ARM Cortex magú mikrovezérlők

NVIC

Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

ARM7, ARM9 megszakítás kezelés

- ARM7, ARM9 két interrupt vonal
  - IRQ: Normál priorítású IT
  - FIQ: Fast IT saját regiszter blokkal
  - A vektoros megszakításkezelés gyártó specifikus
  - Nem volt determinisztikus az interrupt kiszolgálás: attól függött a megszakítás kiszolgálása, hogy éppen milyen utasítás hajtódott végre.
  - Az ARM7, ARM9 hardware-esen nem támogatta az ún. Nested IT-ket. (IT-t megszakító IT)
- A Cortex M sorozat megszakítás kezelője a fenti korlátokra próbál megoldást adni.

Cortex M3 NVIC

- Nested Vector Interrupt Controller
  - Gyártó független standard tartozék, ebből következően gyártó független interrupt struktúra.
    - Könnyű portolhatóság
  - A Thumb2 utasításkészlet több órajelig tartó utasításai megszakíthatóak, így az IT kezelés determinisztikus.
  - Nested interuptokat támogatja
  - Bár az NVIC processzor független, az erőforrás használat minimalizálása miatt a processzor tervezők megszabhatják NVIC bemenő vonalainak számát.
    - Az NVIC képes: 1 nem maszkolható +240 külső periféria + 15 belső Cortex-es IT vonal forrást kezelni
      - Az STM32f107 43-at
      - Az LPC1768 35-öt használ

Az NVIC kezelése

- Meg kell adni az ugrótáblát és a prioritásokat.
- Az ugrótábla a címtartomány alján a 0x00000004-ről indul.
  - A 0x00000000-án a kezdő stack pointer van
    - Minél hamarabb lehessen C-t használni.

32 bites CRC, FCP Frame format check pattern

- Az első 15 megszakítás a Cortex Core-hoz tartozik
- Ezek után jönnek a gyártó specifikus periféria megszakítások

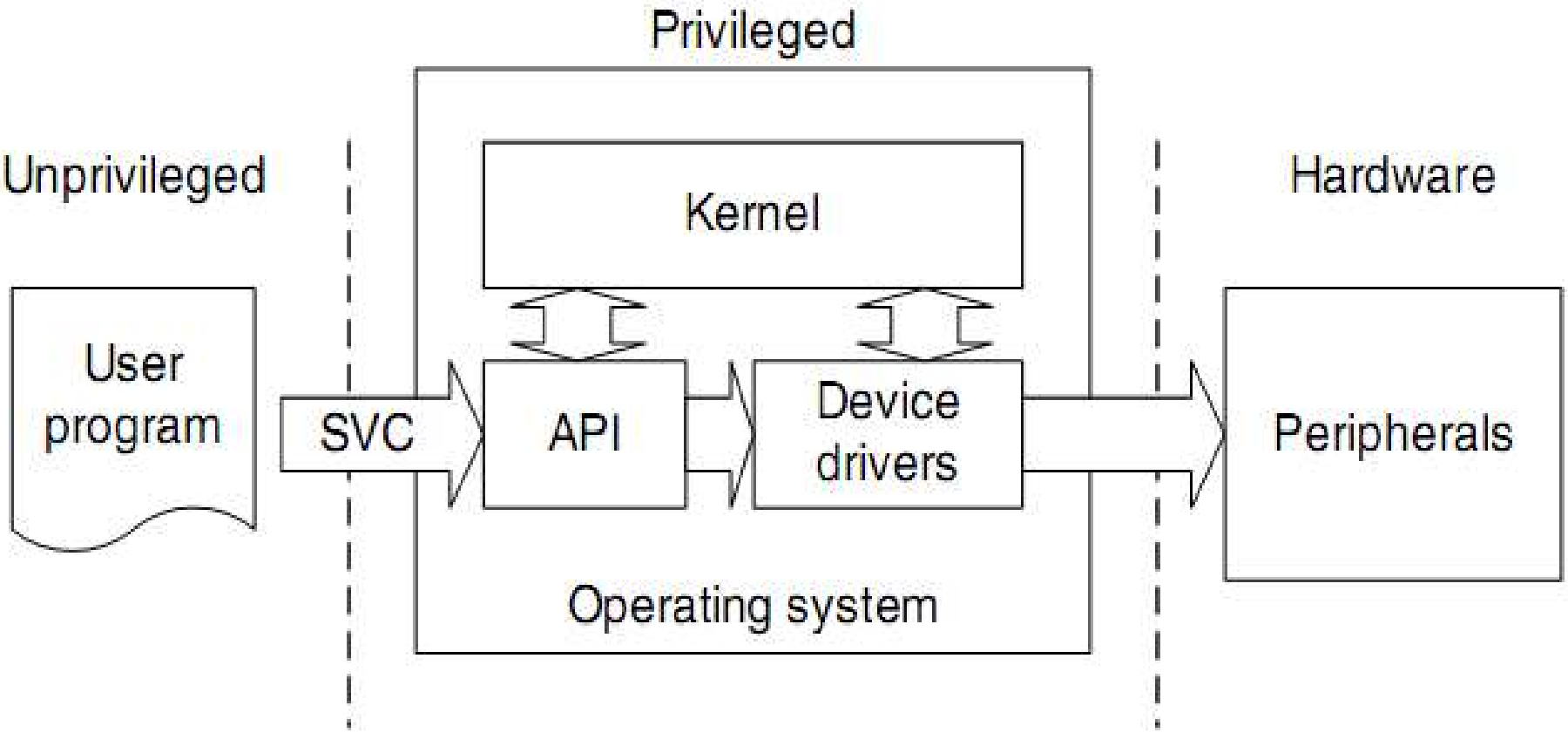
Az NVIC ugrótábla

- Az ugrótábla a címtartomány alján a 0x00000004-ről indul.
  - A 0x00000000-án a kezdő stack pointer van, hogy minél hamarabb lehessen C-t használni.

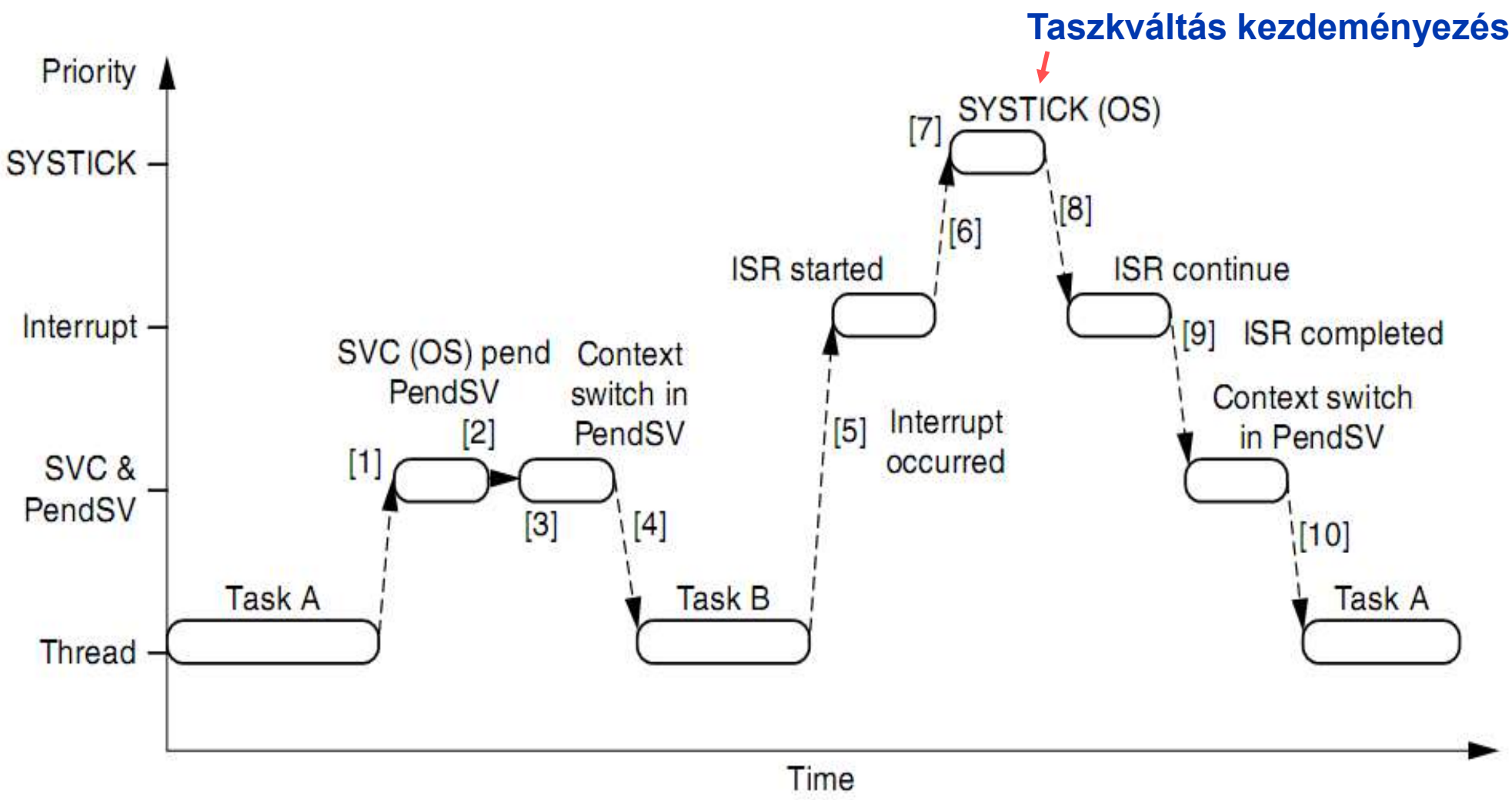
No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
.....	.....	.....	settable	.....
256	Interrupt#240	247	settable	External Interrupt #240

Gyártó specifikus

A SVC használata



A PendSV kezelése

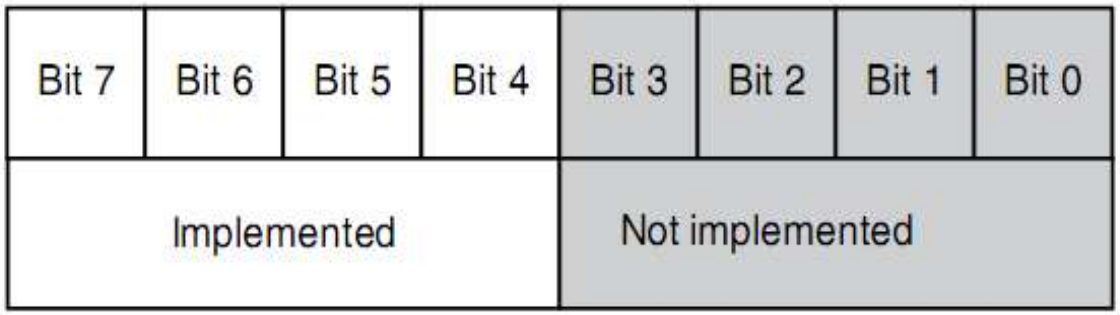


Alap regiszterek

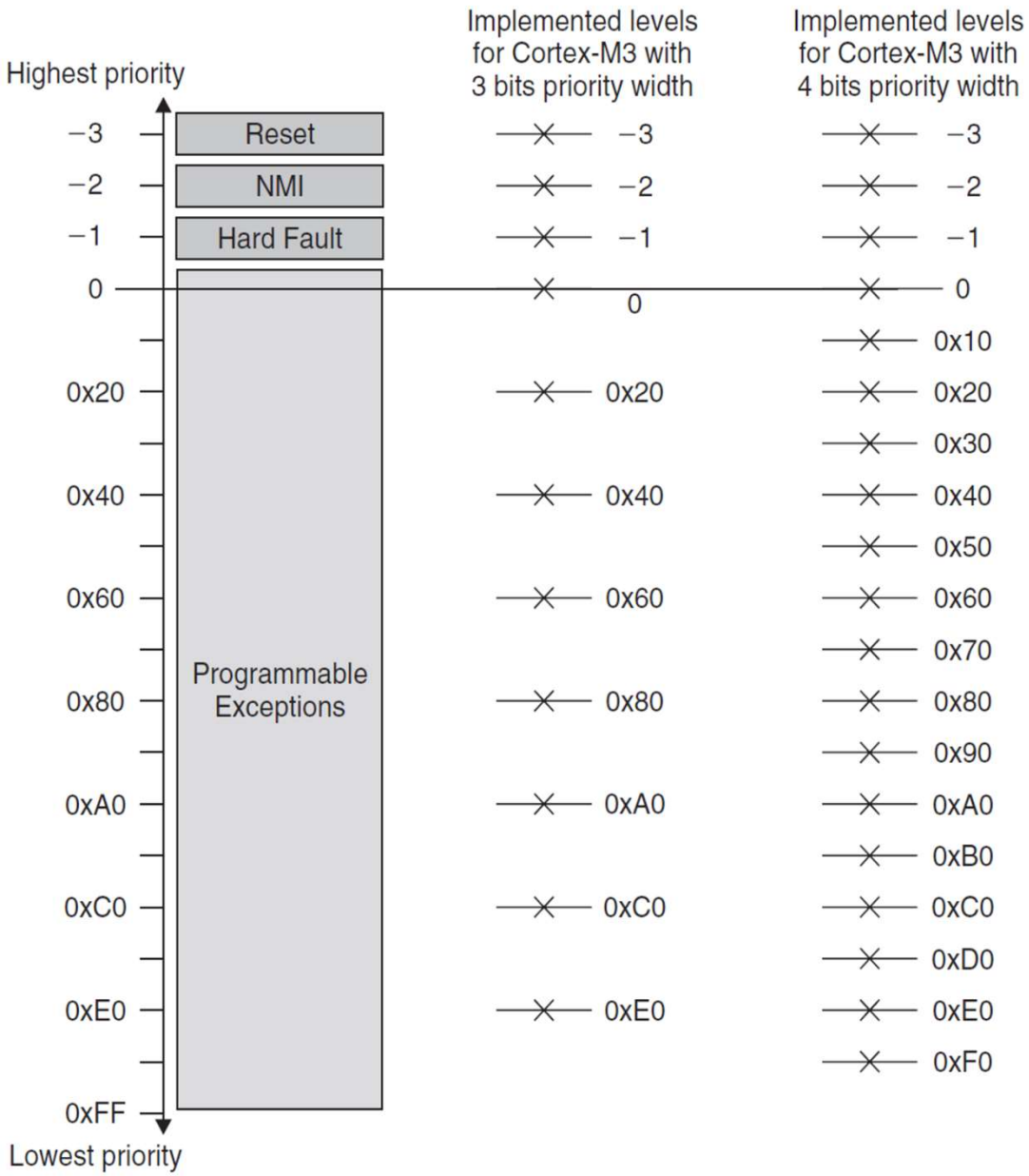
- **Interrupt enable** és **Clear enable** regiszterek
  - SETENA0-n/CLRENA0-n
    - 32bites külön tiltó és engedélyező regiszter
- **Interrupt set pending** és **Clear pending**
  - SETPEND0-n/CLRPEND0-n
    - 32bites regiszterek, amelyekből a megszakított, várakozó interruptokat lehet kiolvasni, törölni.

Prioritás kezelés

- Az alap belső kivételeknek fix prioritása van
- A többi külső megszakításhoz (a mag szempontjából) prioritás regiszter
  - max. 8 bites, min 3 bites prioritás regiszter
  - Korlátozni szokták a szinteket az egyszerűbb hardware kialakítás miatt
  - A MSB bitektől kezdődik az implementálás (könnyebb portolhatóság)
  - Az STM32F107, LPC1768, STM32F429 esetében 4 bites prioritás regi

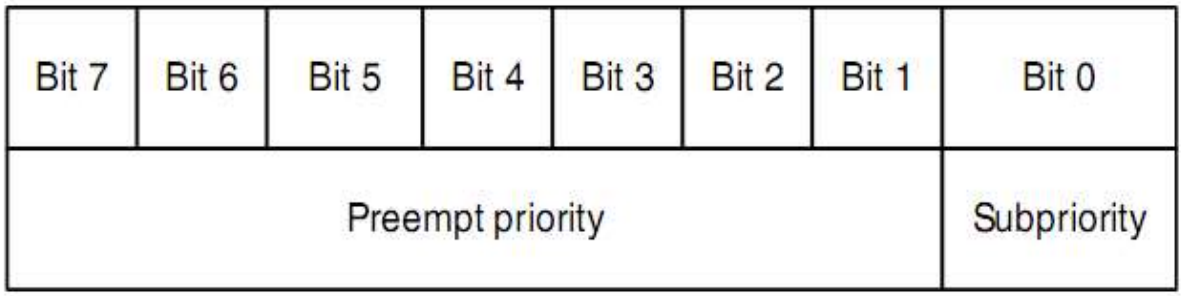


Prioritás kezelés *folytatás*



Preempt priority és Subpriority

- A 8 bit, de csak 127 preemptciós szint létezik
- Subprioritás
  - Azonos preemptciós szintű prioritások szerint az alacsonyabb subprioritású fut le először
  - PRIGROUP regiszter



Preempt priority és Subpriority

- A 8 bit, de csak 127 preemptciós szint létezik
- Subprioritás
  - Azonos preemptciós szintű prioritások szerint az alacsonyabb subprioritású fut le először
  - PRIGROUP regiszter
  - Az STM32F107, LPC1768, STM32F429 esetében 4 bites prioritás regiszter

PRIGROUP (3 Bits)	Binary Point (group.sub)		Preempting Priority (Group Priority)		Sub-Priority	
			Bits	Levels	Bits	Levels
011	4.0	gggg	4	16	0	0
100	3.1	gggs	3	8	1	2
101	2.2	ggss	2	4	2	4
110	1.3	gsss	1	2	3	8
111	0.4	ssss	0	0	4	16

További NVIC regiszterek, opciók

- Megszakítás maszkot
  - PRIMASK mindet kivéve a hibákat
  - FAULTMASK a hibákat is maszkolja -1 ig
  - BASEPRI egy bizonyos prioritás alatt maszkol
- Vector Table Offset Regiszter
  - Áthelyezhető az IT táblázat szinte tetszőleges helyre
  - Boot-olás, Boot-loader segítő lépés

Table 7.7 Vector Table Offset Register (Address 0xE000ED08)

Bits	Name	Type	Reset Value	Description
29	TBLBASE	R/W	0	Table base in Code (0) or RAM (1)
28:7	TBLOFF	R/W	0	Table offset value from Code region or RAM region

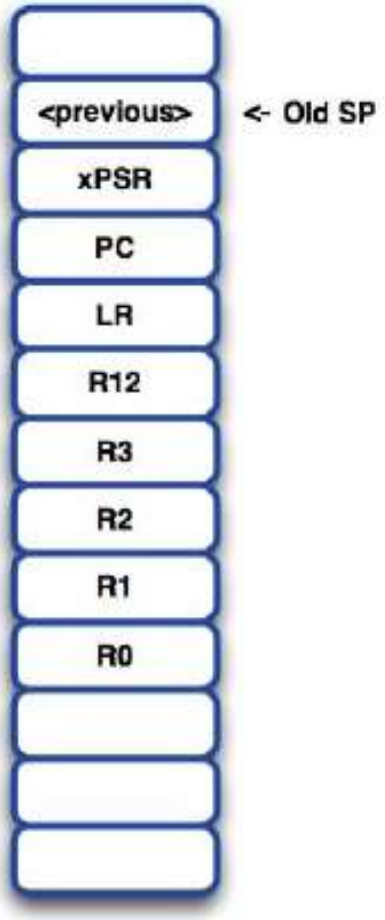
Az NVIC működése, az IT hatása I.

- Az IT hatására a Cortex M3 IT kezelő állapotba megy és lementi a regiszter készletet a stack-re.
  - Ez mikrokódban történik nincs szükség hozzá programozói beavatkozásra. 8 regiszter mentődik el: a
    - Program Status Register
    - Program Counter
    - Link Register
    - R0 – R3 regiszterek (ezek tartalmazzák a függvények paraméter hívásait) és az R12 regiszter (compiler segéd adatregiszter)
    - Process Stack átkapcsolódik a Main Stackre ha szükséges
  - Eközben az IT kiszolgáló címét elkezdi felhozni az utasítás buszon.

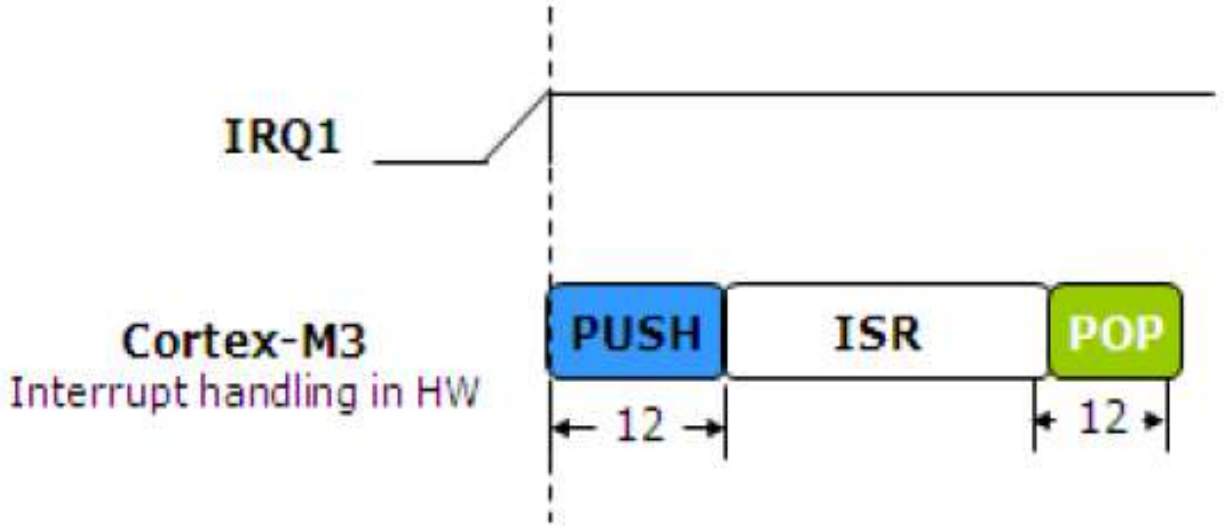
Az NVIC működése, az IT hatása II.

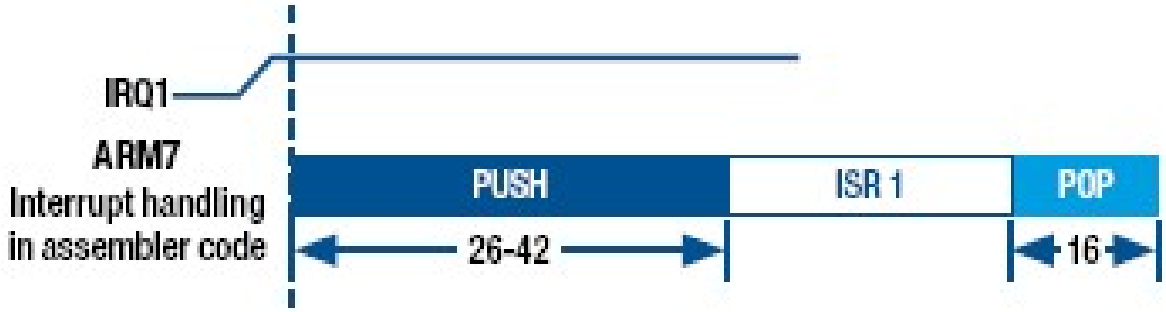
- Az IT kezdőcímének felhozása után aktualizálás
  - Stack Pointer
  - Link Register
  - Program Counter
  - Interupt Program Status Register: IPSR
- Az IT kiváltása után 12 órajellel elkezdődik az IT kiszolgálás.
- Az IT után a visszatérés ugyanúgy 12 órajel ciklus.
  - Nincs speciális visszatérő utasítás

Az NVIC működése összefoglalás



The NVIC will respond to an interrupt with a latency of just six cycles. This includes a microcoded routine to automatically push a set of registers onto the stack.



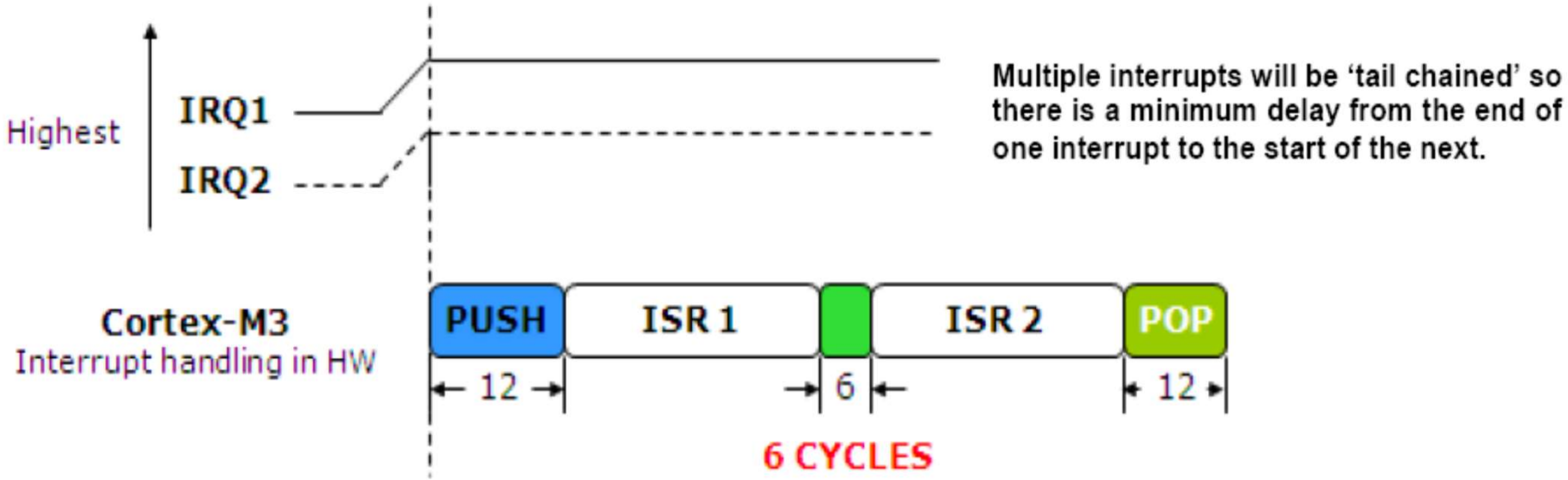


Működés több egyidejű megszakítás esetén I.

- Real-time alkalmazásokban fontos, hogy a megszakítások prioritását is szabályozni tudjuk.
  - Hagyományosan a hard-realtime rendszerek nem, vagy csak nagyon kevés IT-t használnak, pont azért mert a sok IT egymást is késlelteti.
- Preemptív IT kezelés
  - A Cortex M3 biztosítja, hogy egy magasabb prioritású IT meg tudjon szakítani egy alacsonyabb prioritásút.
  - Az alacsonyabb prioritású IT regiszterei, ugyanúgy mint a főprogram regiszterei, lementődnek. A magas prioritású megszakítás 12 órajel ciklus alatt el kezd végrehajtódni.

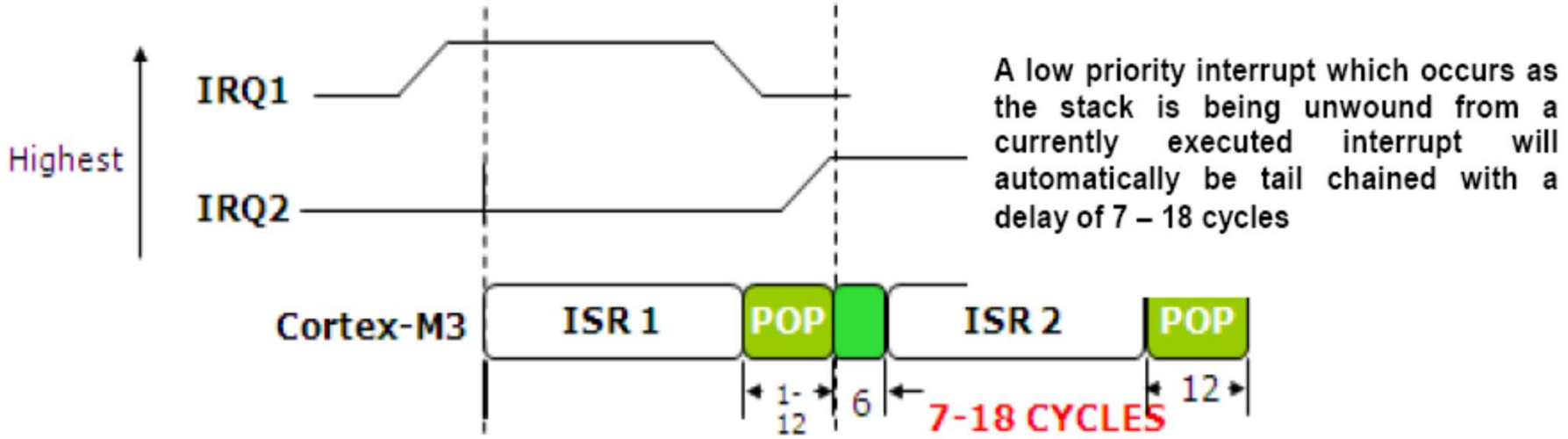
Működés több egyidejű megszakítás esetén II.

- Tail chaining: A megszakítások láncban történő végrehajtása több egyidejű IT esetén.
  - A második IT végrehajtása az ARM7 esetében jóval hosszabb lett volna (42 óraciklus), mert ott egy POP(16) és PUSH(26) is lejátszódott volna.



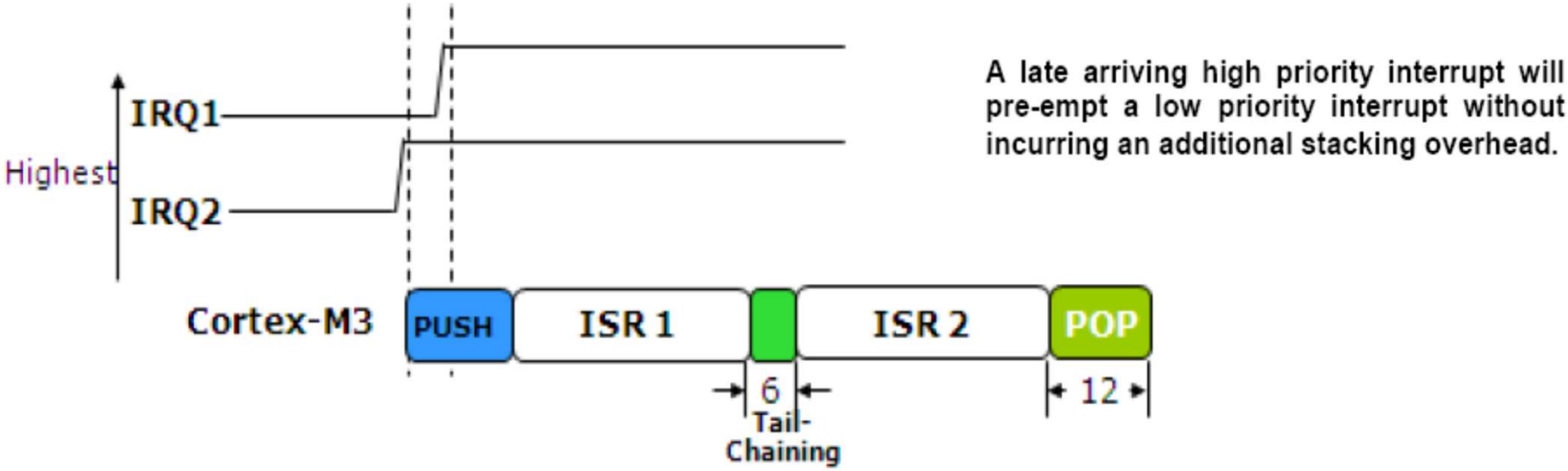
Működés több egyidejű megszakítás esetén III.

- Megszakításból való visszatérés megszakítása
  - A POP művelet itt megszakadhat.



Működés több egyidejű megszakítás esetén IV.

- Elkésett nagyobb prioritású IT nem okoz gondot.
  - A megszakítás kiváltása után 6 órajellel, ami az IT kiszolgáló címének felhozásához kell a nagyobb prioritású IT futáskész.



NVIC programozás:

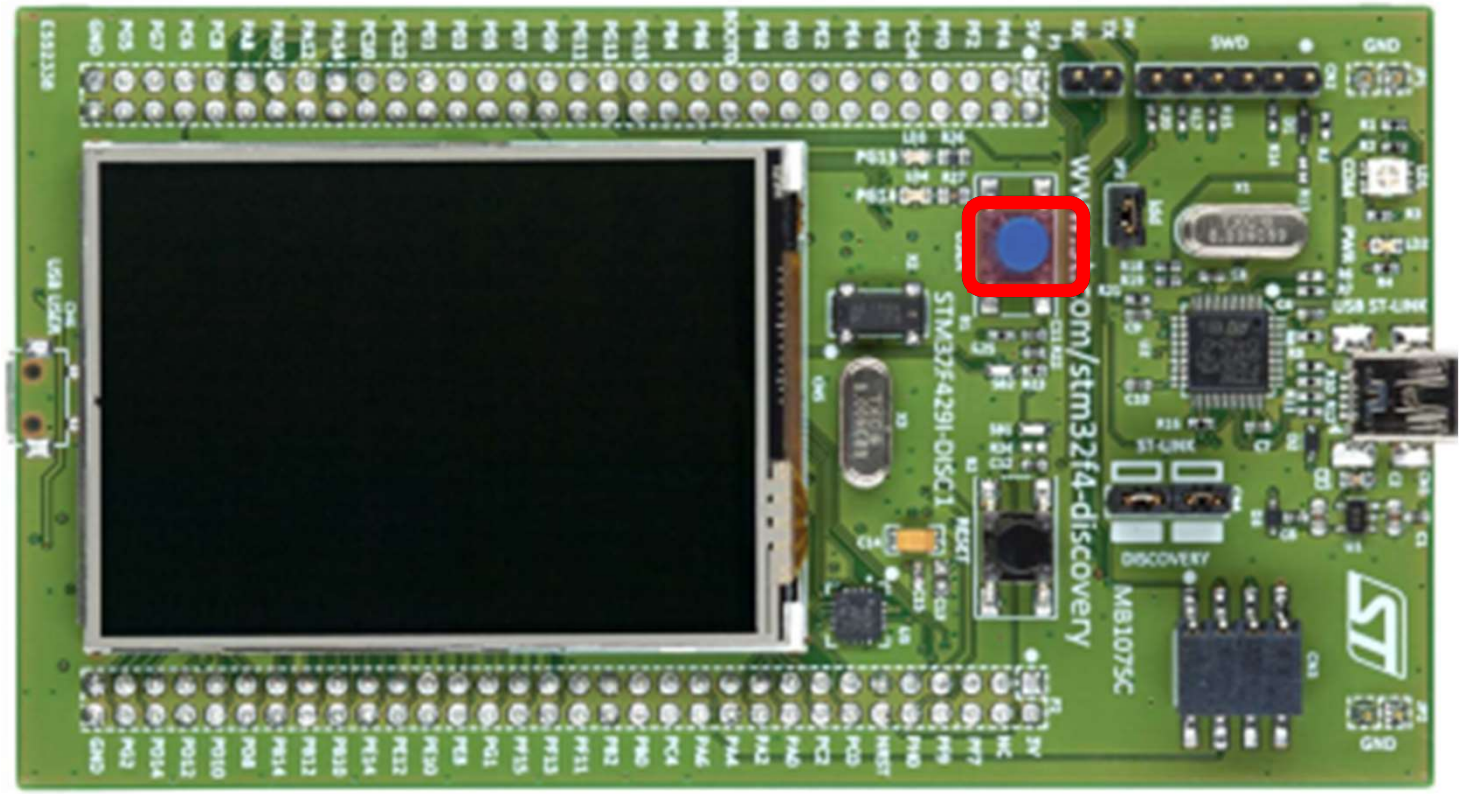
ST nyomógomb IT

Megvalósítás

- 1. Lépés: GPIO láb felkonfigurálása
  - GPIO láb megkeresés
  - GPIO láb konfigurálás
  - EXTI hozzárendelés konfigurálás
  - EXTI konfigurálás
- 2. Lépés: IT kérés
  - Interrupt kontroller felprogramozás
  - IRQ handler

Nyomógomb az STM32F429 Disc1-en

- 6.6 Fejezet: PA0 láb a User button



Lábkonfigurálás

- PA0 input-nak konfigurálás
  - GPIO Init

```
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
GPIO_InitStruct.Pin = LL_GPIO_PIN_0;
GPIO_InitStruct.Alternate = LL_GPIO_AF_0;
GPIO_InitStruct.Pull = LL_GPIO_PULL_DOWN;
LL_GPIO_Init  (GPIOA,&GPIO_InitStruct);
```

Külső interrupt hozzárendelés konfigurálás

- Minden láb lehet külső interrupt, de azért korlátozások léteznek
  - Egy EXTI-hez egy láb tartozik
  - Az EXTI0-hoz, vagy a PA0, vagy a PB0, PC0 … van hozzárendelve
- Konfigurálás a System Control blokkban

```
LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTA ,LL_SYSCFG_EXTI_LINE0 );
```

Külső interrupt konfigurálás

- Külön EXTI periféria blokk
  - Lefutó, felfutó él
  - Engedélyezés

```
LL_EXTI_InitTypeDef EXTI_InitStruct;
EXTI_InitStruct.LineCommand = ENABLE;
EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_0;
EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
LL_EXTI_Init  (&EXTI_InitStruct);
LL_EXTI_EnableIT_0_31 (LL_EXTI_LINE_0);
```

Interrupt konfigurálás

- NVIC interrupt szám megadása: device.h
- Priorítás beállítása
- Engedélyezés

```
NVIC_EnableIRQ(EXTI0_IRQn);
```

Az interrupt kezelő függvény elkészítése

- Startup_device.h tartalmazza az alap definíciót. Ezt kell „felülírnunk”, ami azt jelenti, hogy egy ilyen nevű függvényt kell létrehoznunk, és mivel az alap weak-kén van definiálva, ezért az újonnan létrehozott függvény fog fordításnál érvényre jutni.

```
.weak       EXTI0_IRQHandler
.thumb_set EXTI0_IRQHandler,Default_Handler
```

Az interrupt kezelő függvény elkészítése

- Startup_device.h tartalmazza az alap definíciót. Ezt kell „felülírnunk”, ami azt jelenti, hogy egy ilyen nevű függvényt kell létrehoznunk, és mivel az alap weak-kén van definiálva, ezért az újonnan létrehozott függvény fog fordításnál érvényre jutni.

```
void EXTI0_IRQHandler(void)
{
    if(LL_EXTI_IsActiveFlag_0_31 (LL_EXTI_LINE_0)) // Not needed
    {
        printf(TERM_BRED "Push Button IT \r\n" TERM_DEFAULT);
        LL_EXTI_ClearFlag_0_31  (LL_EXTI_LINE_0);
    }
}
```

NVIC programozás:

System Timer

Megvalósítás

- Egyszerű, de nincs a System Timer-hez firmware library támogatás
- 1. Lépés: System Timer programozás
  - System Timer funkció feltérképezés
  - Load regiszter beállítás
  - Config regiszter beállítás
- 2. Lépés: IT kérés
  - Interrupt kontroller felprogramozás
  - IRQ handler

System Timer regiszterek

- Load regiszter

Table 4.34. SYST_RVR register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	RELOAD	Value to load into the SYST_CVR register when the counter is enabled and when it reaches 0, see *Calculating the RELOAD value*.

System Timer regiszterek

- Control and Status regiszter

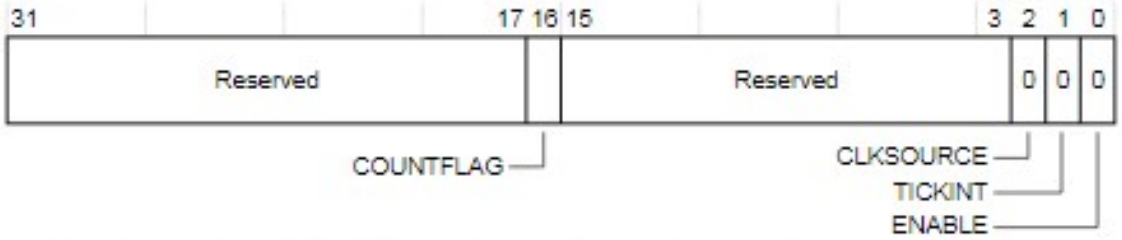


Table 4.33. SysTick SYST_CSR register bit assignments

Bits	Name	Function
[31:17]	-	Reserved.
[16]	COUNTFLAG	Returns 1 if timer counted to 0 since last time this wa
[15:3]	-	Reserved.
[2]	CLKSOURCE	Indicates the clock source: 0 = external clock 1 = processor clock.
[1]	TICKINT	Enables SysTick exception request: 0 = counting down to zero does not assert the SysTic 1 = counting down to zero asserts the SysTick excepti Software can use COUNTFLAG to determine if SysTick
[0]	ENABLE	Enables the counter: 0 = counter disabled 1 = counter enabled.

System Timer regiszterek

- Regiszter beállítások

```
SysTick->LOAD = 1800000; // Tick in every 10ms
SysTick->CTRL = 0x7;     // Enable timer, with interrupt and MCU clock config
```

Interrupt beállítások

- Interrupt engedélyezés

```
/* Interrupt configuration */
NVIC_InitTypeDef  NVIC_InitStruct;
NVIC_InitStruct.NVIC_IRQChannel = SysTick_IRQn;
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStruct);
```

- IT kiszolgáló

```
void SysTick_Handler(void)
{
    counter++;
    if(counter >= 100)
    {
        counter = 0;
        printf("Systic IT \r\n");
    }


}
```

NVIC programozás:

Általános Timer: Timer2

Megvalósítás

- 1. Lépés: Timer 2 felprogramozás
  - Órajel engedélyezés APB1
  - Timer programozás: lefelé számláló
  - Timer IT kérés Update eseményre
  - Timer Elindítás
- 2. Lépés: IT kérés
  - Interrupt kontroller felprogramozás
  - IRQ handler