OBSLUHA PŘERUŠENÍ OBECNĚ

Obsluha přerušení je podprogram, který může být zavolán:

- Z podnětu interní periferie
- Vnější žádosti
- Z programu.

K přerušení, je-li povolené, může dojít **kdykoliv** (v kterékoliv části hlavního programu nebo podprogramu a stavu jednotlivých registrů a příznaků.

Má-li procesor více bank registrů, pak jedna slouží hlavním programu, další pro obsluhy přerušení. Uložení registrů a příznaků do zásobníku realizuje instrukce PUSH. Je-li uložen současný stav, teprve pak může být zahájena vlastní obsluha přerušení.

Na konci přerušení musí být obnoven původní stav registrů a příznaků před návratem do místa, před kterým bylo přerušení vyvoláno. Obnovení zajišťuje instrukce POP.

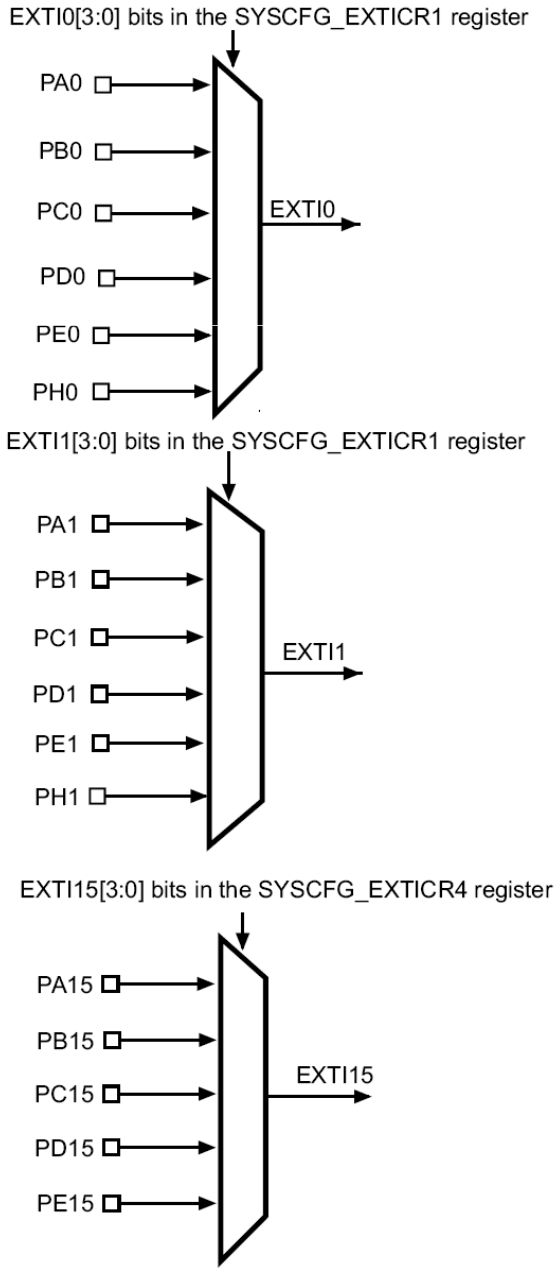
Přerušovací systém ARM je inspirován možnostmi 8048/51 s výrazně rozšířenými možnostmi.

Každý vývod GPIOA, GPIOB, atd. může být zdrojem vnějšího přerušení podle uvedeného schématu. Naprogramováním čtveřice bitů EXTIn[3:0] v registru SYSCFG_EXTICR1, určujeme zvolený bit brány takto:

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

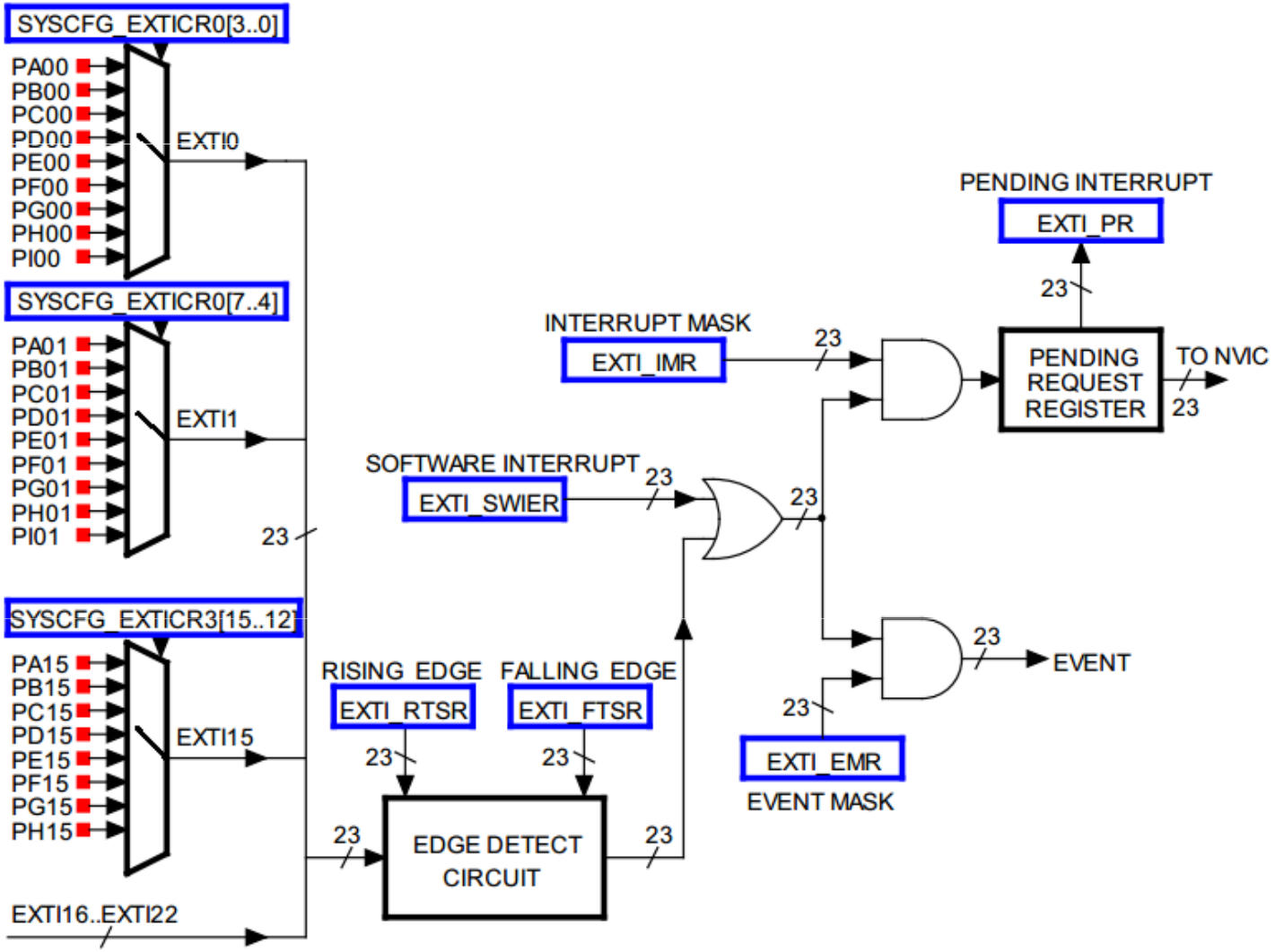
0000: vývod PAn; 0001: vývod PBn;
0010: vývod PCn; 0011: vývod PDn;
0100: vývod PEn; 0101: Rezervován
0110: Rezervován; 0111: vývod PHn;

Pro EXTI4 až EXTI15 slouží SYSCFG_EXTICR2, SYSCFG_EXTICR3 a SYSCFG_EXTICR4 se stejným způsobem přiřazení.



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.



The hardware of controller EXTI: blue boxes are registers, red dots are input pins

VNĚJŠÍ PŘERUŠENÍ NEBO UDÁLOST PROCESORŮ ARM M3 A M4

EXTIn (n=0 až 23) může být obvodové nebo softwarové přerušení nebo událost. Vývod může reagovat na náběžnou (**EXTI_RTSR**) nebo sestupnou (**EXTI_FTSR**) hranu podle naprogramování.

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

Bit TRn registru **EXTI_RTSR** nebo **EXTI_FTSR** povoluje reakci na danou hranu (log.1) nebo ji zakazuje (log.0).
K dosažení aktivace přerušení je třeba povolit přerušení v Interrupt mask registeru (**EXTI_IMR**) nastavením bitu MRx.

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

VNĚJŠÍ PŘERUŠENÍ NEBO UDÁLOST PROCESORŮ ARM M3 A M4

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

Všech 23 možných žádostí může být konfigurováno jako 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 impulz 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**.
- Informaci o stavu zpracování přerušení nesou 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

- Jestliže dojde k přerušení (výjimce) dojde k:
  - Uložení osmi registrů v následujícím pořadí PSR (program status), PC, LR, R12, R3, R2, R1, R0 do zásobníku.
  - Načtení adresy daného přerušení z tabulky vektorů přerušení
  - Aktualizuje se ukazatel zásobníku, link registr (LR) a čítač instrukcí (PC)
- Uložení registrů probíhá do zásobníku (PSP nebo MSP), který používá právě běžící program. Následně je SP aktualizován na (MSP), který bude v přerušení používán i pro další vnořená přerušení (nested interrupts). V registru PSR (program status) bude aktualizována část IPSR na nové číslo přerušení (výjimky). Do čítače instrukcí (PC) je uložena přečtená adresa z tabulky vektorů přerušení. Registr LR je aktualizován na speciální hodnotu nazvanou **EXC_RETURN**. Tato speciální hodnota 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 (reg)*, POP {PC}, LDR nebo LDM. Čtyři bity x určují, zda návrat je do souboru instrukcí ARM nebo Thumb včetně použitého ukazatele zásobníku SP (MSP nebo PSP).
- Výběr přerušení potřebných v předmětu je uveden v tabulce.
- Celá tabulka je v Cortex-M4_Reference manual_F401.pdf.

ADRESY PŘERUŠOVACÍHO SYSTÉMU ARM

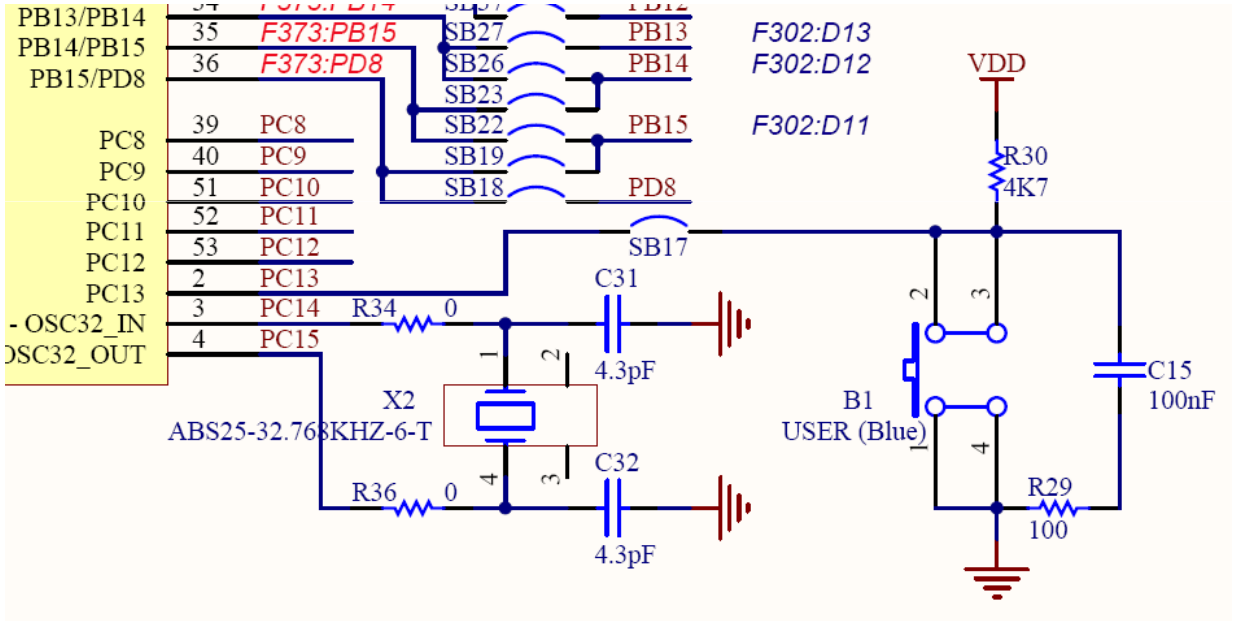
STM32F401 Výběr z tabulky vekrotů přerišení					
Pozice	**Priorita**	**Typ priority**	**Zkratka**	**Popis**	**Adresa obsluhy**
	-	-	-	Rezervováno (nastavení SP,)	0x0000 0000
	-3	neměnný	Reset	Nulování procesoru	0x0000 0004
	-2	neměnný	NMI	Nemaskované přerušení, Clock security System	0x0000 0008
	-1	neměnný	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

KONFIGURACE EXTERNÍHO PŘERUŠENÍ NA VÝVODU 13 BRÁNY GPIOC

```
// INICIALIZACE EXTERNÍHO PŘERUŠENÍ NA VÝVODU GPIOC_13 (MODRÉ TLAČÍTKO)

#include "stm32l1xx.h"                          // Device header
//      Mít nakonfigurován vývod GPIOC_13 na vstup, zvolené speed, bez upnutí

void TLAC_INT_Initialize (void)
{       RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;  // Povoleni hodinového signálu pro bránu GPIOC
                                              // V reg. AHBENR nastaven bit 2 (počítáno od 0)
        RCC->APB2ENR|=RCC_APB2ENR_SYSCFGEN;   // Povolení hodin.signálu v systémovém kontroléru
        EXTI->FTSR |= EXTI_FTSR_TR13;         // Reakce na sestupnou hranu
                                              // Zmáčknutí tlačítka = log.1->log.0
        SYSCFG->EXTICR[3] = SYSCFG_EXTICR4_EXTI13_PC; // Přerušení přichází z brány GPIOC
        EXTI->IMR |= EXTI_IMR_MR13;           // Povoleni přerušení od EXTI line 13
        NVIC_EnableIRQ(EXTI15_10_IRQn);       // Povolení externího přerušení v kontroléru NVIC
}
```



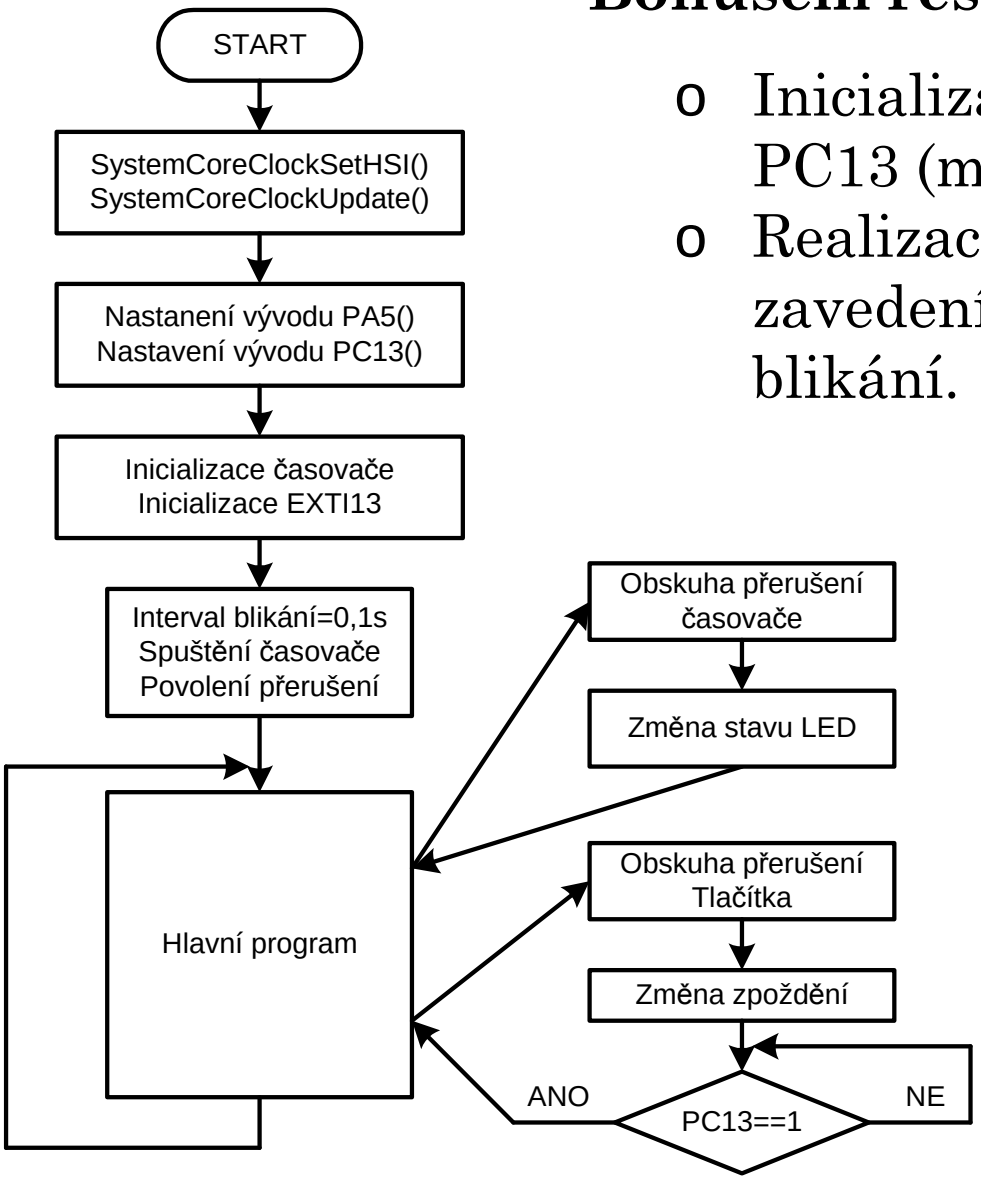
Typický začátek programu v jazyce C

```
#include <stdio.h>
#include "stm32F4xx.h"                    // Device header
#include "LED_NUCLEO_F401.c"
#include "Buttons_NUCLEO_F401.c"
#define setbit(reg, bit)      ((reg) |= (1U << (bit)))
#define clearbit(reg, bit)    ((reg) &= (~(1U << (bit))))
#define togglebit(reg, bit)   ((reg) ^= (1U << (bit)))
//   Definice konstant, připojení, globálních proměnných, atd.
 const uint32_t doba_zpozdeni  =  0x22800; unsigned int btns = 1;

main() int main (void)
{       SystemCoreClockSetHSI();
        SystemCoreClockUpdate();  // Get Core Clock Frequency
        LED_Initialize();
        Buttons_Initialize();
        TLAC_INT_Initialize();
        TIM2_Inicializace();
Hlavni: while(1)                  // Smyčka hlavního programu (nekonečná)
        {    if (btns==1) LED_On (num); else LED_Off(num); }
}
//      PODPROGRAMY PŘERUSENÍ
void EXTI15_10_IRQHandler()
{   if ((EXTI->PR & EXTI_PR_PR13)!=0) EXTI->PR |= EXTI_PR_PR13;  // Byla-li sestupná
                        // hrana na GPIOC_13, pak nuluj příznak přerušení (pending).
    if (btns == 1)  btns = 0;  else  btns = 1;}               // Změna stavu blikání

void TIM2_IRQHandler()
{    TIM2->SR &= ~TIM_SR_UIF;     // Nulovaní příznaku události od časovače TIM2
     if (btns == 1)  btns = 0;    // Změna stavu diody LED - v programu main
     else  btns = 1;}
```

KONCEPCE BONUSU ŘEŠENÍ ÚLOHY 1



Bonusem řešení první úlohy je doplnění o:

- Inicializaci vnějšího přerušení od vývodu PC13 (modré tlačítko)
- Realizaci přerušovací rutiny EXTI13 a zavedení proměnné identifikující periodu blikání.

Postup řešení

- Vyjdeme z funkční úlohy 1
- Budeme inicializovat vnější přerušení od vývodu PC13, které se bude skládat z
  - Přivedení hodinového signálu k bráně C
  - Přivedení hodinového signálu do systémovém kontroléru
  - Určení hrany pro EXTI13, na kterou bude přerušení realizováno (náběžná/sestupná)
  - Určení brány, z které bude EXTI13 realizováno
  - Povolení masky přerušení EXTI13
  - Povolení externího přerušení v kontroléru NVIC
  - **Celá inicializace povolení přerušení od modrého tlačítka je na str.10**

POKRAČOVÁNÍ ŘEŠENÍ ÚLOHY 1

- Vytvoření přerušovací rutiny pro EXTI13
  - V přerušovací rutině nutno zjistit, z kterého zdroje přerušení přichází (EXTI10 až EXTI15) v našem případě lze vynechat.
  - Je-li na jednu adresu je směrováno více zdrojů, příznaky přerušení se nemohou automaticky mazat ⇒ musíme smazat příznak od EXTI13. (U ARM je musíme mazat vždy).
  - Po smazání příznaku realizujeme obsluhu tlačítka. Chceme aby po zmáčknutí došlo ke změně blikání ⇒ Změna indikátoru, proměnné nebo konstanty zpoždění.

- **Celá přerušovací rutina je v dolní části na str.11 spolu s kostrou přerušovací rutiny pro druhý bonus úlohy 1.**

ČÍTAČE NA PROCESORECH STM32 - OBECNĚ

Na procesorech STM32 máme čítače TIM2 až TIM5.

- 16-ti bitové TIM3 a TIM4
- 32-ti bitové TIM2 a TIM5

Čítače svůj obsah inkrementují, dekrementují, reverzují s obvodovým přednastavením. Hodinový signál do čítače přichází z předděliče s dělícím poměrem 1 až 65536.

Čítač může ovládat až 4 nezávislé kanály:

- záchytného nebo komparačního systému
- PWM nebo výstupu jednoho impulzu.

Hodinový signál do časovače přivádí synchronizační kontrolér z

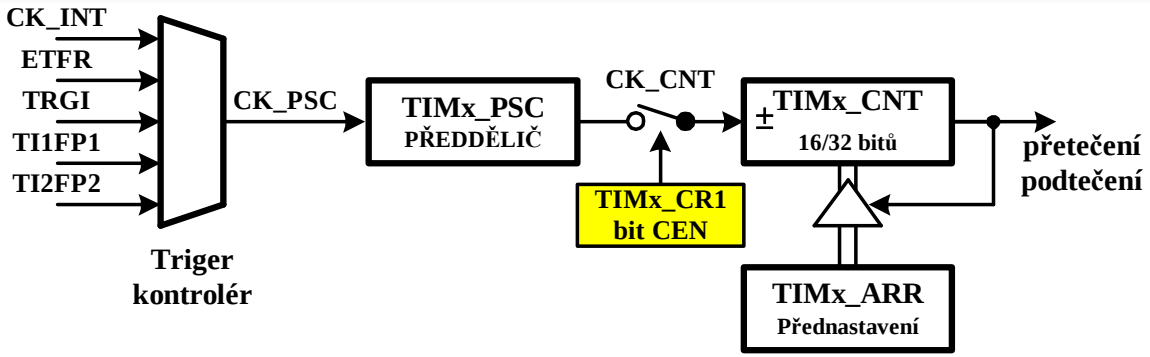
- Interního zdroje signálu
- Externí zdroj signálu
- Z výstupu jiného časovače

ČÍTAČE NA PROCESORECH STM32 - OBECNĚ

Generování přerušení nebo DMA přenos je odvozeno od:

- Přetečení nebo podtečení časovače
- Inicializace čítače (programem nebo interním/externím impulzem)
- Událostí (start, stop, inicializace nebo čítáním interních/externích impulzů)
- Záchytného systému
- Komparačního systému
- Inkrementálního enkodéru a halovou sondou indikující pozici
- Spouštěcího signálu externích hodin nebo obsluhou cycle-by-cycle.

ČÍTAČE/ČASOVAČE - OBECNĚ



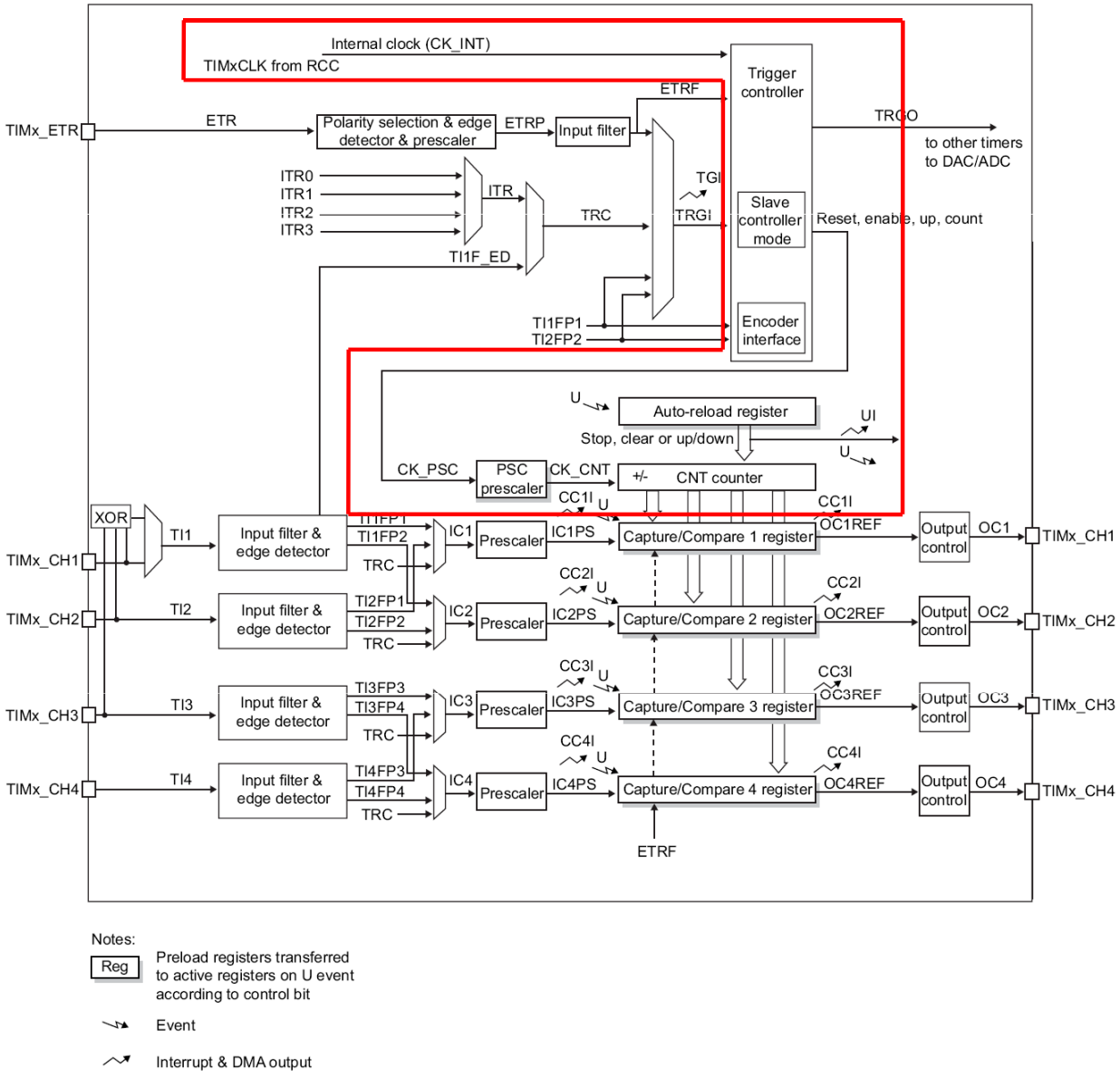
Potřebné doby zpoždění může být dosaženo tzv. zkrácením cyklu, kdy po přetečení, podtečení nebo dosažení maximální hodnoty je stav čítače **nastaven** na skoro libovolnou hodnotu. **Programová změna TIMx_CNT** se může uskutečnit v obsluze **přerušení**. Takto generovaný interval se bude z dlouhodobého pohledu trochu zpožďovat. **Proč???**

Pro generované intervaly (se stabilitou hodinového signálu) je nutno použít čítač **s obvodovým přednastavením**. Po dosažení čítače TIMx_CNT hodnoty v registru **TIMx_ARR** dojde k jeho vynulování a přechodu na hodnotu 0. Při dekrementaci po dosažení nuly dojde k přednastavení na hodnotu **TIMx_ARR**. Přetečení může představovat **událost** nebo **žádost o přerušení**.

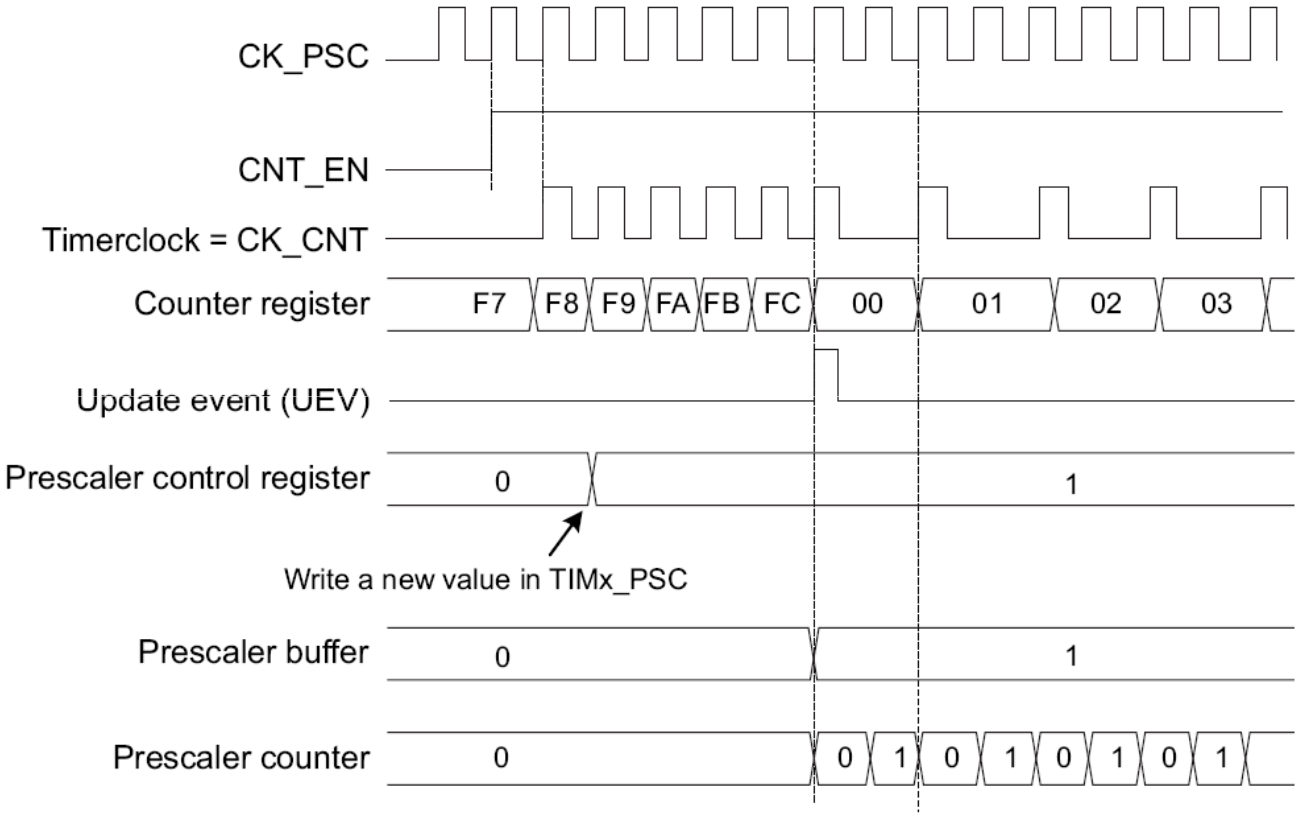
Problém se změnou TIMx_ARR.

ČÍTAČE TIM2 AŽ TIM5 NA PROCESORECH STM32 - OBECNĚ

Ze zapojení čítačů TIM2 až TIM5 je pro úlohy je důležitá červeně označená část. Oblast popisuje výběr hodinového signálu **CK_INT** pro čítač, poslé-ze přivedeného do před-děliče jako signál **CK_PSC**. Z PSC pak vychází signál pro vlastní čítač **CK_CNT**. Pro dosažení periody bude čítač obvodově před-nastavován na hodnotu uloženou v registru **TIMx_ARR** nebo nulo-ván po dosažení jeho hodnoty. Od této události čítače bude generována žádost o přerušení označená na obrázku **UI**.



Předdělič - Dělí hodinový signál pro čítač poměrem 1 až 65536, tvořený 16-bitovým čítačem **TIMx_PSC**. Díky dvojnásobnému ukládání může být jeho hodnota změněna i při jeho činnosti. Nová hodnota poměru z registru TIMx_PSC je přepsána do výkonného registru až z novou aktivační událostí. Na obrázku je příklad chování čítače se změnou dělícího poměru za běhu.



INICIALIZACE ČASOVAČE TIM2 A OBSLUHA JEHO PŘERUŠENÍ

```
void TIM2_Inicializace(void)
{
//   TIM2 je připojen ke sběrnici APB1, která běží na 24MHz
//   frekvence před TIM2 je zdvojnásobena PLL
     RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;     // Povolení hodinového signálu pro TIM2
//   nebo  setbit(RCC->APB1ENR, 0);
     TIM2->PSC = XXXX;                       // Nastavení dělícího poměru XXXX předděliče
     TIM2->ARR = YYYY;                       // Nastavení hodnoty YYYY přednastavení ARR
     TIM2->DIER |= TIM_DIER_UIE;             // Povolení přerušení od TIM2
     NVIC_EnableIRQ(TIM2_IRQn);              // Povolení přerušení od TIM2 v kontroléru NVIC
     TIM2->CR1 |= TIM_CR1_CEN;               // Spuštění časovače TIM2
                                             // nebo setbit(TIM2->CR1, 0);
     TIM2->SR &= ~TIM_SR_UIF;                // Nulovaní příznaku události časovače TIM2
                                             // nebo clearbit(TIM2->SR, 0);
}

void TIM2_IRQHandler()                       // Obslužná rutina přerušení TIM2
{    TIM2->SR &= ~TIM_SR_UIF;                // Nulovaní příznaku události od časovače TIM2
     ...
     ...                                     // Program realizující obsluhu každého přetečení
     ...                                     // časovače TIM2
}
```

OVĚŘENÍ EXTERNÍHO PŘERUŠENÍ A PŘERUŠENÍ OD ČASOVAČE

Cíl: *Doplňte program Blinky_Start o inicializaci externího přerušení od modrého tlačítka GPIOC_13 a inicializaci přerušení od časovače TIM2.*

1. *Inicializace externího přerušení GPIOC_13 je na str.10*
2. *Inicializujte přerušení od časovače TIM2 str.20*
3. *Doplňte obsluhy přerušení od tlačítka a časovače str.11 a 20.*
4. *Doplňte inicializace přerušení tlačítka a časovače za inicializace tlačítka a LED str.11.*
5. *Umístěte po překladu a přechodu do debuggru **break point** do přerušovacích rutin a ověřte, že přerušení fungují.*
6. *K dokončení první úlohy na ARM pak zbývá dopsat obsluhy a hlavni program v souladu se zadáním.*

OVĚŘENÍ EXTERNÍHO PŘERUŠENÍ A PŘERUŠENÍ OD ČASOVAČE

Cíl: *Doplňte program Blinky_Start o inicializaci externího přerušení od modrého tlačítka GPIOC_13 a inicializaci přerušení od časovače TIM2.*

1. *Inicializace externího přerušení GPIOC_13 je na str.10*
2. *Inicializujte přerušení od časovače TIM2 str.20*
3. *Doplňte obsluhy přerušení od tlačítka a časovače str.11 a 20.*
4. *Doplňte inicializace přerušení tlačítka a časovače za inicializace tlačítka a LED str.11.*
5. *Umístěte po překladu a přechodu do debuggru **break point** do přerušovacích rutin a ověřte, že přerušení fungují.*
6. *K dokončení první úlohy na ARM pak zbývá dopsat obsluhy a hlavni program v souladu se zadáním.*