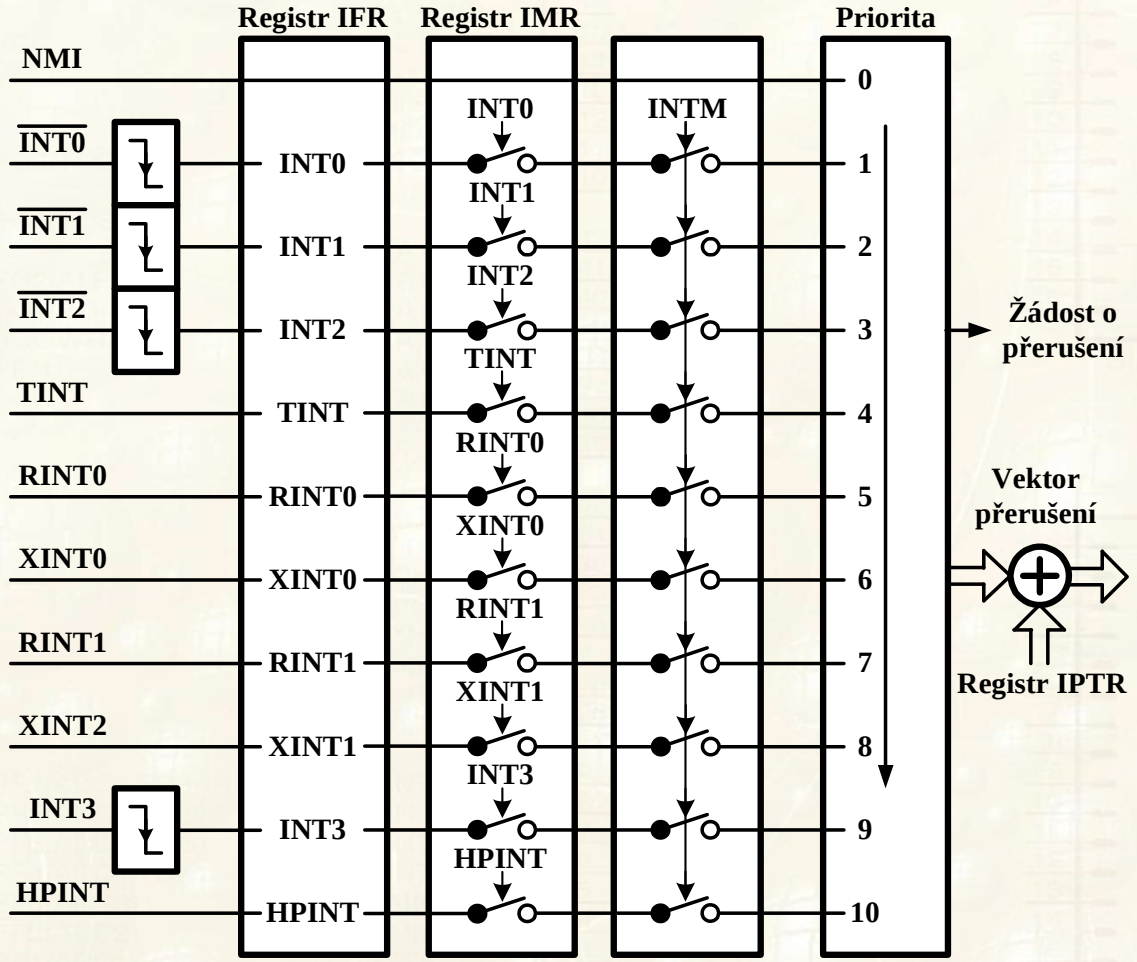
- Obsluha periferií bez přerušovacího systému
  - Snížení výpočetního času pro hlavní program
  - Složitější program s testováním žádostí
  - Obtížnější zajištění zpracování **v potřebných intervalech**
- Obsluha s využitím přerušovacího systému
  - Periferie požádá o přerušení
  - Procesor vyhodnotí žádost
  - Zkontroluje zda přerušení od dané periferie je povolené
  - Vyhodnotí zda není další žádost s vyšší prioritou
  - Přeruší ve vhodném okamžiku probíhající program (dokončí právě rozpracovanou instrukci nebo instrukce („pipeline“))
  - Přejde ke **zpracování obslužného programu**
- Žádost periferie o přerušení je vyvolána obvodově obvykle pomocí instrukce **CALL adresa_přerušení**.
- **Rozdíl mezi CALL podprogram a Zavolání přerušení?**

PŘECHOD DO PŘERUŠENÍ - OBECNĚ

- Uloží PC do zásobníkové paměti (adresu následující instrukce po poslední vykonané)
- Nastaví PC na adresu volaného přerušení (přepisem PC se ztrácí informace o místě, kde byl program přerušen).
- Je zablokována daná úroveň přerušovacího systému
- Procesor zahájí svoji činnost od nastavené adresy obslužného programu
- **Používané registry v obslužném programu je nutné uložit včetně stavu příznakového registru**
- Na konci přerušení musí být stav registrů a příznaků obnoven.
- Přerušení je ukončeno instrukcí RETI (povolení přerušení). Atypicky možnost návratu bez povolení přerušovacího systému.
- Jiné varianty uložení obsahu registrů
  - Existence dvou/více bank registrů (Z80, ADSP21xx, 8051, atd.)
  - Jednoúrovňové FIFO základních registrů (TMS320C5x).

PŘERUŠOVACÍ SYSTÉM PROCESORŮ - OBECNĚ

- Každá periferie má **bit** (**interrupt flag**) indikující žádost o přerušení a bit (**interrupt mask**) povolující nebo zakazující jeho přerušení.
- **Global mask** – povolující všechna povolená přerušení nebo zakazující všechna přerušení.
- Priority nastaveny výrobcem s ohledem na předpokládané použití.
- Vyhodnocení dvou žádostí ve stejnou dobu
- **Nested interrupt**
- **Nemaskované přerušení (NMI)**
- Změna priorit
- Přeadresování vektorů přerušení



PROBLÉMY VĚTŠÍHO POČTU PŘERUŠENÍ A VOLBA PRIORITY

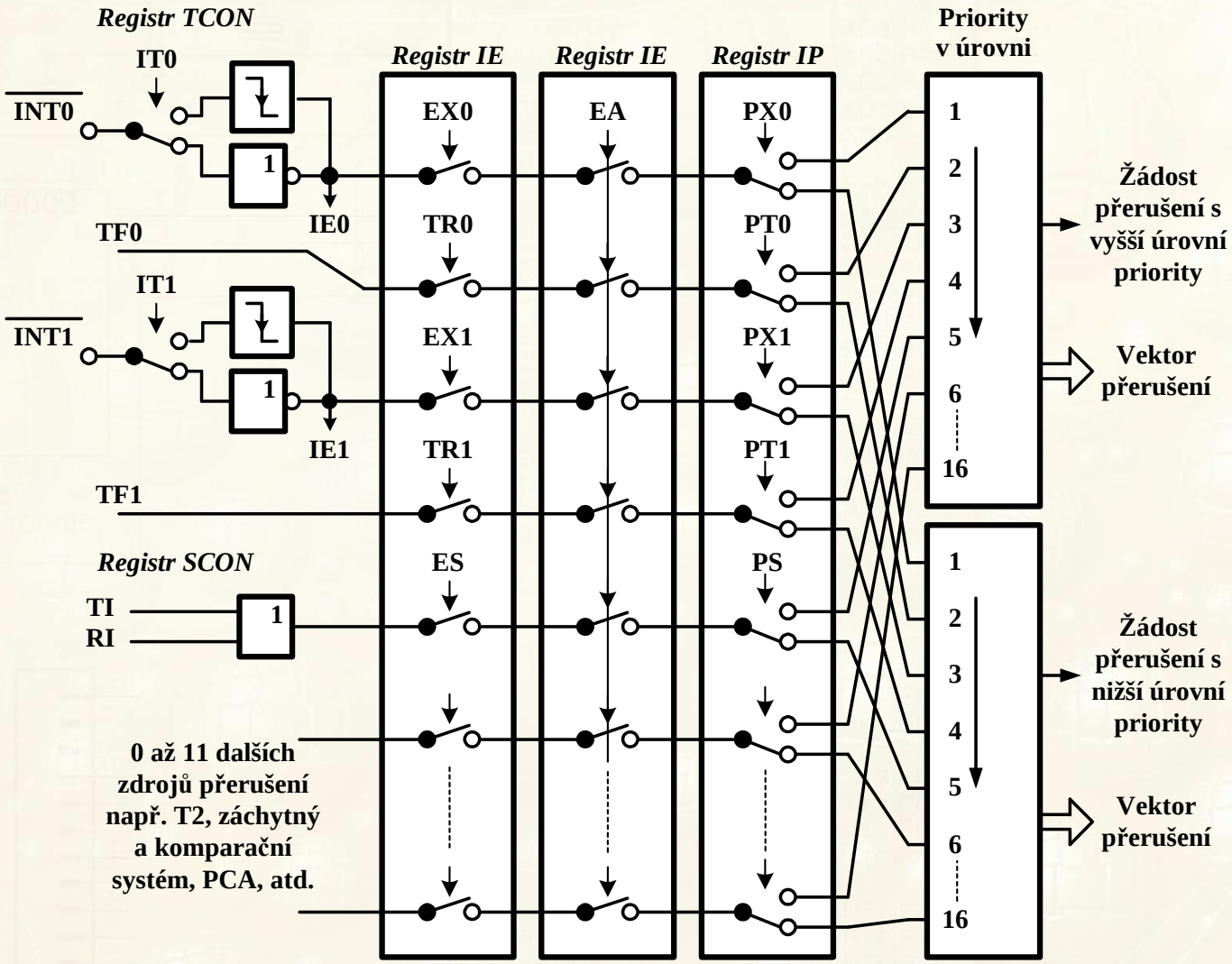
Je-li v aplikaci více zdrojů přerušení, mohou nastat problémy s dobou přístupu do zpracování jednotlivých obsluh. Volba procesoru pak ovlivňuje:

- ❑ Počet úrovní přerušovacího sytému.
- ❑ Možnost změny priority daného přerušení.
- ❑ Možnost přerušení v přerušení

➢ **Jednoúrovňový systém s pevně danými prioritami** zdrojů

- o Každé přerušení má svoji adresu a po vstupu do obsluhy přerušení **maže procesor jeho příznak**. Změna priorit jednotlivých přerušení je **obtížná** (AVR).
- o Více přerušení má jednu adresu – **příznak musí být mazán programově**. Programová změna priorit jednoadresových přerušení – je **možná** (TMS320C15, některé 8051, ARM)
- o **Nested interrupt** – Způsob jak vyhovět časově kritickému přerušení při zpracovávání jiného déle trvajícího přerušení. Podpořeno **více úrovněmi** nebo definovaným postupem.

PŘERUŠOVACÍ SYSTÉM S VOLITELNOU ÚROVNÍ PRIORITY



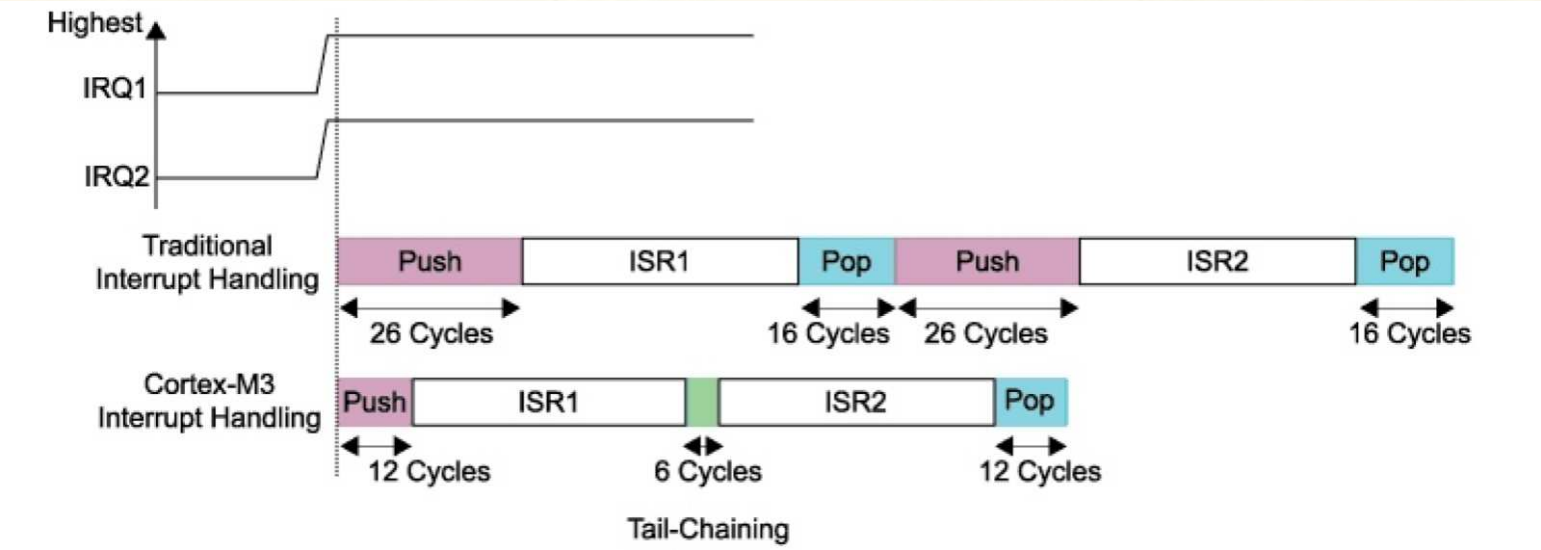
PŘERUŠOVACÍ SYSTÉM S VOLITELNOU ÚROVNÍ PRIORITY

- **Více úrovňový systém s pevně danými prioritami**
  - Přerušení s jakoukoliv prioritou lze přesunout do vyšší úrovně priority, z které přeruší i přerušení s vyšší prioritou umístěné v nižší úrovni priorit – tzv. **přerušení v přerušení.**

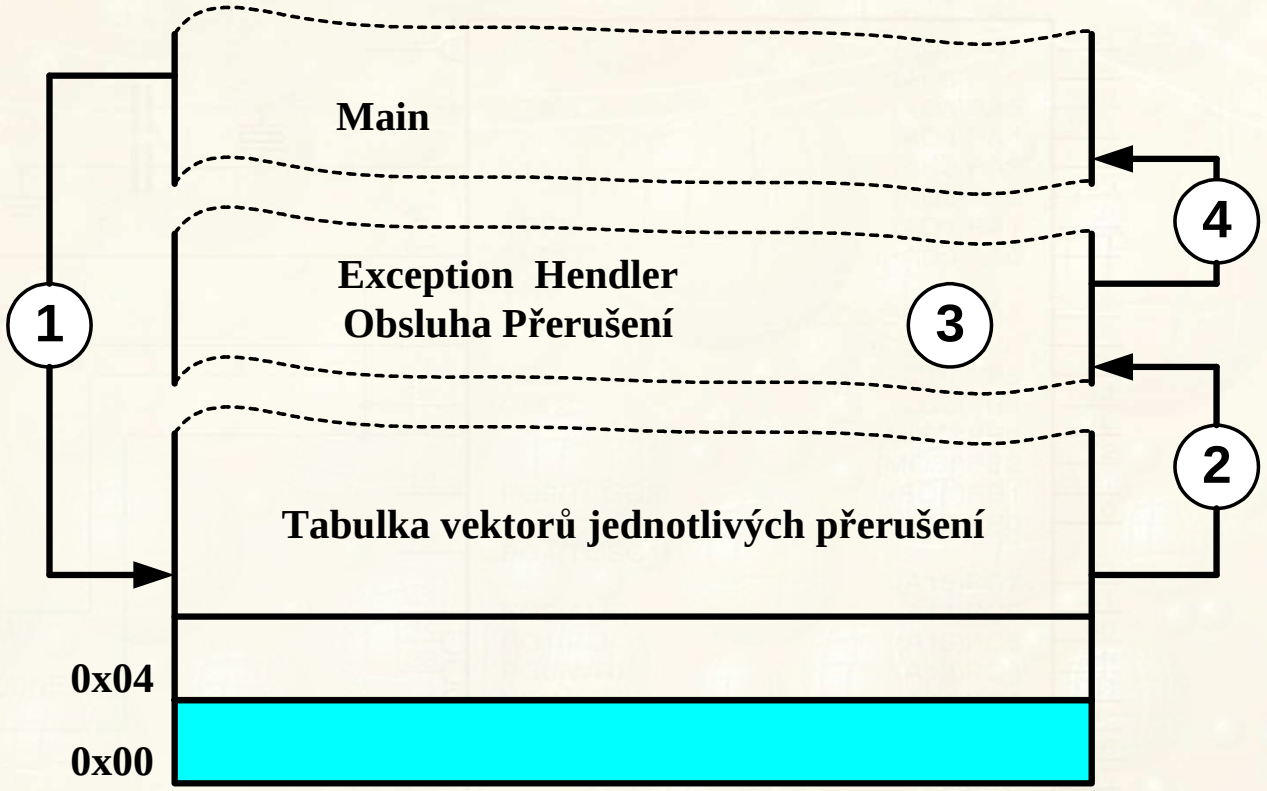
- **Žádosti o přerušení** se testují v definovaném okamžiku strojového cyklu procesoru a posléze se vyhodnocují.
- **Vyhodnocení** – je-li přerušení povolené, není další žádost přerušení s vyšší prioritou a je povolený přerušovací systém.
- **Vyvolání přerušení** – instrukcí LCALL adresa přerušení nebo přečtením adresy. Dojde k uložení návratové adresy, přenesení adresy přerušení do PC, zakázání přerušovacího systému, případně uložení některých registrů).
- **Zpracováním přerušení** nesmí být ovlivněny hodnoty **registrů** a **příznaků** před přechodem do přerušení.

ODEZVA PŘERUŠOVACÍHO SYSTÉMU PROCESORŮ

- **Odezva** přerušovacího systému na periodické požadavky **fluktuuje** – počet strojových cyklů instrukcí, asynchronnost hodinového kmitočtu procesoru a žádostí o přerušení.
- Vzorkování A/D a D/A převodníků opřené o přerušovací systém ⇒ vede k **fázové nestabilitě** a zmenšení poměru S/N nebo SFDR.
- Zkrácení odezvy na obsluhu přerušení ⇒
  - ✓ Umístění zásobníku (datová paměť, interní paměť, registr)
  - ✓ Uložení registrů (banky registrů, jednoúrovňové FIFO)
  - ✓ Segmentace instrukcí



CHOVÁNÍ PŘI PŘIJETÍ PŘERUŠENÍ ARM



1. Přijetí žádosti o přerušení, dokončení aktuální instrukce, přístup do tabulky vektorů přerušení
2. Uložení adresy přerušení do PC
3. Spuštění obsluhy přerušení v Handler mode
4. Ukončení obsluhy a návrat do hlavního programu

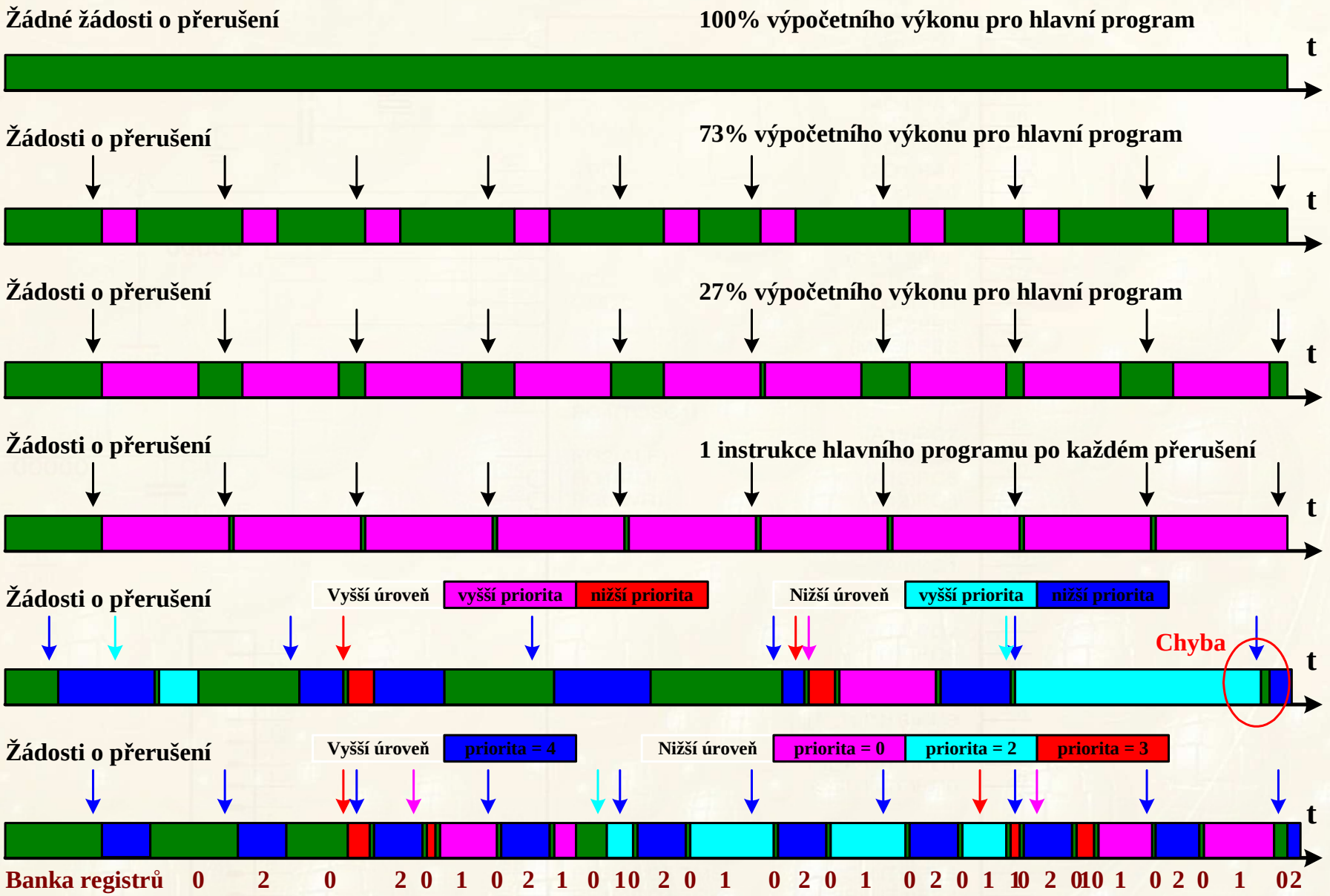
ADRESY PŘERUŠOVACÍHO SYSTÉMU ARM

- Při přerušení (výjimce) dojde k:
  - Uložení osmi registrů v následujícím pořadí PSR (program status), PC, LR, R12, R3, R2, R1, R0 do zásobníku.
  - Načtení adresy přerušení z tabulky vektorů přerušení
  - Aktualizuje se ukazatel zásobníku, link registr (LR) a čítač instrukcí (PC)
- Uložení registrů se realizuje do zásobníku (PSP/MSP), podle aktuálně používaného. SP je aktualizován na (MSP) používaný v přerušení i pro další vnořená přerušení (nested interrupts).
- V registru PSR (program status) bude aktualizována hodnota IPSR na číslo přerušení.
- Registr LR je aktualizován na speciální hodnotu nazvanou EXC_RETURN (0xFFFFFFFx), která slouží jako informace k návratu z přerušení.

ADRESY PŘERUŠOVACÍHO SYSTÉMU ARM

- Oproti řadě procesorů, kde obslužný podprogram musí být zakončen instrukcí RETI (RETurn from Interrupt), je u procesorů ARM přerušovací rutina ukončena skokem na speciální adresu EXC_RETURN (0xFFFFFFFx) uloženou do registru LR na začátku přerušovací rutiny.
- Návrat z přerušení pak může realizovat instrukce (*BX LR nebo BX (reg)*, POP {PC}, LDR nebo LDM.
- Spodní čtyři bity **x** speciální adresy EXC_RETURN určují, zda návrat je do souboru instrukcí ARM nebo Thumb včetně použitého ukazatele zásobníku SP (MSP nebo PSP).
- Přehled přerušení potřebných v předmětu MAM je uveden v tabulce na stránce 23. Celá tabulka je v Cortex-M4_Reference manuálu F401.pdf.

ZATÍŽENÍ PROCESORU PŘERUŠOVACÍM SYSTÉMEM - OBECNĚ



ADRESY PŘERUŠOVACÍHO SYSTÉMU ARM

STM32F401 Výběr z tabulky vekrotů přerišení					
Pozice	**Priorita**	**Typ priority**	**Zkratka**	**Popis**	**Adresa obsluhy**
	-	-	-	Rezervováno (nastavení SP,)	0x0000 0000
	-3	neměnný	Reset	Nulování procesoru	0x0000 0004
	-2	neměnný	NMI	Nemaskované přerušení, Clock security System	0x0000 0008
	-1	neměnný	HardFault	Všechny skupiny chyb	0x0000 000C
	3	volitelný	SVCall	Volání systémových rutin přes SWI instrukci	0x0000 002C
	6	volitelný	Systick	Systémový časovač	0x0000 003C
0	7	volitelný	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	volitelný	EXTI16/PVD	EXTI16 line 16/ PVD přes EXTI line	0x0000 0044
3	10	volitelný	EXTI22/RTC_WKUP	RTC Wakeup přes EXTI line	0x0000 004C
5	12	volitelný	RCC	RCC globalní přerušení	0x0000 0054
6	13	volitelný	EXTI0	EXTI line0 přerušení	0x0000 0058
7	14	volitelný	EXTI1	EXTI line1 přerušení	0x0000 005C
8	15	volitelný	EXTI2	EXTI line2 přerušení	0x0000 0060
9	16	volitelný	EXTI3	EXTI line3 přerušení	0x0000 0064
10	17	volitelný	EXTI4	EXTI line4 přerušení	0x0000 0068
18	25	volitelný	ADC	ADC1 globální přerušení	0x0000 0088
23	30	volitelný	EXTI9-5	EXTI Line[9:5] přerušení	0x0000 009C
28	35	volitelný	TIM2	TIM2 globální přerušení	0x0000 00B0
29	36	volitelný	TIM3	TIM3 globální přerušení	0x0000 00B4
31	38	volitelný	I2C1_EV	I2C1 událostní přerušení	0x0000 00BC
32	39	volitelný	I2C1_ER	I2C1 chybové přerušení	0x0000 00C0
35	42	volitelný	SPI 1	SPI1 globální přerušení	0x0000 00CC
36	43	volitelný	SPI 2	SPI2 globální přerušení	0x0000 00D0
37	44	volitelný	USART1	USART1 globální přerušení	0x0000 00D4
38	45	volitelný	USART2	USART2 globální přerušení	0x0000 00D8
40	47	volitelný	EXTI15-10	EXTI Line[15:10] přerušení	0x0000 00E0

ADRESY PŘERUŠOVACÍHO SYSTÉMU ARM

- Obsluhy přerušení mají přiřazeny pevné adresy.
- Tabulku adres obsluh přerušení lze v paměti přemístit na základě hodnoty v registru **VTOR** (Vector Table Offset Register) do programové nebo datové paměti.
- Velikost posunu je dána počtem použitých přerušení zvětšeným o hodnotu 16 (systémová přerušení). Hodnota možných posunů tabulky vektorů jednotlivých přerušení je dána hodnotu

$$2^n \geq (\text{počet\_přerušení}+16)*4.$$

- Adresy jsou od sebe vzdáleny o 4 byty (dvě slova). V tabulce je **uložena adresa obsluhy přerušení zvětšená o hodnotu 1**.
- Některé zdroje přerušení mají svoji **samostatnou adresu**, některé ji mají **sdruženou**. V případě **sdružené adresy** je potřeba v obsluze přerušení programově zjistit, které z přerušení žádá o obsluhu. Pořadí v testu jednotlivých žádostí určuje, které žádosti bude vyhověno nejdříve.

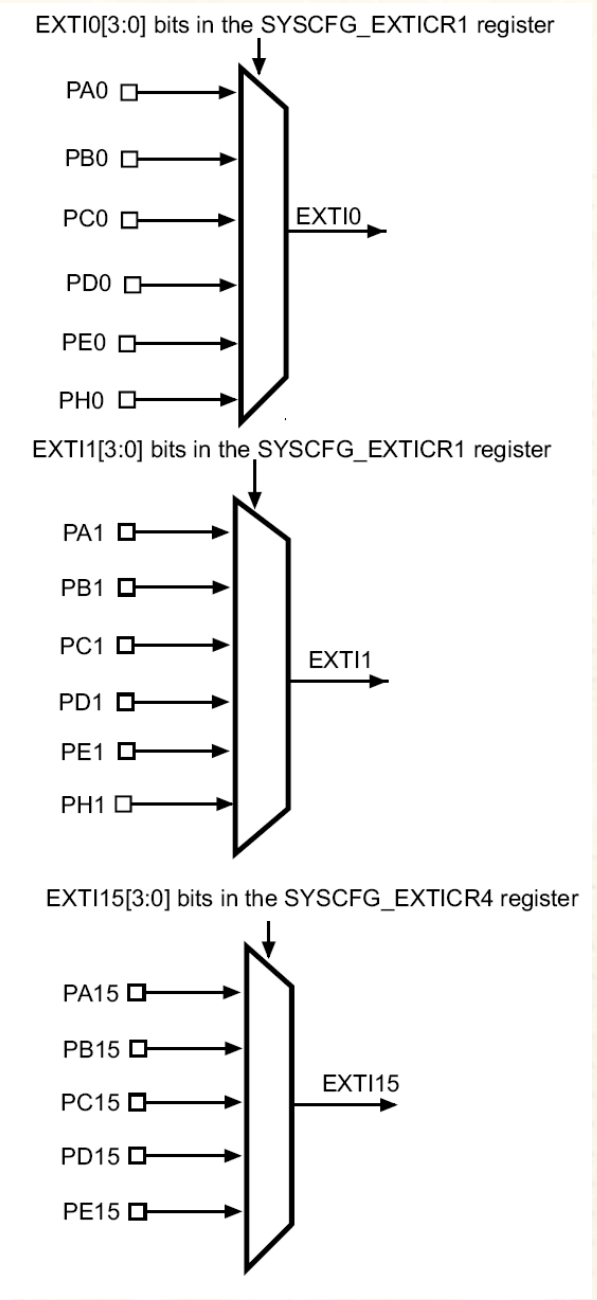
VNĚJŠÍ PŘERUŠENÍ PROCESORŮ ARM M3 A M4

Každý vývod GPIOx může být zdrojem vnějšího přerušení viz. obrázek. Který bit n z GPIOx bude zdrojem externího přerušení EXTIn stanovují registry **SYSCFG_EXTICRy** y∈(1;4). Čtyři bity registru určují pro každé EXTIn přiřazení daného vývodu dané brány.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

0000: vývod PAn; 0001: vývod PBn;
0010: vývod PCn; 0011: vývod PDn;
0100: vývod PEn; 0101: Rezervován
0110: Rezervován; 0111: vývod PHn;

Z blokového schématu vyplývá, že EXTIn (n=0 až 22) může představovat obvodové a softwarové přerušení nebo událost.



VNĚJŠÍ PŘERUŠENÍ NEBO UDÁLOST PROCESORŮ ARM M3 A M4

Každé externí přerušení/událost má v registrech **EXTI_RTSR** a **EXTI_FTSR** bit TRn určující reakci na příslušnou hranu. Log.0 reakci zakazuje a log.1 povoluje. S nastavením těchto bitů je třeba

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	Reserved		TR18	TR17	TR16
									rw	rw			rw	rw	rw

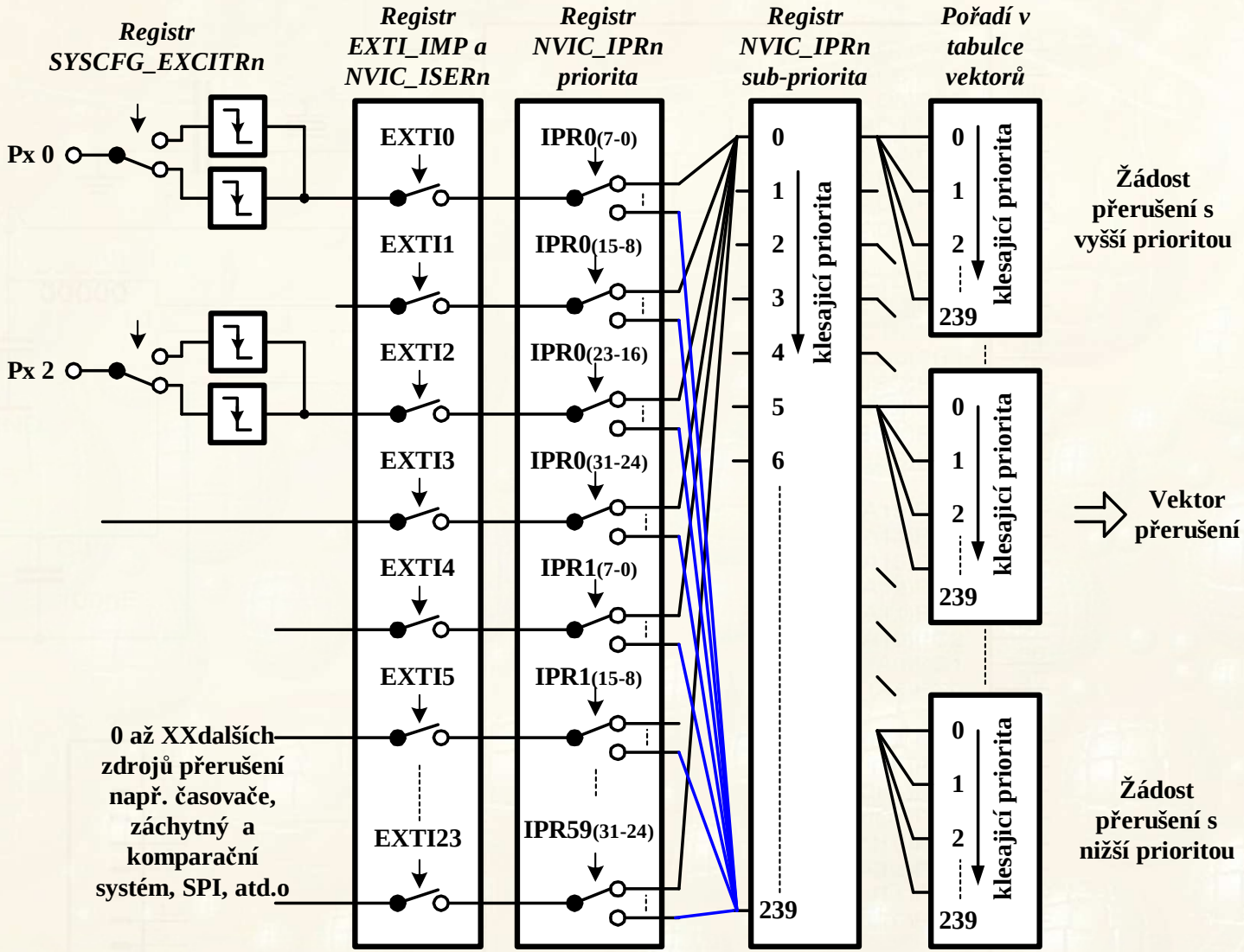
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

nastavit bit v registru **EXTI_IMR** (Interrupt mask)/**EXTI_EMR** (Event mask) a nastavit řídící kontrolér **NVIC IRQ.** Rozmístění a funkce bitů MRn v registru **EXTI_IMR / EXTI_EMR** určuje: 0-žádost od vývodu n je maskována, 1-žádost n není askována.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	Reserved		MR18	MR17	MR16
									rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

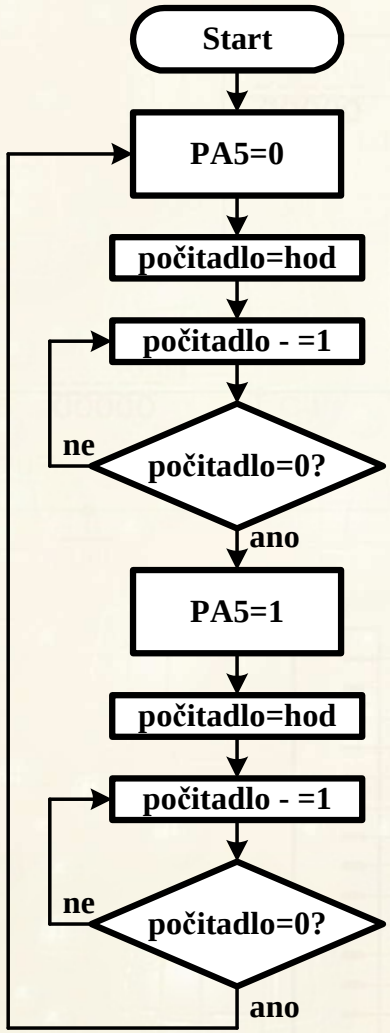
PŘERUŠOVACÍ SYSTÉM ARM



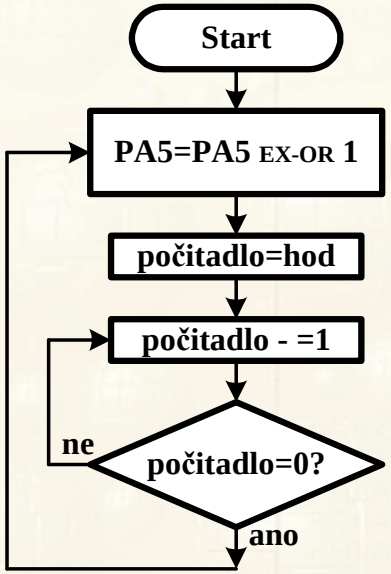
Přerušovací systém umožňuje každému přerušení přiřadit libovolnou úroveň priority nebo subpriority a realizovat tak potřebná vnořená přerušení mimo pevné priority (Reset, NMI, Hard fault).

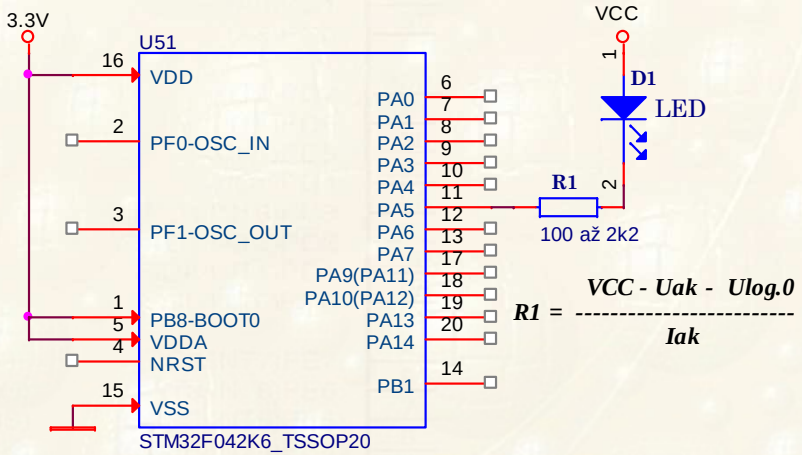
PŘÍKLAD NA VYUŽITÍ PŘERUŠOVACÍHO SYSTÉMU

Příklad: *Blikání diodou LED připojenou k vývodu PA5 procesoru z řady ARM v 0,25s intervalu.*

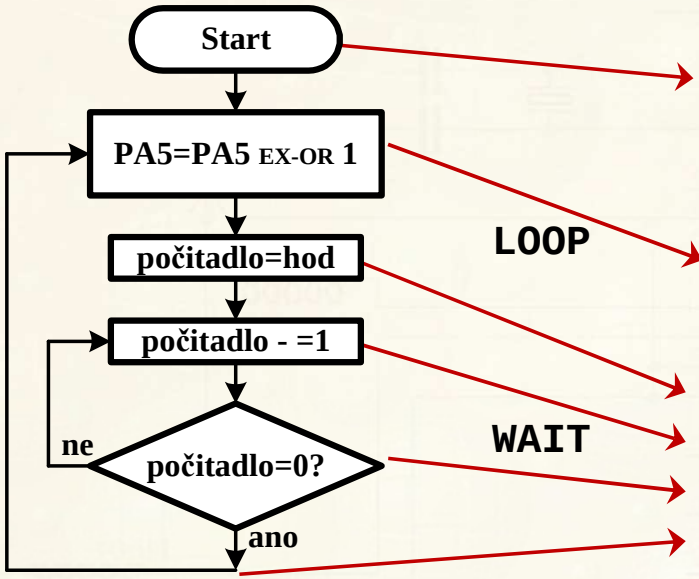


- Jedno z možných připojení LED k bráně µP je zobrazeno. Při připojení **nesmí** být překročen maximální proud pro V/V vývod procesoru, ani maximum pro součet proudů vývody celé brány procesoru (zde PA).
- Programové řešení vychází z vývojového diagramu realizující programové zpoždění 0,25s.



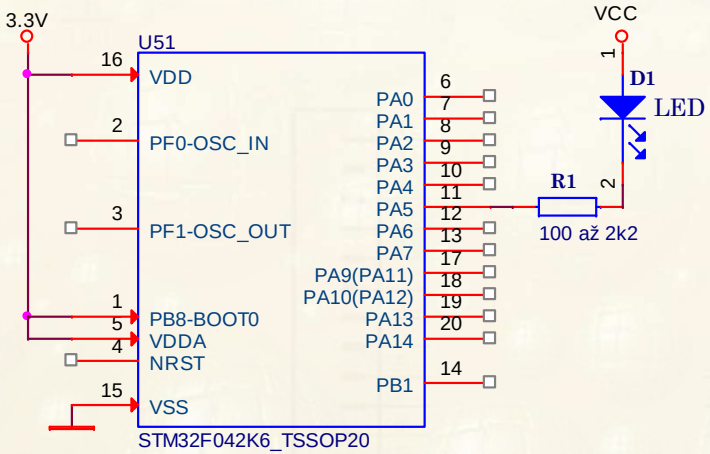


BLIKÁNÍ DIODOU LED – PROGRAMOVÉ ŘEŠENÍ



Program pro řadu ARM

```
Inicializace systémových hodin na 8MHz
Nastavení vývodu PA5 na výstup Push-Pull
LDR  R2, =GPIOA_ODR ; Adresa brány A_ODR do R2
LDR  R1, =0x00000020; Hodnota pro EX-OR
LDR  R0, [R2]       ; Načtení obsahu brány PA
EOR  R0,  R1        ; EX-OR PA5 s log.1
STR  R0, [R2]       ; Zápis do brány PA
LDR  R3, =0x0007A11F; Hodnota počitadla
SUBS R3, R3,#1      ; Odecteni 1
BNE  WAIT           ; Skok není-li R3=0
B    LOOP           ; Opakování
```



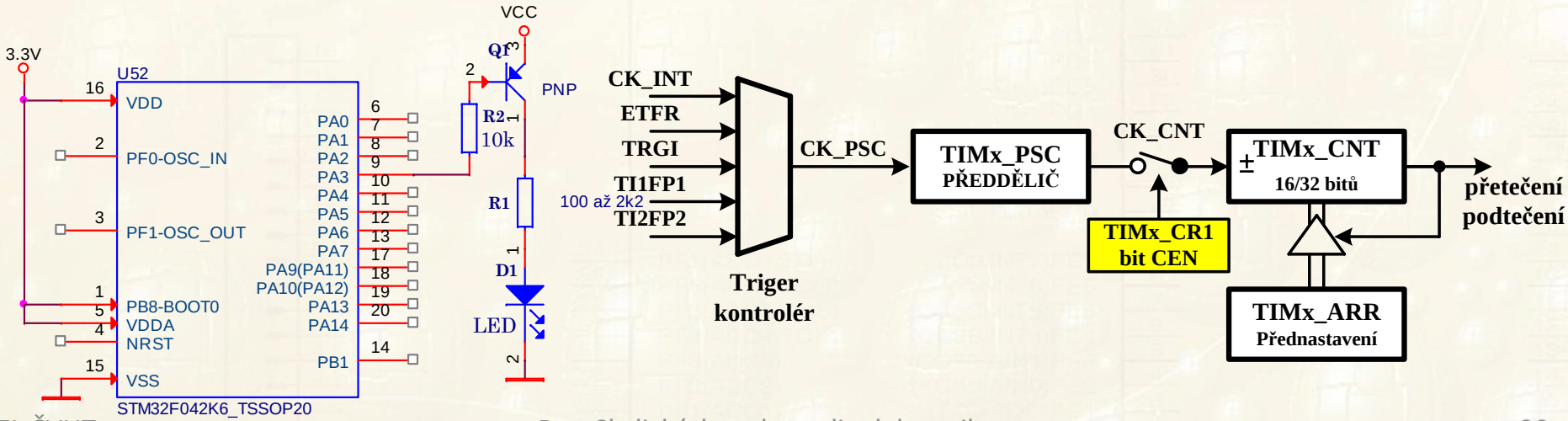
Určení hodnoty **R3** pro zpoždění 0,25s.

- Při hodinovém kmitočtu 8MHz ⇒ strojový cyklus =125ns. Instrukce LDR, STR s pamětí trvají 2cykly, skok 3 cykly a ostatní jsou jedno cyklové. Celá smyčka WAIT 4 cykly.
- Počitadlo+1=250000/(4*0,125)≈500000
- Do počitadla (registru R3) uložíme hodnotu 0x7A11F.

BLIKÁNÍ DIODY LED S POUŽITÍM ČÍTAČE/ČASOVAČE

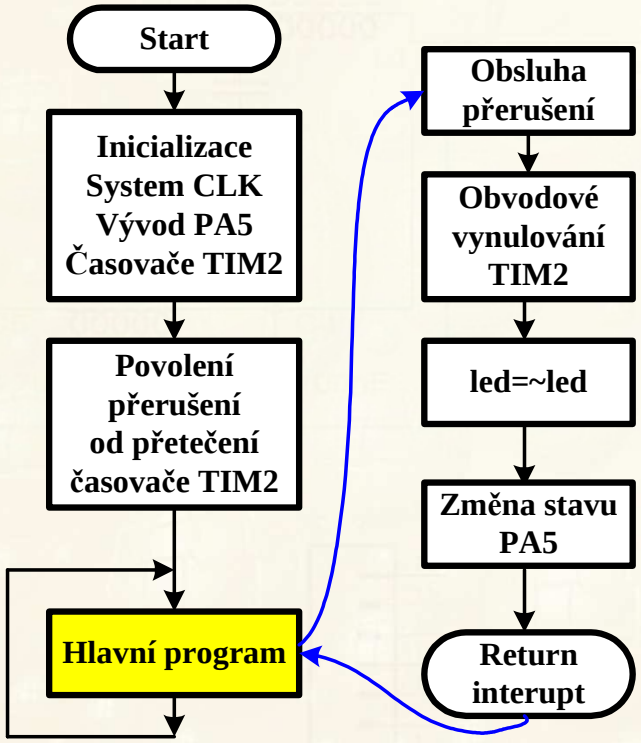
Programové zpoždění ⇒ **spotřebuje 100%** výpočetního výkonu procesoru. Jiná efektivnější a přesnější řešení.

- Zpoždění realizované úseky hlavního programu – **problémy?**
- Zpoždění realizované násobky hlavního programu – **problémy?**
- Využití prostého čítače/časovače a přerušovacího systému
  - Čítač 8,16 nebo 32 bitový čítá interní oscilátor nebo signál z externího vstupu.
  - Při přetečení čítače/časovače (přechod do stavu 0000h z naprogramovaného maxima nebo FFFFh) bude nastaven příznak žádosti o přerušení. Bude-li žádost povolena dojde k zavolání obsluhy **přerušení** – **přesnost?**



BLIKÁNÍ DIODY LED S POUŽITÍM ČÍTAČE/ČASOVAČE

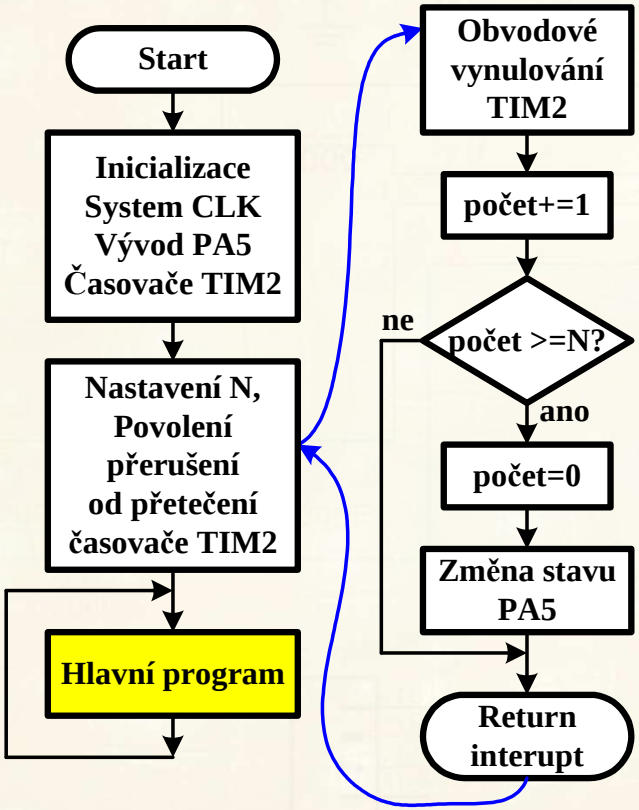
Pro zvolený časovač a čítaný kmitočet zvolíme hodnotu předděliče a nejvyšší hodnotu časovače tak, aby bylo dosaženo požadovaného časového intervalu. Dosáhne-li časovač mezní hodnoty, dojde k jeho vynulování a generování žádosti o přerušení.




```
main()  int main (void)
{       SystemCoreClockSetHSI();
        SystemCoreClockUpdate();  // Set Clock
        // Povoleni hodinového signálu pro GPIOA
        RCC->AHBENR |=  RCC_AHBENR_GPIOAEN;
        PIN_OUTPP_Initialize (GPIOA,3);
        LED_Initialize();
        TIM2_Inicializace();
Hlavni: while(1) // Nekonečná smyčka hlavního programu
        {   . . .  }
}
//      PODPROGRAM PŘERUSENÍ
void TIM2_IRQHandler()
{       // Nulovaní příznaku události od časovače TIM2
        TIM2->SR &= ~TIM_SR_UIF;
        // Změna stavu diody LED
        if (led != 0)  { led = 0; LED_On(3);}
        else  {led = 1; LED_Off(3);}
```

Z 250000 strojových cyklů bude cca 62 cyklů potřeba na změnu stavu diody. Procesor může z 99,975 % pracovat na hlavním programu. **Přesnost?**

GENEROVÁNÍ DLOUHÉ PRODLEVY S POUŽITÍM ČÍTAČE/ČASOVAČE



Pokud nemůžeme dosáhnout požadované prodlevy pro změnu stavu můžeme:

- Nastavit vyšší hodnotu předděliče čítače.
- Použít další časovač jako zdroj hodinového signálu pro používaný časovač.
- Použít externí zdroj hodinového signálu.
- Upravit obsluhu přerušení časovače zavedením počitadla **počet**, určujícího kolikrát byla přerušovací rutina zavolána. Generovaný interval pro změnu stavu pak vytvoříme jako celočíselný násobek doby mezi žádostmi o přerušení.