

Vstupy a výstupy jednočipového mikroprocesoru

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) 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 V/V jednočipových procesorů pro maximální přizpůsobení k ovládání realizované aplikace.

Každý vývod je konfigurovatelný do

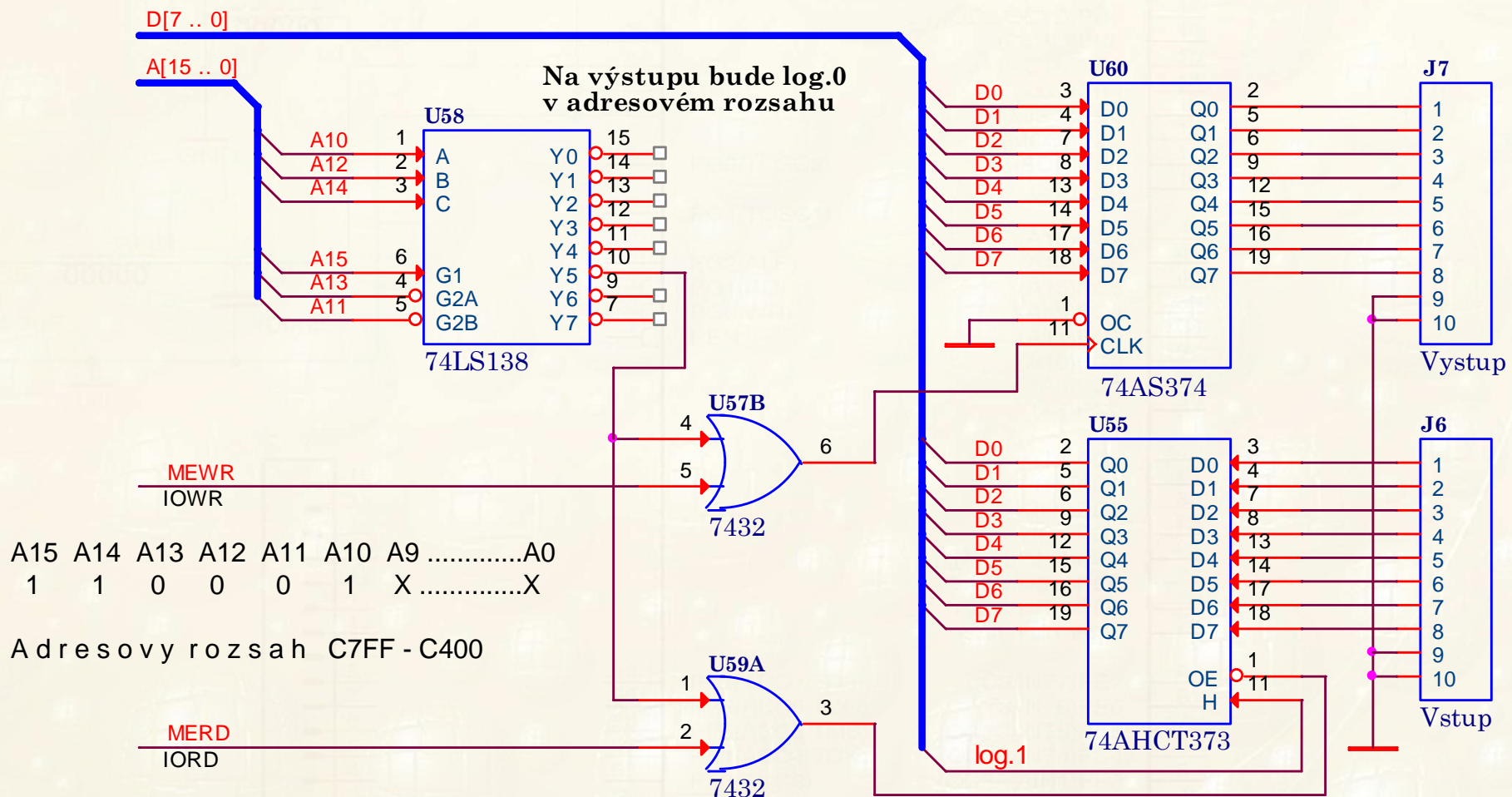
- Vstupního nebo výstupního režimu
- Stavů pro měření analogové veličiny (je-li vybaven A/D)
- U některých μP upnutí vývodu přes odpor k napájení nebo k zemi.
- Řízení doby náběžné a sestupné hrany (přechod $0 \rightarrow 1$ a obráceně).

VSTUPY A VÝSTUPY MIKROPROCESOROVÉHO SYSTÉMU

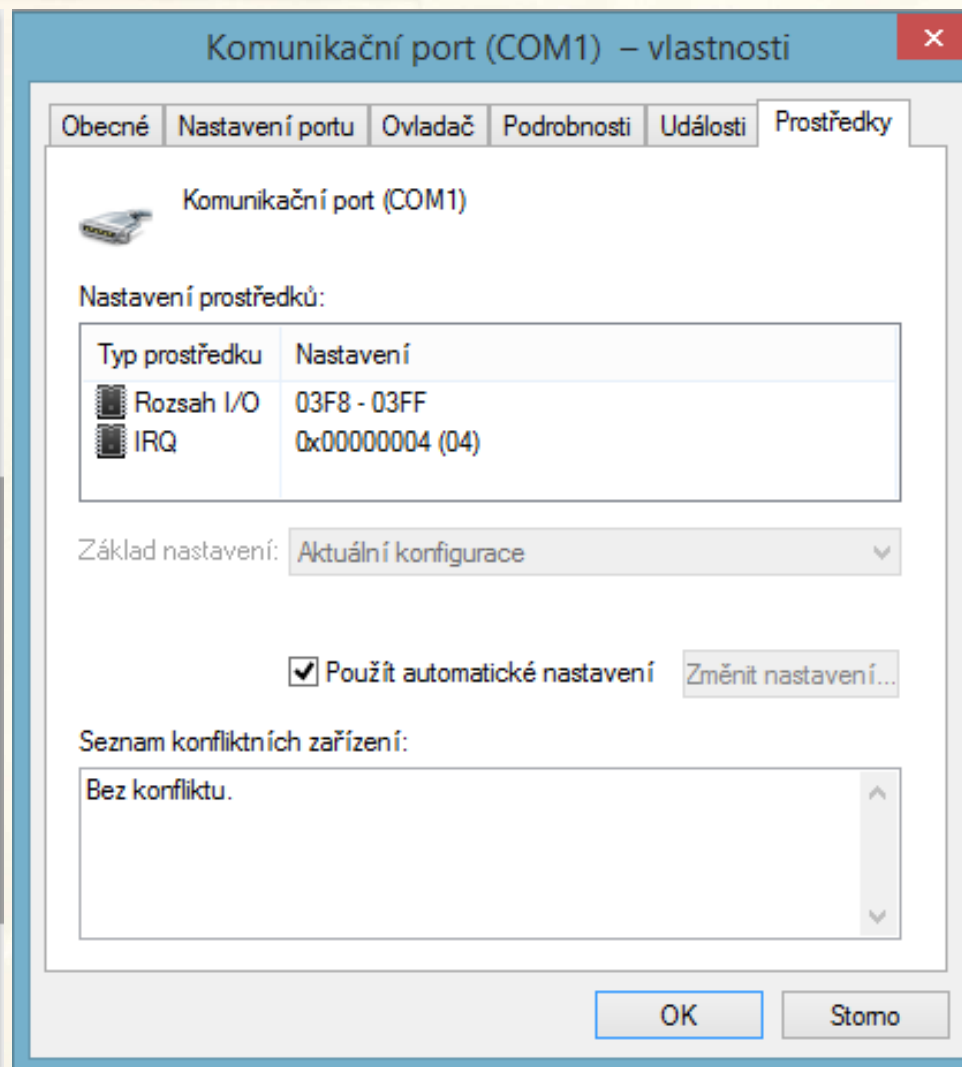
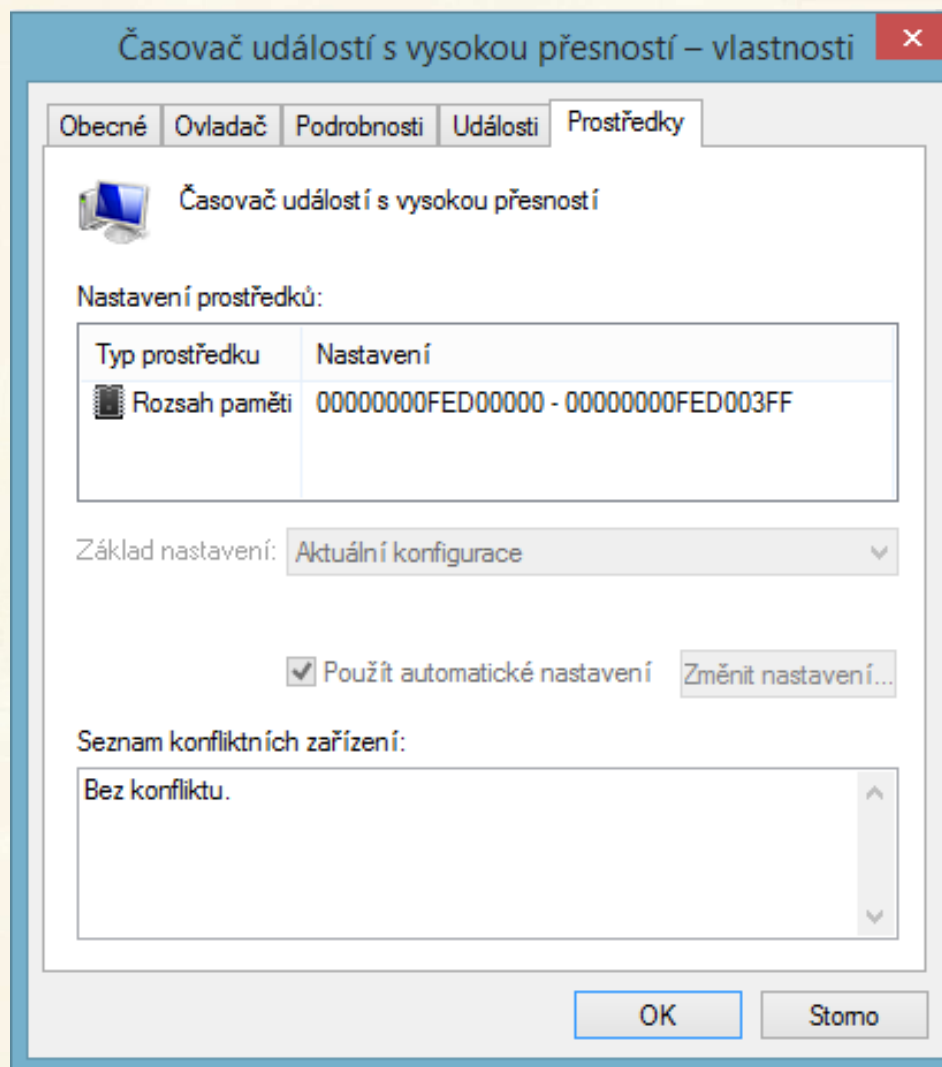
Připojení vstupu a výstupu do adresového prostoru C7FF÷C400.

Signály MEWR a MERD – paměťově mapovaná periférie

Signály IOWR a IORD – periférie ve vstupně-výstupním prostoru



MAPOVÁNÍ PERIFERIÍ V POČÍTAČI PC



Paměťově mapovaná periferie

periferie ve vstupně-výstupním prostoru

MAPOVÁNÍ PERIFERIÍ V POČÍTAČI PC

The image displays two side-by-side screenshots of Windows Device Manager property windows, illustrating memory mapping for hardware devices.

Left Window: Most Intel(R) 82801 sběrnice PCI - 244E – vlastnosti

- Tab: Prostředky
- Typ prostředku: Most Intel(R) 82801 sběrnice PCI - 244E
- Nastavení prostředků:

Typ prostředku	Nastavení
Rozsah paměti	00000000FEB00000 - 00000000FEBFFFFF
Rozsah I/O	E000 - EFFF
- Základ nastavení: (dropdown menu)
- Použít automatické nastavení
- Seznam konfliktních zařízení: Bez konfliktu.
- Buttons: OK, Storno

Right Window: Řadič sběrnice SMBus čipové sady řady Intel(R) ICH9 ...

- Tab: Prostředky
- Typ prostředku: Řadič sběrnice SMBus čipové sady řady Intel(R) ICH9 - 2930
- Nastavení prostředků:

Typ prostředku	Nastavení
Rozsah paměti	00000000F9FFF400 - 00000000F9FFF4FF
Rozsah I/O	0400 - 041F
IRQ	0x00000005 (05)
- Základ nastavení: (dropdown menu)
- Použít automatické nastavení
- Seznam konfliktních zařízení: Bez konfliktu.
- Buttons: OK, Storno

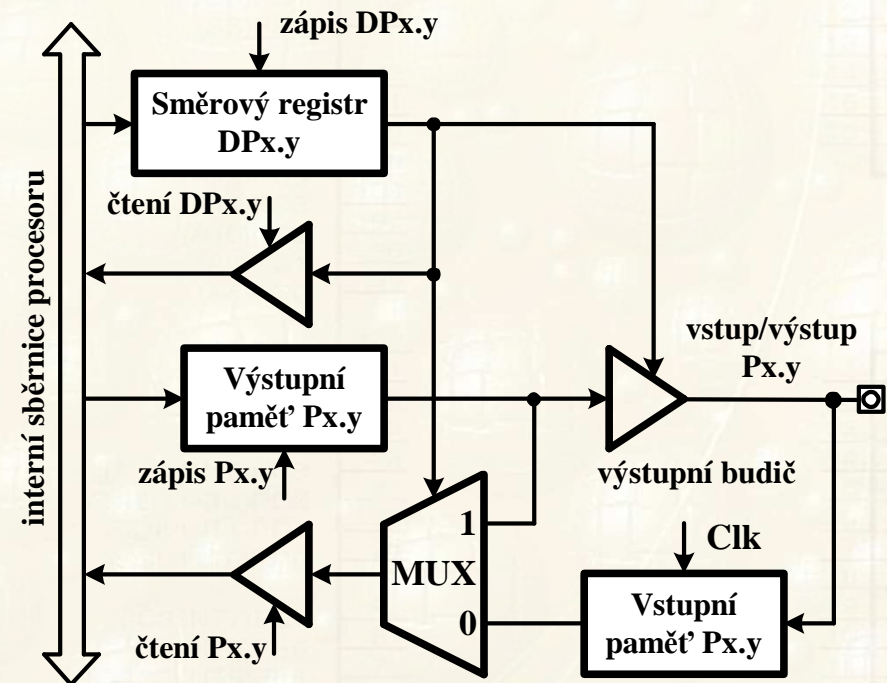
Paměťově mapované periferie se stránkováním a pravá spojená s přerušovacím systémem.

VSTUPY A VÝSTUPY JEDNOČIPOVÝCH MIKROPROCESORŮ

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

- ❖ **Paměťového členu**
- ❖ **Multiplexeru** pro zpětné čtení zapsané hodnoty do výstupu nebo hodnoty přivedené na vývod brány.
- ❖ **Směrového registru** určujícího vývod jako vstup nebo výstup.
- ❖ **Třístavových budičů** připojených k datové sběrnici a vývodu V/V.
- ❖ **Synchronizaci** vstupního signálu zajišťuje **dvojnásobný** synchronizátor.

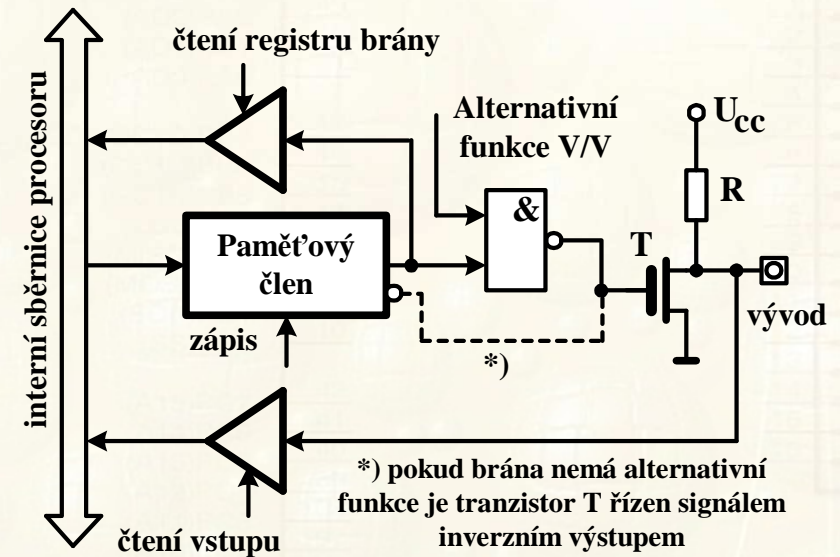
Nulování procesoru – směrový registr nastaven do **stavu vysoké impedance** nebo **vstupního režimu**.



VSTUPY A VÝSTUPY JEDNOČIPOVÝCH MIKROPROCESORŮ

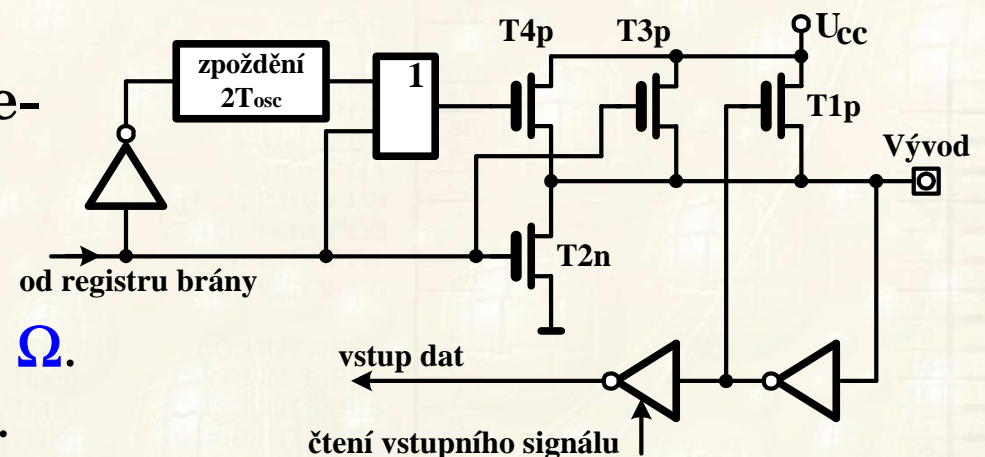
Pseudoobousměrná brána (obvod s otevřeným kolektorem) se skládá z:

- ❖ Paměťového členu
- ❖ Slučovače výstupu s alternativním funkcí (periferií)
- ❖ Výstupního tranzistoru
- ❖ Oddělovače pro čtení vstupu
- ❖ Oddělovače pro čtení zapsané hodnoty v paměťovém členu.



Zpětné čtení stavu výstupu souvisí s připojením přechodu **báze-emitor**.

Odpor (T1p) - 30÷ 50kΩ, při přechodu 0→1 snížen (T4p) na **500 Ω**.
na dvě periody hodinového signálu.



8051 - Výstup s otevřeným kolektorem (OK)

log.0 – pozor na připojení výkonného zdroje

log.1 – výstup je schopen poskytnout proud cca $140\mu\text{A}$

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

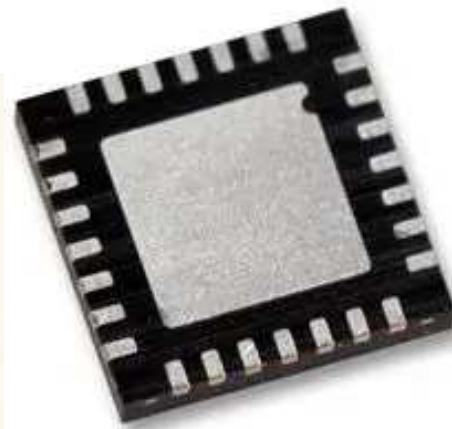
U vývodů je volitelná rychlost přeběhu, upnutí k napájení (Pull-UP) nebo k zemi (Pull-DOWN) a nastavení stavu po připojení napájení.

JEDNOČIPOVÝ MIKROPROCESOR S PAMĚTÍ EPROM, OTP A FLASH

DIL

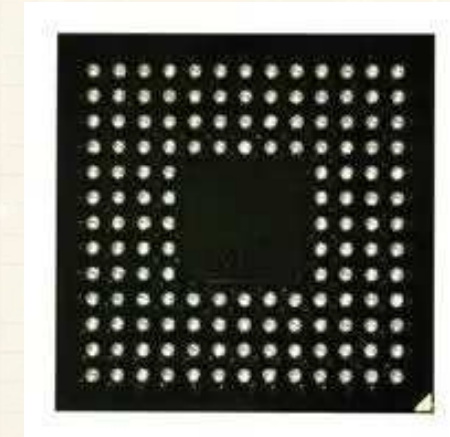


PLCC



UFQFN

TFBGA



TSSOP



TQFP

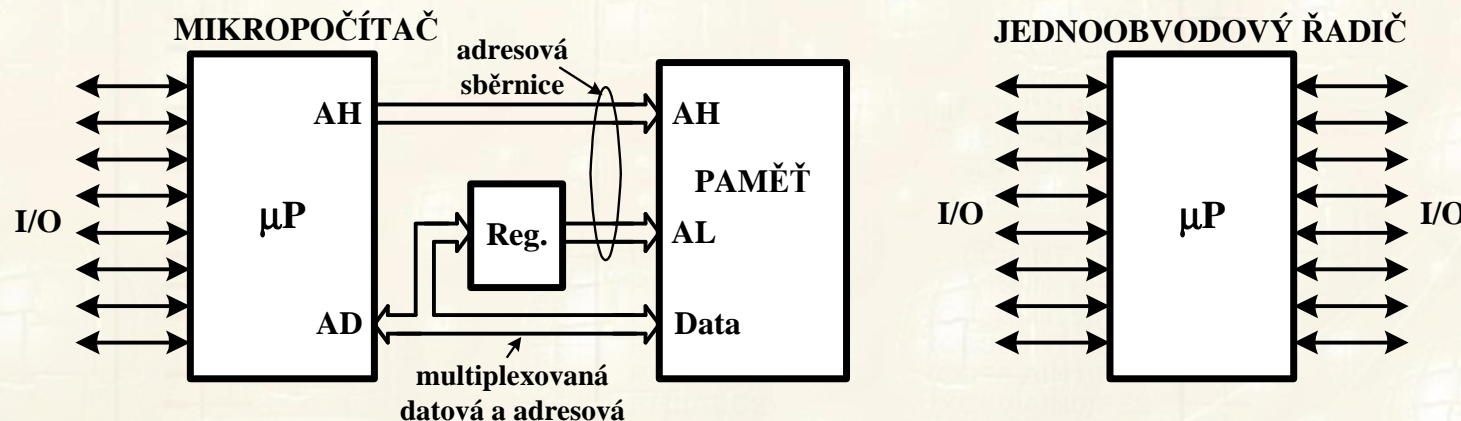
JEDNOOBVODOVÉ MIKROPROCESORY A ŘADIČE

Historicky se ustálilo rozdělení jednočipových procesorů na

- ❖ **Mikroprocesory** - V/V vývody je možné využívat k
 - **připojení periférií**
 - **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 systémem (**Interrupt systém**) podstatně zjednodušuje a zefektivňuje styk s vnitřními i vnějšími periferiemi. Přerušování je takový „zvonek“ .

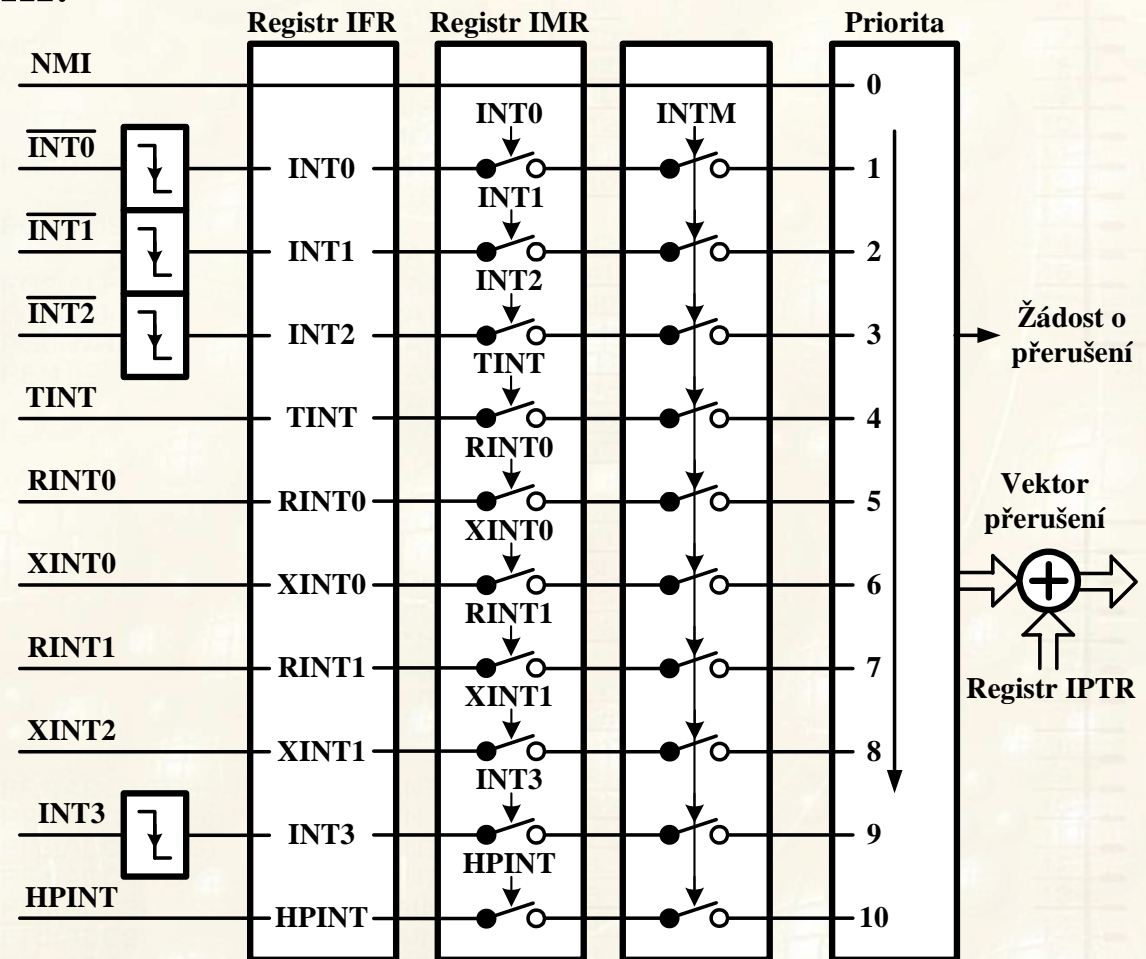
- ❖ Obsluha periferií bez přerušovacího systému
 - Snížení výpočetního času pro hlavní program
 - Program rozšířen o pravidelné testování žádostí
 - Obtížné zpracování **v potřebných intervalech**
- ❖ Obsluha s využitím přerušovacího systému
 - Periferie požádá o přerušování
 - Procesor uloží žádost ke zpracování
 - Testuje zda přerušování od dané periferie je povolené
 - Vyhodnotí zda není jiná žádost s vyšší prioritou
 - Přerušuje ve vhodném okamžiku probíhající program (dokončí právě rozpracovanou instrukci nebo instrukce („pipeline“))
 - Přejde ke **zpracování obslužného programu**
- ❖ Žádost periferie o přerušování je vyvolána obvodově obvykle pomocí instrukce **CALL adresa_přerušeni**.
- ❖ **Rozdíl mezi CALL podprogram a Zavolání přerušeni?**

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 přerušeni a bit (**interrupt mask**) povolující nebo zakazující jeho přerušeni.
- ❖ **Global mask** – povolující všechna povolená přerušeni nebo zakazující všechna přerušeni.
- ❖ Priority nastaveny výrobcem s ohledem na předpokládané použití.
- ❖ Vyhodnoceni dvou žádostí ve stejnou dobu
- ❖ **Nested interrupt**
- ❖ **Nemaskované přerušeni (NMI)**
- ❖ Změna priorit
- ❖ Přeadresování vektorů přerušeni



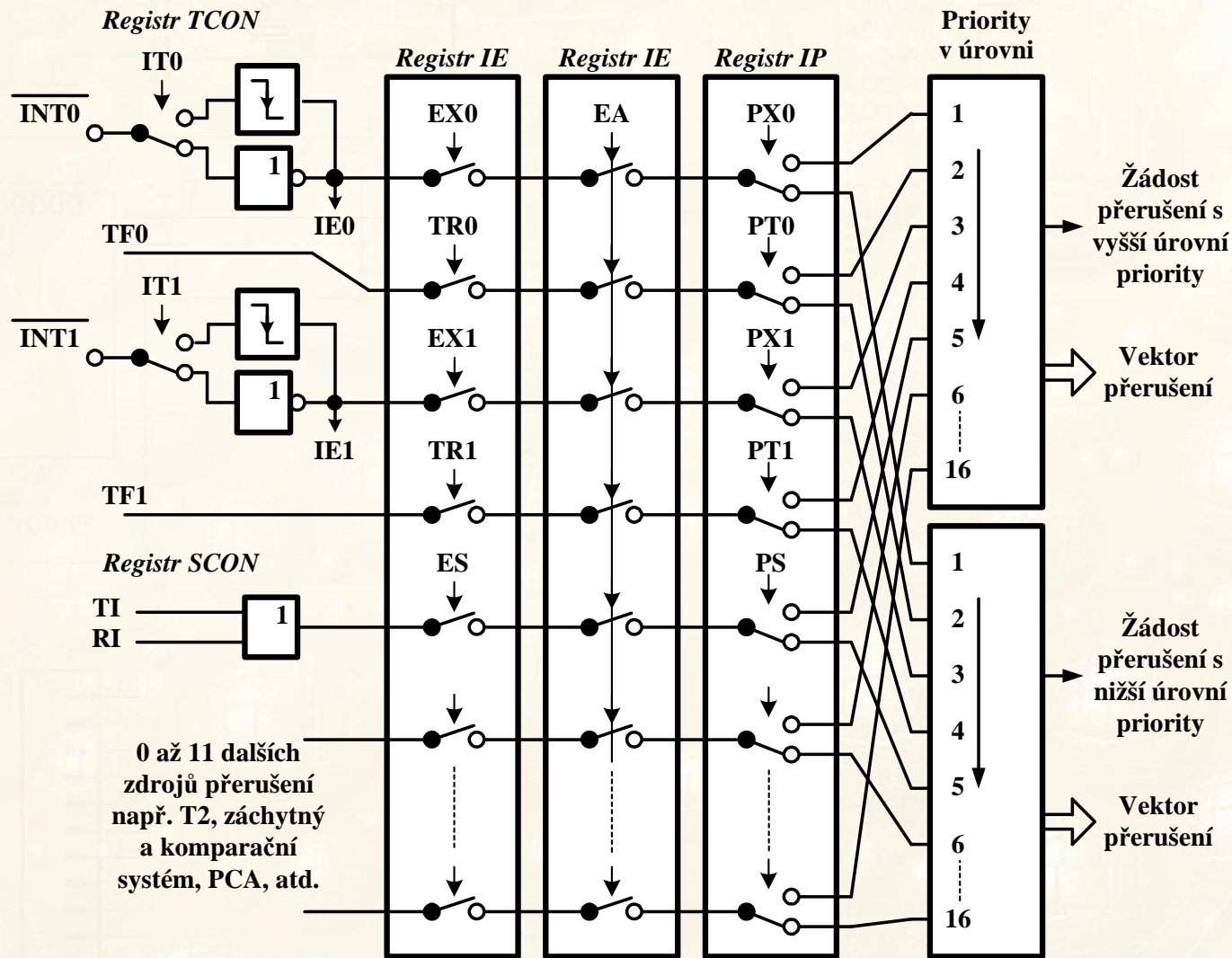
Je-li v aplikaci více zdrojů přerušení, mohou nastat problémy s dobou přístupu do zpracování jednotlivých obsluh. Volba procesoru pak ovlivňuje:

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

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

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

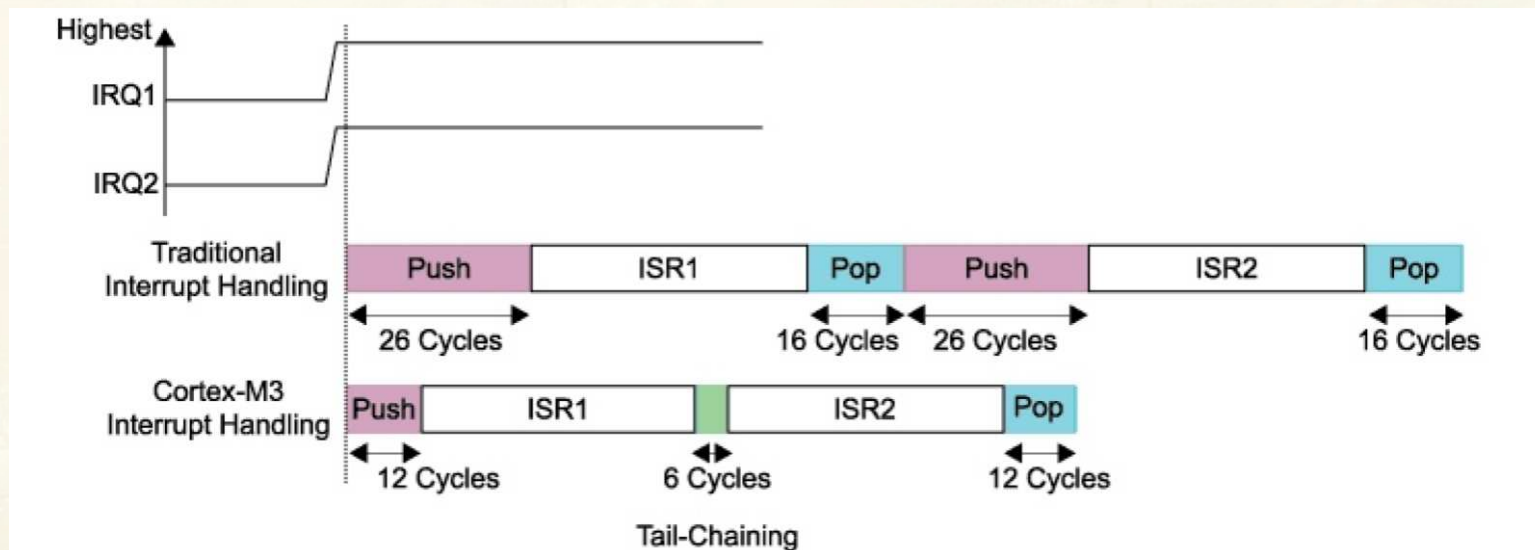
PŘERUŠOVACÍ SYSTÉM S VOLITELNOU ÚROVNÍ PRIORITY



- **Více úroňový systém s pevně danými prioritami**
 - Přerušeni s jakoukoliv prioritou lze přesunout do vyšší úrovně priority, z které přeruší i přerušeni s vyšší prioritou umístěné v nižší úrovni priorit – tzv. **přerušeni v přerušeni**.
- ❖ **Žádosti o přerušeni** se testují v definovaném okamžiku strojového cyklu procesoru a posléze se vyhodnocují.
- ❖ **Vyhodnoceni** – je-li přerušeni povoleno, není další žádost přerušeni s vyšší prioritou a je povolený přerušovací systém.
- ❖ **Vyvoláni přerušeni** – instrukcí **LCALL** adresa přerušeni nebo přečtením adresy. Dojde k uložení návratové adresy, přenesení adresy přerušeni do PC, zakázání přerušovacího systému, případně uloženi některých registrů).
- ❖ **Zpracováním přerušeni** nesmí být ovlivněny hodnoty **registrů** a **příznaků** před přechodem do přerušeni.

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



ZATÍŽENÍ PROCESORU PŘERUŠOVACÍM SYSTÉMEM - OBECNĚ

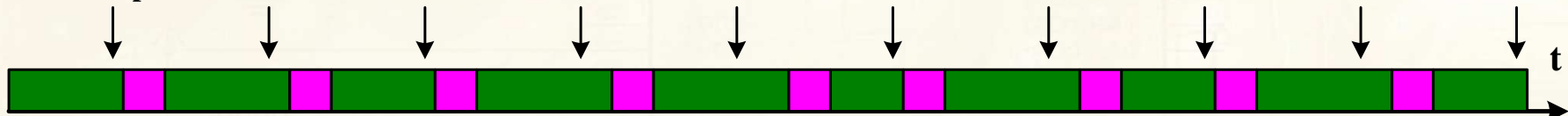
Žádné žádosti o přerušení

100% výpočetního výkonu pro hlavní program



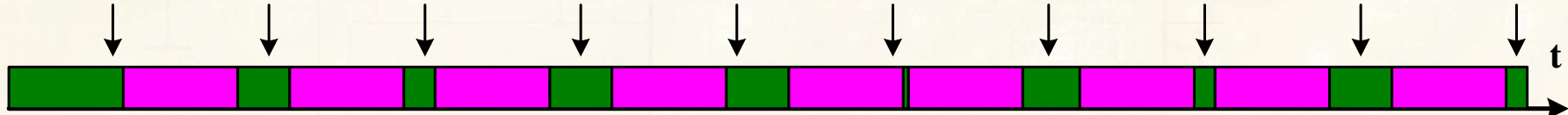
Žádosti o přerušení

73% výpočetního výkonu pro hlavní program



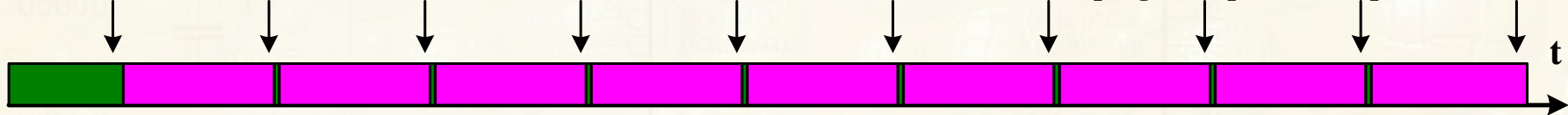
Žádosti o přerušení

27% výpočetního výkonu pro hlavní program



Žádosti o přerušení

1 instrukce hlavního programu po každém přerušení



Žádosti o přerušení

Vyšší úroveň vyšší priorita nižší priorita

Nižší úroveň vyšší priorita nižší priorita

Chyba



Žádosti o přerušení

Vyšší úroveň priorita = 4

Nižší úroveň priorita = 0 priorita = 2 priorita = 3

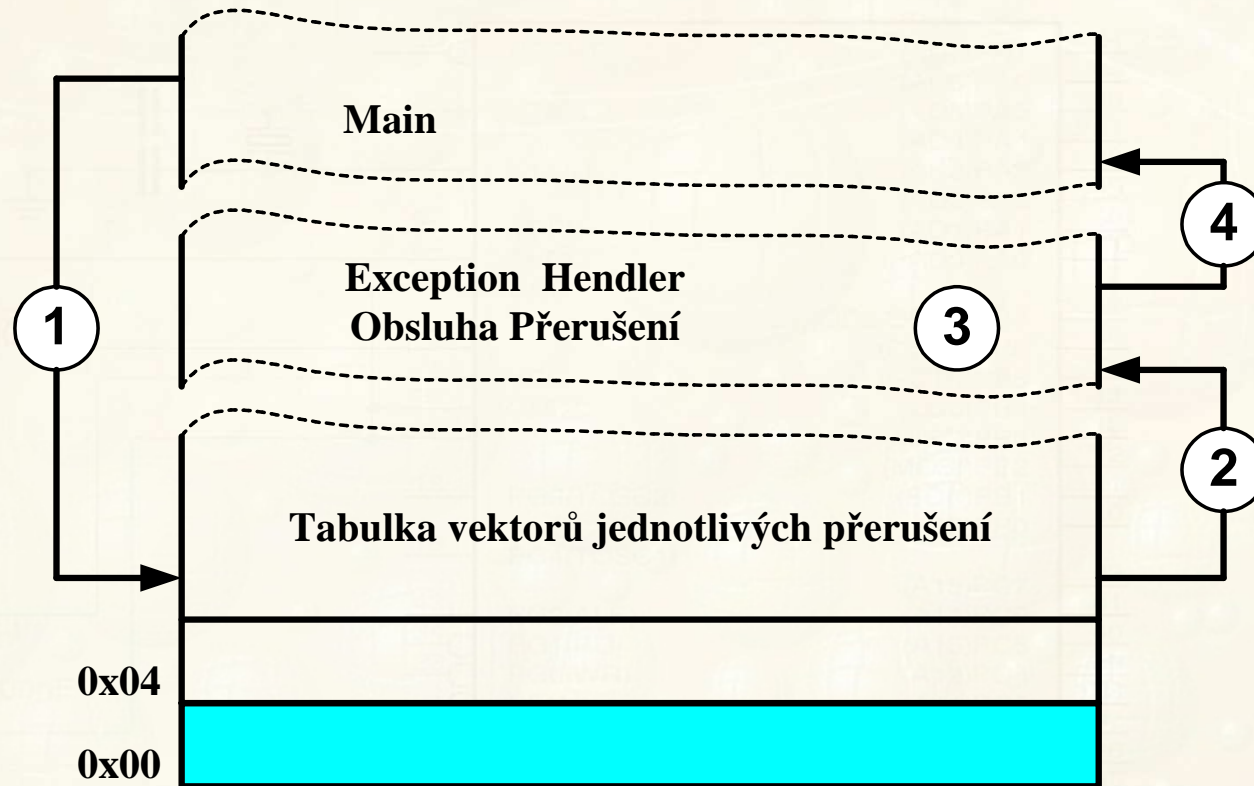


Banka registrů 0 2 0 2 0 1 0 2 1 0 10 2 0 1 0 2 0 1 0 2 0 1 10 2 0 1 0 2 0 1 0 2 0 1 0 2

Zásady vhodné dodržování v přerušovacím systému

- ❖ **Realizace přerušení by měla být co nejkratší**
- ❖ Delší obsluhu případně přesunout do main a zpracovat na základě návěští nastaveného v přerušení
- ❖ Pozor na operace s globálními proměnnými používanými i jinde v programu (přerušení může nastat kdykoliv)
- ❖ Větvení v main na základě proměnné přijaté v přerušení
- ❖ Práce v main s více bytovou proměnnou přijímanou v přerušení.
- ❖ Větvení pro více jak 3 možnosti realizovat operací switch a nikoliv if-else.
- ❖ Pozor na systém používaný pro ukládání lokálních proměnných
- ❖ Vzorkování A/D a D/A převodníků opřené o přerušovací systém ⇒ vede k **fázové nestabilitě** a zmenšení poměru S/N nebo SFDR.

CHOVÁNÍ PŘI PŘIJETÍ PŘERUŠENÍ ARM



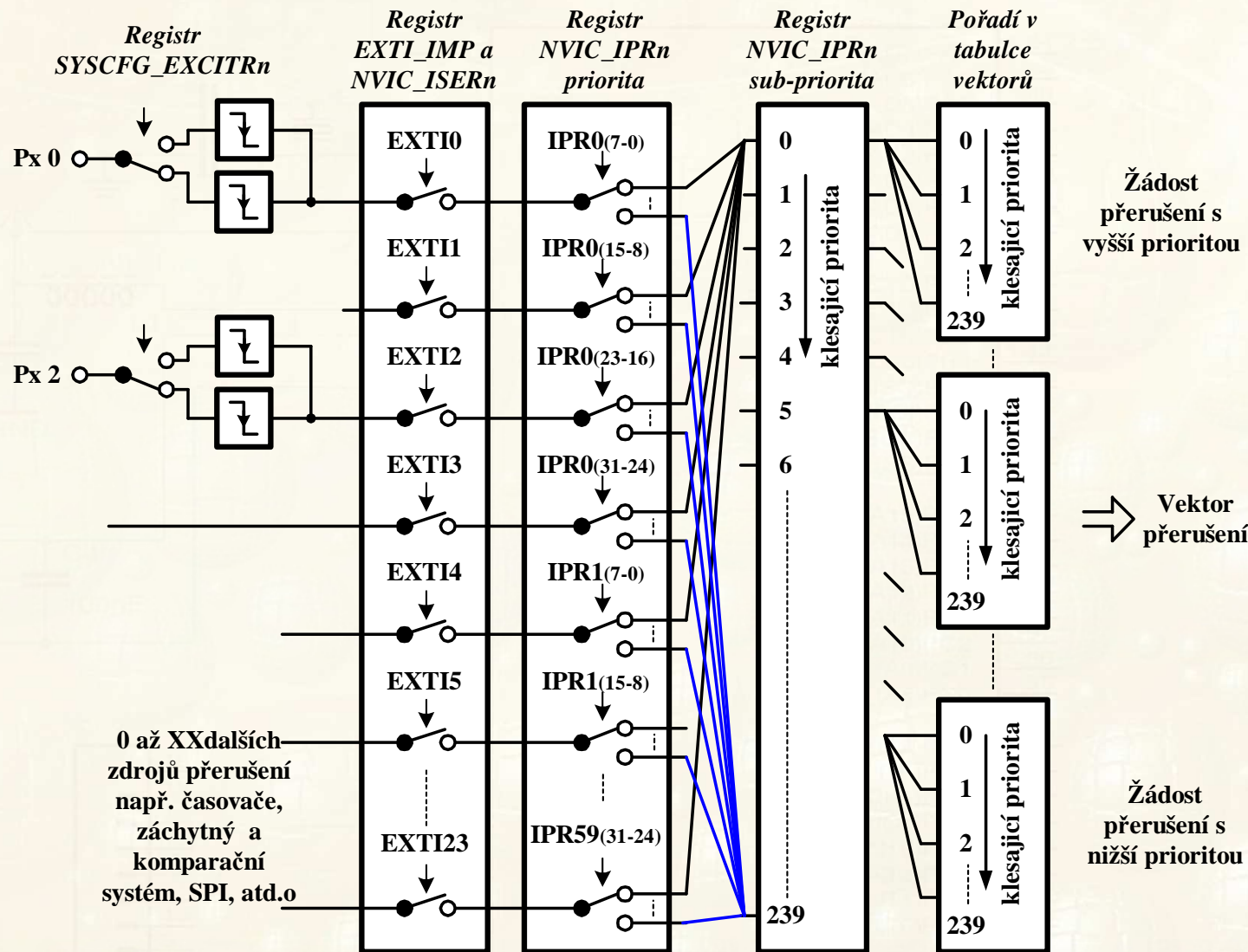
1. Přijetí žádosti o přerušení, dokončení aktuální instrukce, přístup do tabulky vektorů přerušení
2. Uložení adresy přerušení do PC
3. Spuštění obsluhy přerušení v Handler mode
4. Ukončení obsluhy a návrat do hlavního programu

- ❖ 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 (0xFFFFFFFFx), 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 31. Celá tabulka je v Cortex-M4_Reference manuálu F401.pdf.

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).

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 hodnotou
$$2^n \geq (\text{počet_přerušení} + 16) * 4.$$
- ❖ 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 **sdílenou**. V případě sdílené 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

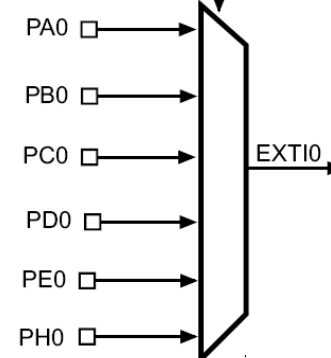
Každý vývod GPIOx může být zdrojem vnějšího přerušení viz. obrázek. Který bit n z GPIOx bude zdrojem externího přerušení $EXTIn$ stanovují registry **SYSCFG_EXTICR y** $y \in (1;4)$. Čtyři bity registru určují pro každé $EXTIn$ přiřazení daného vývodu dané brány.

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

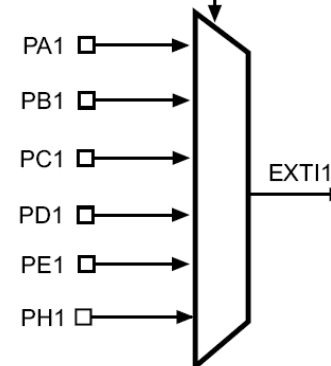
0000: vývod PAn;	0001: vývod PBn;
0010: vývod PCn;	0011: vývod PDn;
0100: vývod PEn;	0101: Rezervován
0110: Rezervován;	0111: vývod PHn;

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

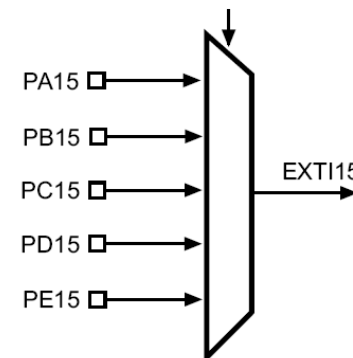
EXTI0[3:0] bits in the SYSCFG_EXTICR1 register



EXTI1[3:0] bits in the SYSCFG_EXTICR1 register

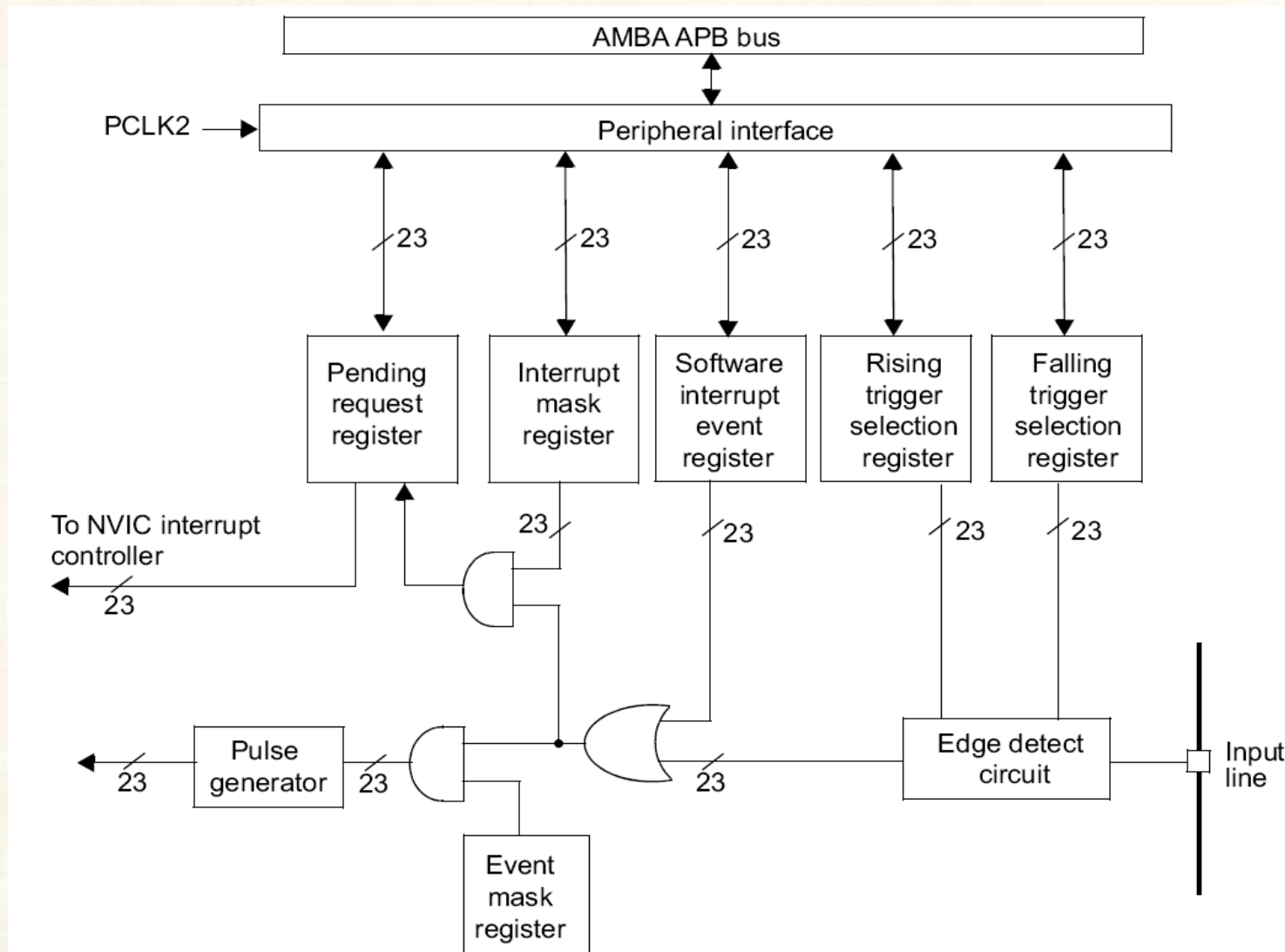


EXTI15[3:0] bits in the SYSCFG_EXTICR4 register



VNĚJŠÍ PŘERUŠENÍ NEBO UDÁLOST PROCESORŮ ARM M3 A M4

Každé přerušení nebo událost EXTIn (n=0 až 23) může být konfigurováno podle uvedeného blokového schématu.



VNĚJŠÍ PŘERUŠENÍ NEBO UDÁLOST PROCESORŮ ARM M3 A M4

Každé externí přerušení/událost má v registrech **EXTI_RTSR** a **EXTI_FTSR** bit TR_n 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 řídicí kontrolér **NVIC IRQ**. Rozmístění a funkce bitů MR_n v registru **EXTI_IMR** / **EXTI_EMR** určuje: 0-žádost od vývodu n je maskována, 1-žádost n není maskována.

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	

Je-li změna na vývodu aktivována jako **událost**, pak bit MR_n nastavujeme (log.1) v Event mask registru (**EXTI_EMR**).

Všech 23 možných žádostí může být konfigurováno jako přerušení, softwarové přerušení nebo událost po povolení v registrech **EXTI_IMR** nebo **EXTI_EMR**.

Je-li softwarové přerušení žádosti povoleno (**EXTI_EMR**), pak se do bitu registru **EXTI_SWIER** zapisuje 1, a následně i do Pending registru **EXTI_PR** jako nevyřízená žádost o přerušení.

Jeli přerušení žádosti povoleno (**EXTI_IMR**), pak se do bitu registru **EXTI_PR** zapisuje log.1 jako nevyřízená žádost o přerušení.

Bit v **EXTI_PR** značí: 0-nedošlo k požadavku, 1-nastal požadavek vybrané události.

Požadavek čekajícího přerušení lze **smazat zápisem log.1** do příslušného bitu registru **EXTI_PR**.

Existence naprogramované hrany je vzorkována systémovými hodinami a proto **vnější impuls musí být delší**, než perioda **Fclk**. Žádost o IRQ (interrupt request) povoleného přerušení pak vstupuje do **Nested Vectored Interrupt Controller NVIC**.

- Povolení přerušení se realizuje nastavením registrů **NVIC_ISER0 ÷ NVIC_ISER7** nebo zavoláním knihovní funkce **NVIC_EnableIRQ(EXTI15_10_IRQn);**

kde v závorce je identifikace daného zdroje přerušení.

- Zakázání IRQ se realizuje nastavením registrů **NVIC_ICER0 ÷ NVIC_ICER7**.
- Informace o stavu zpracování přerušení obsahují registry **NVIC_ISPR0 ÷ NVIC_ISPR7** (Interrupt Set-pending), **NVIC_ICPR0 ÷ NVIC_ICPR7** (Interrupt Clear-pending), **NVIC_IABR0 ÷ NVIC_IABR7** (Interrupt Active Bit).
- Registr **ICTR** (Interrupt Controller Type Register) určuje, kolik přerušení bude NVIC zpracovávat.

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 globální 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

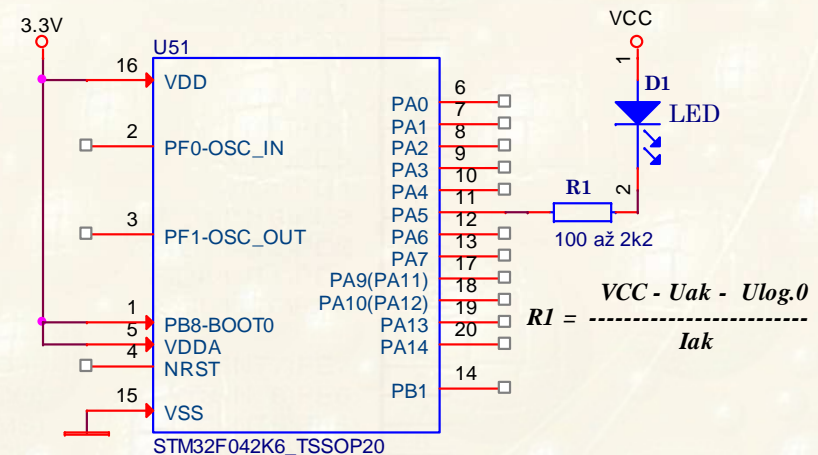
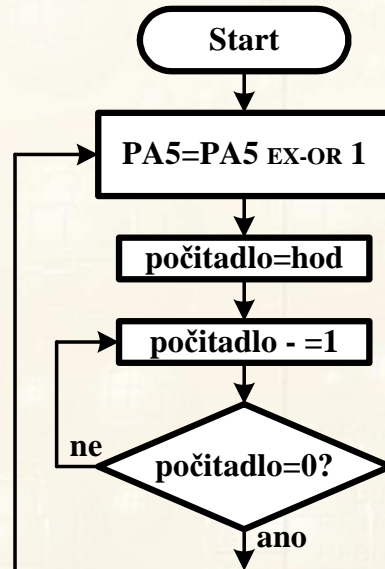
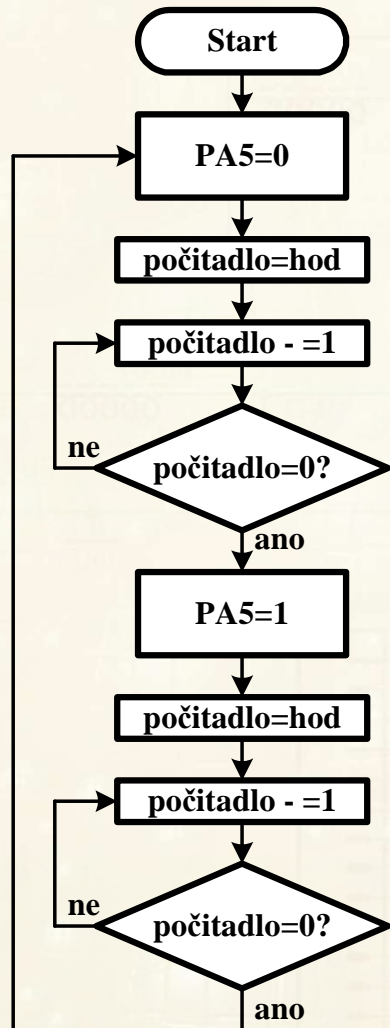
Přerušování a jeho stav je uložen v několika registrech:

- ❖ Enable and Clear Enable registers
- ❖ Set-Pending and Clear-Pending registers
- ❖ Priority level
- ❖ Active status
- ❖ Exception-masking registers
 - PRIMASK - LSB blokuje aktivaci přerušování s konfigurovatelnou prioritou
 - FAULTMASK - LSB blokuje aktivaci všech přerušování vyjma NMI
 - BASEPRI - určuje minimální prioritu zpracovávaných přerušování
- ❖ **STIR** - Software Trigger Interrupt Register obsahuje číslo spouštěného přerušování
- ❖ **AIRCR** – obsahuje 3 bity **PRIGROUP** rozdělující registr priority na bity priority a subpriority

PŘÍKLAD NA VYUŽITÍ PŘERUŠOVACÍHO SYSTÉMU

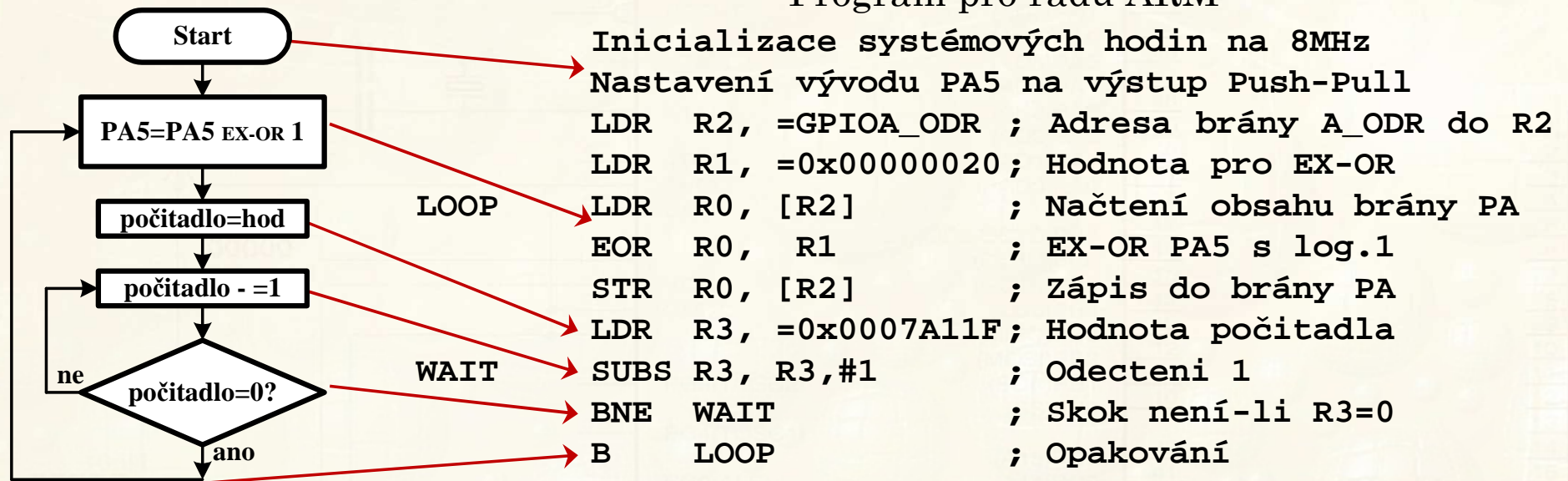
Příklad: Blikání diodou LED připojenou k vývodu PA5 procesoru z řady ARM v 0,25s intervalu.

- ❖ 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.



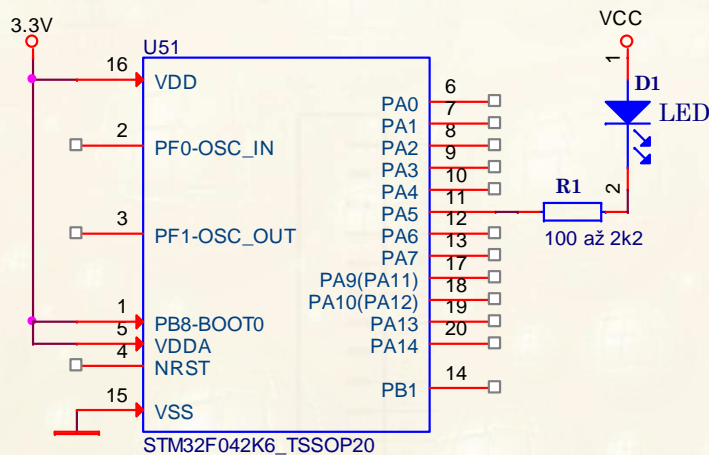
BLIKÁNÍ DIODOU LED – PROGRAMOVÉ ŘEŠENÍ

Program pro řadu ARM



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

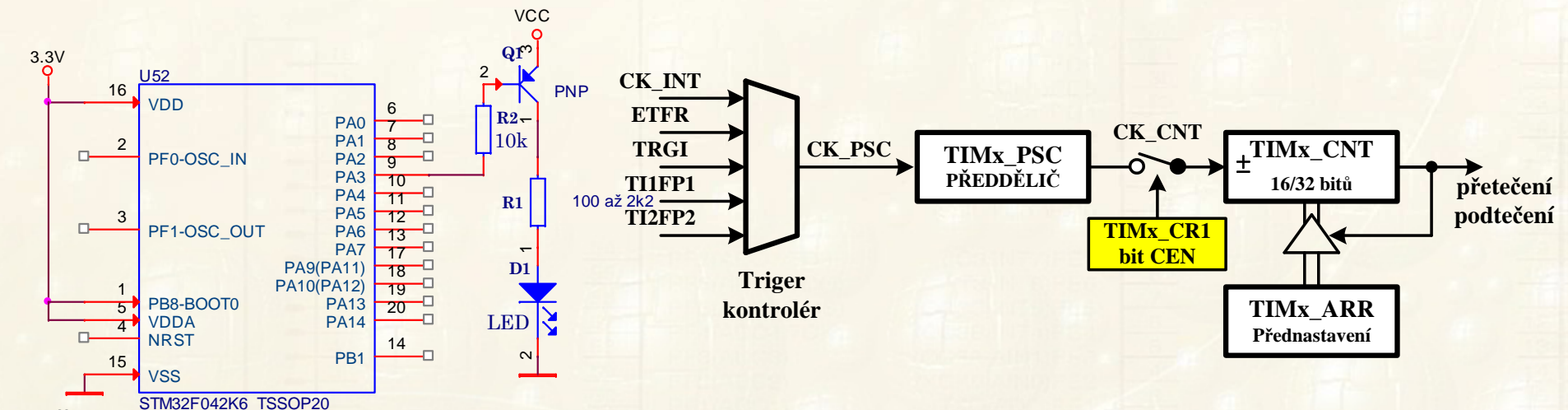
- ❖ Při hodinovém kmitočtu 8MHz \Rightarrow strojový cyklus = 125ns. Instrukce LDR, STR s pamětí trvají 2cykly, skok 3 cykly a ostatní jsou jedno cyklové. Celá smyčka WAIT 4 cykly.
- ❖ $Počítadlo+1=250000/(4*0,125)\approx 500000$
- ❖ Do počítadla (registru R3) uložíme hodnotu 0x7A11F.



BLIKÁNÍ DIODY LED S POUŽITÍM ČÍTAČE/ČASOVAČE

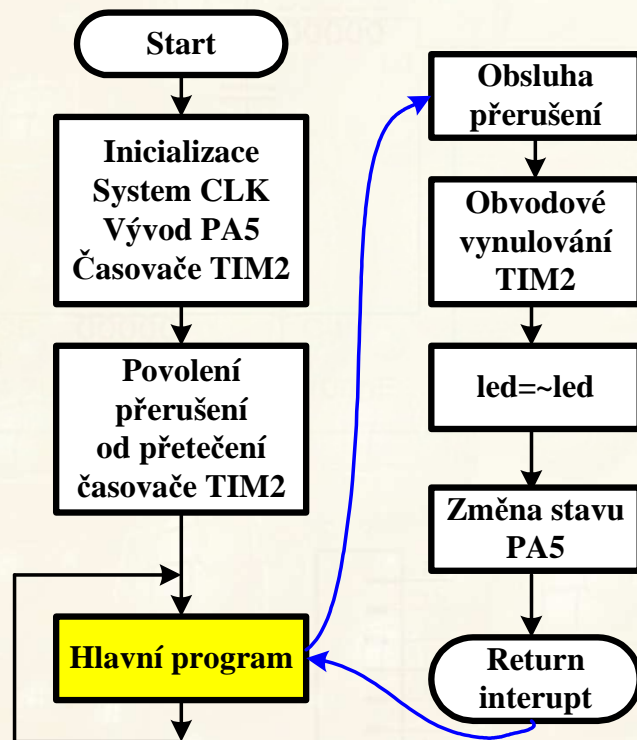
Programové zpoždění \Rightarrow **spotřebuje 100%** výpočetního výkonu procesoru. Jiná efektivnější a přesnější řešení.

- ❖ 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 hodnoty FFFFh) bude nastaven příznak žádosti o přerušování. Bude-li žádost povolena dojde k zavolání obsluhy **přerušování** – **přesnost?**



BLIKÁNÍ DIODY LED S POUŽITÍM ČÍTAČE/ČASOVAČE

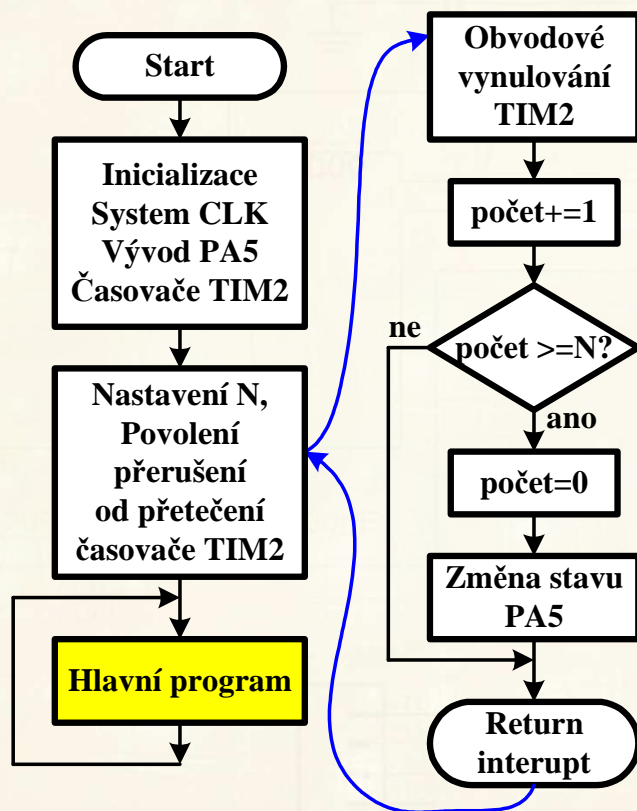
Pro zvolený časovač a čítaný kmitočet zvolíme hodnotu předděliče a nejvyšší hodnotu časovače tak, aby bylo dosaženo požadovaného časového intervalu. Dosáhne-li časovač mezní hodnoty, dojde k jeho vynulování a generování žádosti o přerušeni.



```
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()
    {
        // Nulování 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 předdělice čítače.
- Použít další časovač jako zdroj hodinového signálu pro používaný časovač.
- Použít externí zdroj hodinového signálu.
- Upravit obsluhu přerušení časovače zavedením počítadla **počet**, určujícího kolikrát byla přerušovací rutina zavolána. Generovaný interval pro změnu stavu pak vytvoříme jako celočíselný násobek doby mezi žádostmi o přerušení.