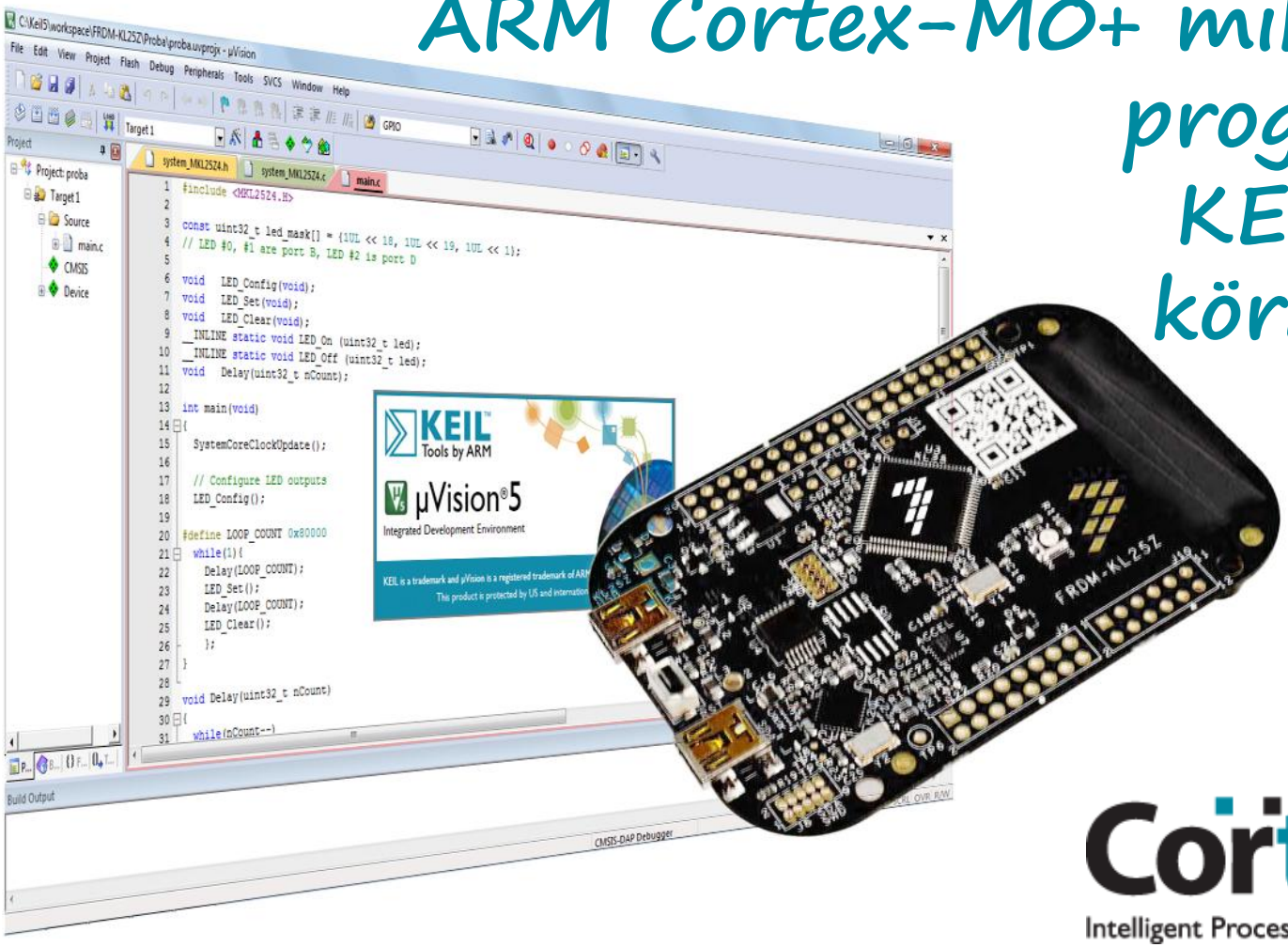
ARM Cortex-M0+ mikrovezérlő programozása KEIL MDK 5 környezetben



Cortex
Intelligent Processors by ARM

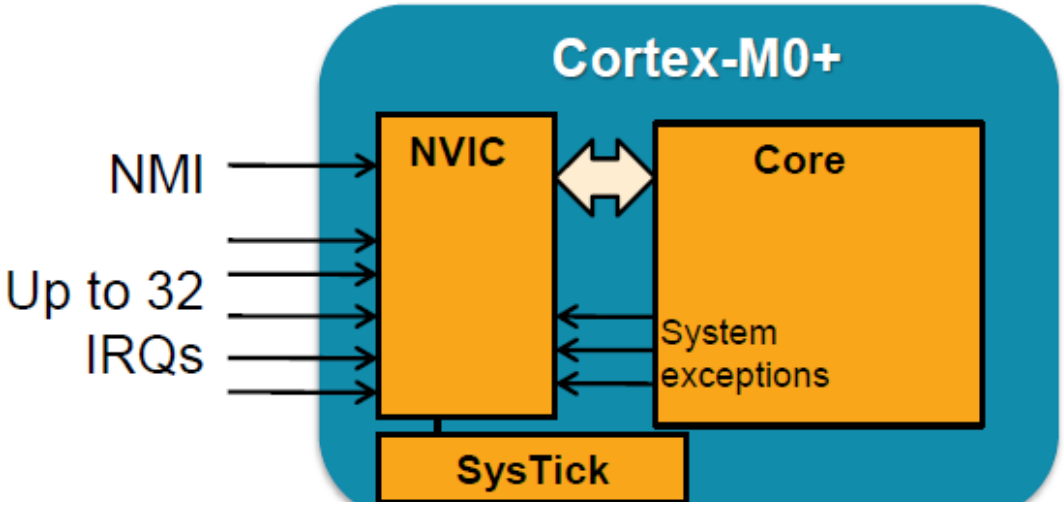
6. Programmegszakítások

Felhasznált anyagok, ajánlott irodalom

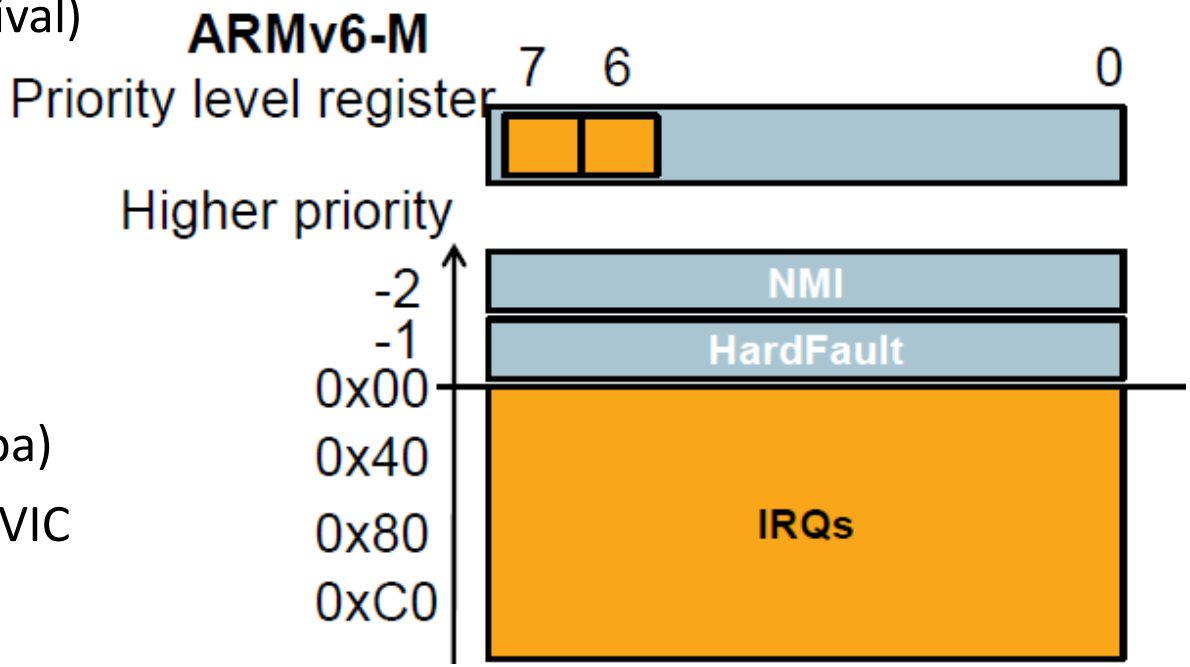
- Joseph Yiu: **The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors** (2nd Ed.)
- Joseph Yiu: **The Anatomy of the ARM Cortex-M0+ Processor**
- Muhammad Ali Mazidi, Shujen Chen, Sarmad Naimi, Sepehr Naimi: **Freescale ARM Cortex-M Embedded Programming**
- ARM University Program: **Course/Lab Material for Teaching Embedded Systems/MCUs** (for the **FRDM-KL25Z** board)
- ARM Information Center: **Cortex-M0+ Devices Generic User Guide**
- Freescale: **MKL25Z128VLK4 MCU datasheet**
- Freescale: **KL25 Sub-Family Reference Manual**
- Freescale: **FRDM-KL25Z User Manual**

Megszakítási rendszer

- **Többszintű vektoros megszakításkezelő (NVIC)**
  - Megszakítás priorizálás
  - Megszakítások maszkolása
  - Többszintű megszakítás
  - Megszakítások láncolása (tail chaining & late arrival)

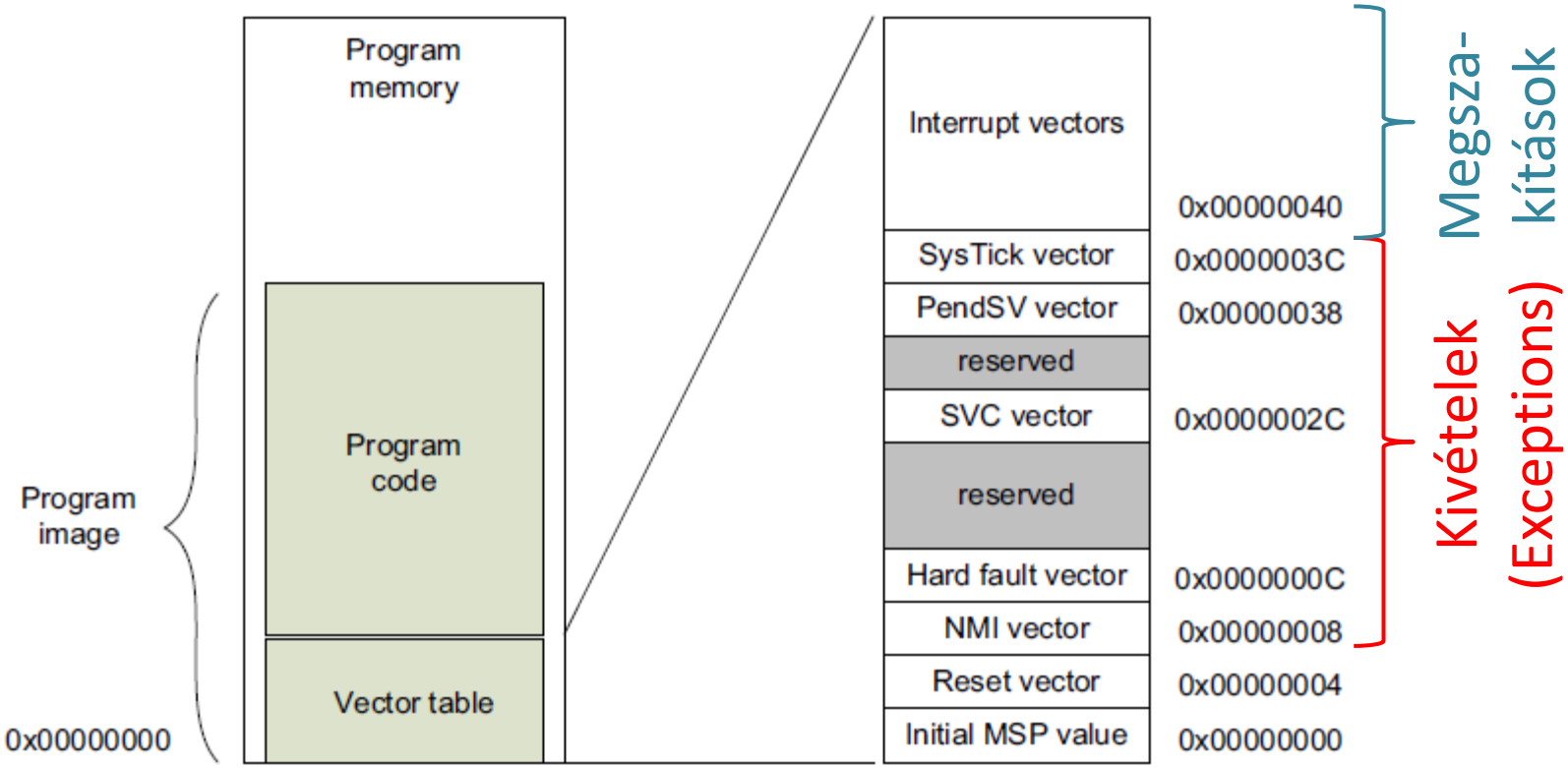


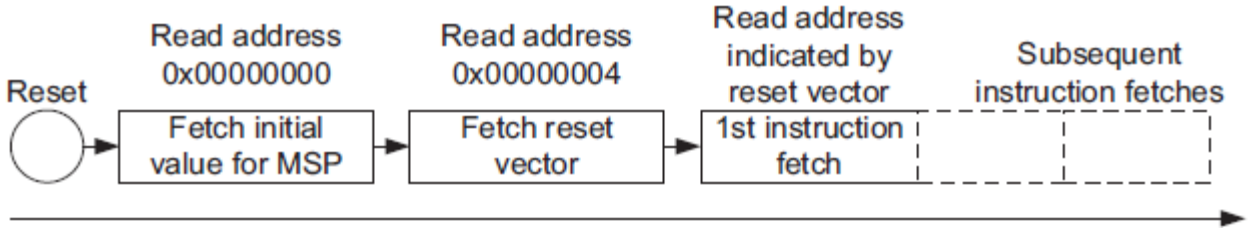
- **Könnyű használat**
  - Megszakításkezelő C nyelven írható
  - Automatikus hardveres adatmentés (veremtárba)
  - CMSIS függvények az NVIC kezelés támogatására



A programindítás folyamata

A vektortábla első két eleme a veremtár és a futtatandó program kezdőcímét tartalmazza. RESET után ezek *automatikusan betöltődnek* a megfelelő regiszterekbe (R13 és R15).





A programmegszakítás folyamata

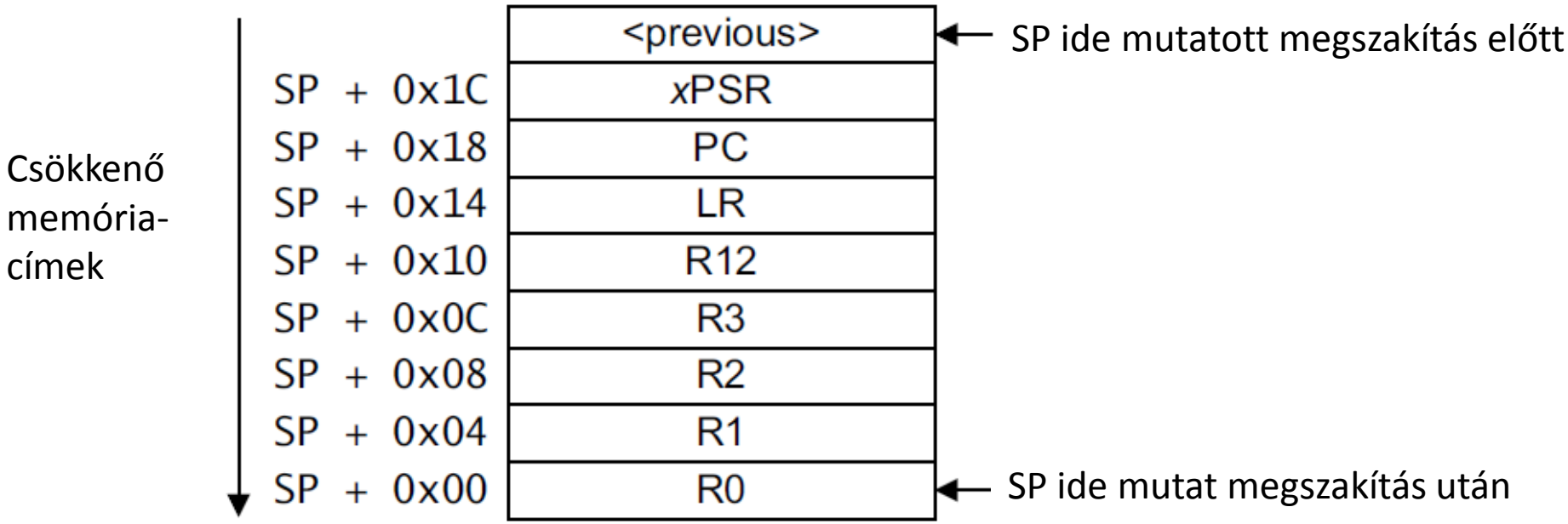
1. Befejeződik az aktuális utasítás (ha nem túl hosszú…)
2. Kontextus elmentése (8 x 32-bites szó) az aktuális veremtárba (MSP vagy PSP)
   - xPSR, visszatérési cím , LR (R14), R12, R3, R2, R1, R0
3. Átkapcsolás handler/privilegizált módba, MSP használatra
4. PC (programszámláló) betöltése a vektortábla megfelelő rovatából
5. LR feltöltése EXC_RETURN kóddal
6. IPSR feltöltése a kivétel/IRQ számával
7. A kivétel/IRQ kezelőprogram végrehajtása

Latency: Általában 15 utasításciklus a megszakítási kérelem megjelenésétől a megszakítást kiszolgáló kód első utasításának megkezdéséig.

1. Az aktuális utasítás befejezése

- A legtöbb utasítás rövid és gyorsan befejeződik
- Néhány utasítás azonban sok utasításciklusig tart
  - Load Multiple (LDM), Store Multiple (STM), Push, Pop, MULS (némely CPU esetében akár 32 ciklus is lehet)
- A sokciklusú utasítás befejezése késleltetné a megszakítások kiszolgálását, ezért ha ilyen utasítás végrehajtása van folyamatban, amikor megszakítási kérelem érkezik, akkor a CPU:
  - *Eldobja az utasítást*
  - Reagál a megszakításra
  - Végrehajtja a megszakítást kiszolgáló eljárást
  - Visszatér a megszakításból
  - *Újrakezdi az eldobott utasítást*

2. Kontextus mentése

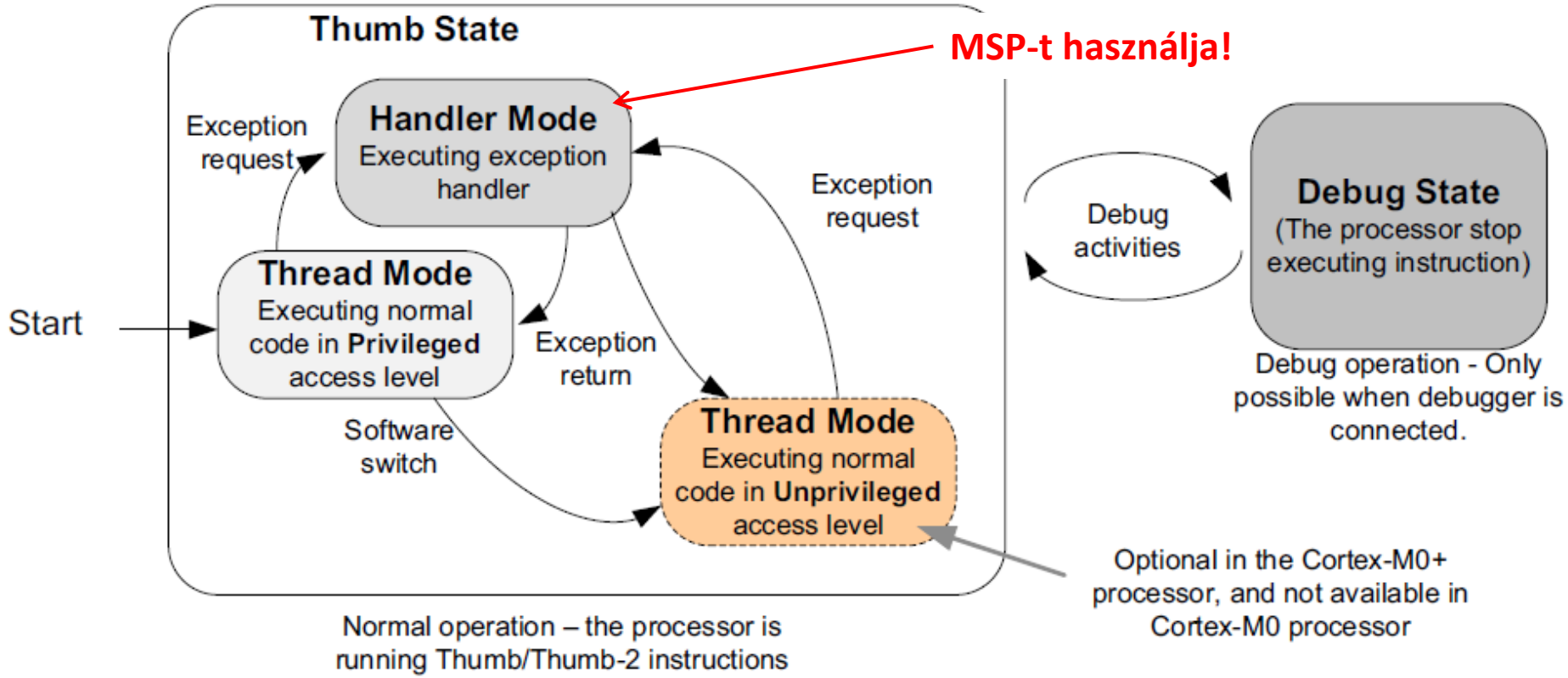


- Két veremmutató van: Main (MSP), process (PSP)
- A működési módtól függ, hogy melyik aktív: CONTROL regiszter bit 1 jelöli ki
- A verem az alacsonyabb címek felé növekszik

3. Átkapcsolás Handler módba

Működési módok:

Két állapot: Kódfuttatás / nyomkövetés
Két működési mód: Kezelői mód (megszakítás kiszolgálás) / Programszál mód
Két privilégium szint: Privilegizált / Nem privilegizált



4. PC betöltése a vektortáblából

A vektortábla a memória elején helyezkedik el, az első két szó kivételével a megszakítást kiszolgáló eljárás belépési pontjára mutatnak. 16 vektor az ARM Cortex M0+ rendszermaghoz, 32 vektor pedig a (gyártóspecifikus) perifériákhoz tartozik.

Cím	Vektor	IRQ	Prioritás	Forrás	Rövid leírás
0x0000_0000	0	-	-	ARM core	Kezdeti veremmutató
0x0000_0004	1	-	-3	ARM core	Kezdeti programszámláló
0x0000_0008	2	-14	-2	ARM core	Nem maszkolható megszakítás (NMI)
0x0000_000C	3	-13	-1	ARM core	Hardver hiba
. . .		-	-	. . .	Fenntartva
0x0000_002C	11	-5	állítható	ARM core	Supervisor hívása (SVC)
. . .		-	-	. . .	Fenntartva
0x0000_0038	14	-2	állítható	ARM core	Beállítható megszakításkérelem
0x0000_003C	15	-1	állítható	ARM core	SysTick megszakítás

A tábla folytatódik a következő oldalon…

A perifériás megszakítások vektorai

Non-Core Vectors	Vector	IRQ	IPR	Source	Source description
0x0000_0040	16	0	0	DMA	DMA channel 0 transfer complete and error
0x0000_0044	17	1	0	DMA	DMA channel 1 transfer complete and error
0x0000_0048	18	2	0	DMA	DMA channel 2 transfer complete and error
0x0000_004C	19	3	0	DMA	DMA channel 3 transfer complete and error
0x0000_0050	20	4	1	—	—
0x0000_0054	21	5	1	FTFA	Command complete and read collision
0x0000_0058	22	6	1	PMC	Low-voltage detect, low-voltage warning
0x0000_005C	23	7	1	LLWU	Low Leakage Wakeup
0x0000_0060	24	8	2	I^2C0	
0x0000_0064	25	9	2	I^2C1	
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_006C	27	11	2	SPI1	Single interrupt vector for all sources
0x0000_0070	28	12	3	UART0	Status and error
0x0000_0074	29	13	3	UART1	Status and error
0x0000_0078	30	14	3	UART2	Status and error
0x0000_007C	31	15	3	ADC0	
0x0000_0080	32	16	4	CMP0	
0x0000_0084	33	17	4	TPM0	
0x0000_0088	34	18	4	TPM1	
0x0000_008C	35	19	4	TPM2	

IPR az érintett **NVIC_IPRx** regiszter sorszáma

A tábla folytatódik …

NVIC regiszterek

Cím	Regiszter	Funkció
0xE000E100	NVIC_ISER	Megszakítás engedélyezés (Set-enable)
0xE000E180	NVIC_ICER	Megszakítás tiltás (Clear-enable)
0xE000E200	NVIC_ISPR	Megszakítás beállítás (Set-pending)
0xE000E280	NVIC_ICPR	Megszakítás törlés (Clear-pending)
0xE000E400 – 0xE000E41F	NVIC_IPR0 – NVIC_IPR7	Megszakítások prioritása (Interrupt Priority Registers)

NVIC_ISER, NVIC_ICER, NVIC_ISPR, NVIC_ICPR: a megfelelő bit ’1’-be állítása végzi el a kért feladatot.

NVIC_IPR*x*: egy-egy 32 bites regiszter bájtonként egy-egy megszakítás prioritását állítja be. Bájtonként csak a legfelső két bit van használatban. A kisebb szám nagyobb prioritást jelent.

Néhány kivétel prioritása ***rögzített:***

- Reset: -3 (legmagasabb prioritás)
- NMI: -2
- Hard Fault: -1

A perifériák megszakításai beállítható prioritásúak (az **NVIC_IPRx** regiszterekbe írt értékekkel): 0x00, 0x40, 0x80, 0xC0 (legalacsonyabb prioritás)

A többi kivétel prioritása az **SCB->SHP[0]** és az **SCB->SHP[1]** regiszterben állítható.

Lásd: **ARM Cortex-M0+ Devices Generic User Guide** 4.3.8 alfejezetében!

A perifériás megszakítások vektorai

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0090	36	20	5	RTC	Alarm interrupt
0x0000_0094	37	21	5	RTC	Seconds interrupt
0x0000_0098	38	22	5	PIT	Single interrupt vector for all channels
0x0000_009C	39	23	5	—	—
0x0000_00A0	40	24	6	USB OTG	
0x0000_00A4	41	25	6	DAC0	
0x0000_00A8	42	26	6	TSI0	
0x0000_00AC	43	27	6	MCG	
0x0000_00B0	44	28	7	LPTMR0	
0x0000_00B4	45	29	7	—	
0x0000_00B8	46	30	7	Port control module	Pin detect (Port A)
0x0000_00BC	47	31	7	Port control module	Pin detect (Port D)

A mai anyagrészben az előző oldalon és itt piros keretezéssel megjelölt megszakításokkal foglalkozunk.

CMSIS core támogatás

A **CMSIS Core** modul az alábbi támogatói függvényeket nyújtja a megszakítások kezeléséhez. A dupla aláhúzással kezdődőek ún. intrinsic függvények.

core_cmFunc.h

- **__disable_irq()** – megszakítások globális tiltása
- **__enable_irq()** – megszakítások globális engedélyezése

core_cm0plus.h

- **NVIC_EnableIRQ**(*IRQn_Type IRQn*) – adott megszakítási vektor engedélyezése
- **NVIC_DisableIRQ**(*IRQn_Type IRQn*) – adott megszakítási vektor tiltása
- **NVIC_GetPendingIRQ**(*IRQn_Type IRQn*) – IRQn megszakítás állapotának viszgálata
- **NVIC_SetPendingIRQ**(*IRQn_Type IRQn*) – IRQn megszakítási kérelem aktiválása
- **NVIC_ClearPendingIRQ**(*IRQn_Type IRQn*) – IRQn megszakítási kérelem törlése
- **NVIC_SetPriority**(*IRQn_Type IRQn, uint32_t priority*) – IRQn prioritásának beállítása
- **NVIC_GetPriority**(*IRQn_Type IRQn*) – IRQn prioritásának lekérdezése

startup_MKL25Z4.s – ez az állomány tartalmazza a vektortáblát, a megszakításokat kiszolgáló függvények neveit és a hozzájuk rendelt alapértelmezett kiszolgáló rutint. **Ezek [WEAK] („gyenge”) definíciók, tehát fölüldefiniálhatók!**

Megszakítás a GPIO portokkal

A GPIO portok segítségével külső jeleket az **NVIC** (és más eszközök) bemeneteire vezethetünk. Az **MKL25Z128VLK4** mikrovezérlő esetében csak az **A** és a **D** porthoz tartozó kivezetések használhatók programmegszakítások keltésére.

Az érintett regiszterek:

- **PORT*x*_PCR*n*** – Portkivezetés vezérlő regiszterek:
  - portonként 32 db. regiszter
  - mindegyik regiszter megfelel egy-egy kivezetésnek.
- **PORTx_ISFR** – Megszakításjelző bitek:
  - portonként 1 db. regiszter
  - Minden bit megfelel egy-egy kivezetésnek
  - A bit ’1’-be áll, ha a megszakítási esemény bekövetkezett és kiszolgálásra vár

A PORTx_PCRn regiszter

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					MUX			0	DSE	0	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	x*	x*	x*	0	x*	0	x*	0	x*	x*	x*

* Notes:
- x = Undefined at reset.

ISF – megszakítási kérelem esetén ’1’
IQRC – megszakítási mód vezérlés:
0000 Megszakítás/DMA kérelem letiltva.
0001 DMA kérelem felfutó élre.
0010 DMA kérelem felfutó élre.
0011 DMA kérelem bármely élre
1000 Megszakítás alacsony szintre.
1001 Megszakítás felfutó élre.
1010 Megszakítás lefutó élre.
1011 Megszakítás bármely élre.
1100 Megszakítás magas szintre.

PS – pull select (0: pull-down, 1: pull-up)
PE – pull enable (0: disable, 1: enable)
SRE – slew rate enable (0: fast, 1: slow)
PFE – passive filter enable (0: disable, 1: enable)
DSE – Drive Strength Enable (0: low, 1: high)
MUX – multiplexer select (0: analog, 1: GPIO, 2: Alt2, 3: Alt3, 4: Alt4, 5: Alt5, 6: Alt6, 7: Alt7 chip specific function selection)

ISF: DMA átvitel után automatikusan törlődik

Program6_1: megszakítás nyomógombbal

PTA12 vagy **PTA13** bemeneten lehúzásra bekötött nyomógombbal megszakítást keltünk és kiszolgáláskor a zöld LED-del 3-at villantunk

```
#include "MKL25Z4.h"
void delayMs(int n);
int main(void) {
    __disable_irq();                // Megszakítások globális tiltása
    SIM->SCGC5 |= 0x400;            // PORTB engedélyezése (LED-ek)
    PORTB->PCR[18] = 0x100;         // PTB18 legyen GPIO módban
    PORTB->PCR[19] = 0x100;         // PTB19 legyen GPIO módban
    PTB->PDDR |= 0xC0000;           // PTB18 és PTB19 legyen kimenet
    PTB->PDOR |= 0xC0000;           // LED-ek lekapcsolása
    SIM->SCGC5 |= 0x200;            // PORTA engedélyezése (nyomógomb bemenetek)
    PORTA->PCR[12] |=  0x00100;     // PTA12 legyen GPIO
    PORTA->PCR[12] |=  0x00003;     // PTA12 belső felhúzás engedélyezése
    PTA->PDDR &= ~0x1000;           // PTA12 legyen bemenet
    PORTA->PCR[12] &= ~0xF0000;     // Megszakítási beállítások törlése
    PORTA->PCR[12] |=  0xA0000;     // Megszakítás aktiválás lefutó élre
    PORTA->PCR[13] |=  0x00100;     // PTA13 legyen GPIO
    PORTA->PCR[13] |=  0x00003;     // PTA13 belső felhúzás engedélyezése
    PTA->PDDR &= ~0x2000;           // PTA13 legyen bemenet
    PORTA->PCR[13] &= ~0xF0000;     // Megszakítási beállítások törlése
    PORTA->PCR[13] |=  0xA0000;     // Megszakítás aktiválás lefutó élre
    NVIC->ISER[0] |= 0x40000000;    // INT30 engedélyezése (ISER[0] bit30)
    __enable_irq();                 // Megszakítások globális engedélyezése
    while(1) {
        PTB->PTOR = 0x40000;        // Piros LED állapotváltás
        delayMs(500);
    }
}
```

PORTA IRQn = 30

Program6_1: megszakítás nyomógombbal

A megszakítást kiszolgáló függvény neve kötött (a **startup_MKL25Z4.s** állomány definiálja – mi pedig itt fölüldefiniáljuk), s minden **PORTA**-hoz tartozó megszakításkor meghívásra kerül. Kilépés előtt ne feledjük el törölni a megszakításjelző bitet!

```
//-------------------------------------------------------------
// A PTA12-re, vagy PTA13-ra kötött nyomógombok aktiválják
//-------------------------------------------------------------
void PORTA_IRQHandler(void) {
    int i;
    for (i = 0; i < 3; i++) {  // Zöld LED felvillantása háromszor
        PTB->PDOR &= ~0x80000; // Zöld LED be
        delayMs(500);
        PTB->PDOR |= 0x80000;  // Zöld LED ki
        delayMs(500);
    }
    PORTA->ISFR |= 0x00003000;  // Megszakításjelző bitek törlése
}

//-----------------------------------------------------------------
// Késleltető függvény alapértelmezett órajelhez (20.97152 MHz)
//-----------------------------------------------------------------
void delayMs(int n) {
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++);
}
```

Program6_2: megszakítás nyomógombokkal

Ez a program csupán abban tér el az előzőtől, hogy a **PTA12** és **PTA13**-ra kötött nyomógombokat megkülönböztetjük. Ezért csak releváns részét mutatjuk be a programnak.

```
void PORTA_IRQHandler(void) {
    int i;
    while (PORTA->ISFR & 0x00003000) { // Van-e PTA12 vagy PTA13 megszakítás?
        if(PORTA->ISFR & 0x00001000) { // PTA12 megszakítás?
            for (i = 0; i < 3; i++) {  // Zöld LED felvillantása háromszor
                PTB->PDOR &= ~0x80000; // Zöld LED be
                delayMs(500);
                PTB->PDOR |= 0x80000;  // Zöld LED ki
                delayMs(500);
            }
            PORTA->ISFR |= 0x00001000;  // Megszakításjelző bit törlése
        }
        if(PORTA->ISFR & 0x00002000) { // PTA13 megszakítás?
            for (i = 0; i < 3; i++) {  // Kék LED felvillantása háromszor
                PTD->PDOR &= ~0x02;    // Kék LED be
                delayMs(500);
                PTD->PDOR |= 0x02;     // Kék LED ki
                delayMs(500);
            }
            PORTA->ISFR |= 0x00002000;  // Megszakításjelző bit törlése
        }
    }
}
```

Program6_3: LED átbillentés minden élre

Most a változatosság kedvéért a **PTD4** bemenetet használjuk megszakításhoz, s mindkét élre érzékenyítjük…

```
#include "MKL25Z4.h"
int main(void) {
    __disable_irq();              // Megszakítások globális tiltása
//--- A kék LED konfigurálása -----------
    SIM->SCGC5 |= 0x1000;         // PORTD engedélyezése
    PORTD->PCR[1] = 0x100;        // PTD1 legyen GPIO módban
    PTD->PDDR |= 0x02;            // PTD1 kimenet legyen
    PTD->PDOR |= 0x02;            // Kék LED lekapcsolása
//--- PTD4 megszakítás konfigurálása ----
    PORTD->PCR[4] |=  0x00100;    // PTD4 legyen GPIO módban
    PORTD->PCR[4] |=  0x00003;    // Felhúzás engedélyezése
    PTD->PDDR &= ~0x0010;         // PTD4 legyen bemenet
    PORTD->PCR[4] &= ~0xF0000;    // Megszakítási beállítások törlése
    PORTD->PCR[4] |=  0xB0000;    // Mindkét élre legyen érzékeny

    NVIC->ISER[0] |= 0x80000000;// INT31 engedélyezése (ISER[0] bit31) */
    __enable_irq();               // Megszakítások globális engedélyezése

    while(1) {                    // Nincs mit tenni!
    }
}

void PORTD_IRQHandler(void) {
    if (PORTD->ISFR & 0x00000010) {
        PTD->PTOR = 0x0002;     // PTD1 (kék LED) átbillentése
        PORTD->ISFR = 0x0010;   // A megszakításjelző bit törlése
    }
}
```

PORTD IRQn = 31

UART0 programmegszakítások

UART0 vezérlő regiszter 2 (UART0_C2)

Bit	7	6	5	4	3	2	1	0
Read Write	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

- **Megszakítások engedélyezése**
  - **TIE:** Megszakítás ha a küldő adatregiszter üres
  - **TCIE:** Megszakítás az átvitel végén
  - **RIE:** Megszakítás ha van beérkezett adat
  - **ILIE:** Megszakítás ha tétlen a vonal
- **Modul engedélyezés**
  - **TE:** Adó (transmitter) engedélyezés
  - **RE:** Vevő (receiver) engedélyezés
- **Továbbiak:**
  - **RWU:** a vevőt standby módba teszi, felébresztés az előírt feltételek esetén
  - **SBK:** Break karakter (csupa nulla) küldése

UART0 IRQn = 12

Program6_4: UART0 vétel megszakításban

Ez a program a korábban bemutatott **Program 4_2** mintapélda módosított változata. Most az **UART0** vétel a megszakítást kiszolgáló függvénybe került át.

```
#include <MKL25Z4.h>

void UART0_init(void);                  // UART0 Rx bemenet inicializálás
void LED_init(void);                    // RGB LED inicializálás
void LED_set(int value);                // RGB LED színbeállítás

int main (void) {
    UART0_init();                       // UART0 Rx bemenet inicializálása
    LED_init();                         // RGB LED inicializálása
    while (1) {
       // Az UART0 vétel a megszakításba került át ...
    }
}

//-------------------------------------
// UART0 megszakítás kiszolgálása
//-------------------------------------
void UART0_IRQHandler(void) {
    char c;
    c = UART0->D;                       // Beolvassuk a vett karaktert
    LED_set(c);                         // és beállítjuk a LED-et
}
```

Az adat kiolvasása törli a jelzőbitet?!

Folytatás a következő lapon…

Program6_4: UART0 vétel megszakításban

Pirossal kiemeltük azokat a sorokat, amelyek különböznek **Program4_2**-től.

```
/* UART0 Rx inicializálása 9600 Baud átviteli sebességre
 * Baudrate = UART0 clock freq / [OSR+1] / BDH:BDL
 * Esetünkben 20.97 MHz / [15 + 1] / 0x00:0x88 =  9637 bps,
 * ami elfogadható eltérés a névlegestol.
 */
void UART0_init(void) {
    SIM->SCGC4 |= 0x0400;          // UART0 periféria engedélyezése
    SIM->SOPT2 |= 0x04000000;      // MCGFLLCLK választása UART0 baudrate órajelnek
    UART0->C2 = 0;                 // UART0 letiltása a konfigurálás idotartamára
    UART0->BDH = 0x00;             // Baudrate = 9600 bps (bit8 - bit12)
    UART0->BDL = 0x88;             // Baudrate = 9600 bps (bit0 - bit7)
    UART0->C4 = 0x0F;              // Túlmintavételezési arány = 16
    UART0->C1 = 0x00;              // 8-bites adatformátum
    UART0->C2 = 0x24;              // Adatfogadás és megszakítás engedélyezése
    NVIC->ISER[0] |= 0x00001000;   // INT12 engedélyezése (ISER[0] bit12)

    SIM->SCGC5 |= 0x0200;          // PORTA engedélyezése
    PORTA->PCR[1] = 0x0200;        // PTA1 legyen UART0_Rx üzemmódban
}
```

UART0 IRQn = 12

A LED-ek inicializálással és beállításával kapcsolatos programsorokat itt most nem részletezzük.

Timer megszakítások

Az 5. fejezetben lekérdezéssel (polling) használtuk az időzítőket. Hatékonyabban tudjuk használni ezeket megszakításos módban – akár több időzítőt is futtathatunk egyidejűleg.

A megszakítás engedélyezéséhez a **TOIE** (Timer Overflow Interrupt Enable) bitet kell ’1’-be állítani a **TPM*x*_SC** regiszterben. Minden túlcsorduláskor (amikor **TOF** jelző ’1’-be áll) megszakítás keletkezik.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							DMA	TOF	TOIE	CPWMS	CMOD		PS		
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A TPMx_SC regiszter bitkiosztása

A 10. oldalon a periféria megszakítások vektortáblájában láthatjuk, hogy a **TPM0, TPM1** és **TPM2** időzítőkhöz rendre az **IRQ17, IRQ18** és **IRQ19** megszakítások tartoznak.

Program6_5: TPM0 és TPM1 megszakítások

A LED-ek inicializálását itt nem részletezzük.

```
#include <MKL25Z4.H>
int main (void) {
    LED_init();
    __disable_irq();                // Megszakítások globális tiltása
    SIM->SOPT2 |= 0x01000000;       // MCGFLLCLK legyen a közös TPM órajel
    SIM->SCGC6 |= 0x01000000;       // TPM0 engedélyezése
    TPM0->SC = 0;                   // Konfigurálás idejére letiltjuk
    TPM0->SC = 0x07;                // Előosztó /128 legyen
    TPM0->MOD = 0xFFFF;             // Maximális modulo érték
    TPM0->SC |= 0x80;               // TOF túlcsordulásjelző törlése
    TPM0->SC |= 0x40;               // Megszakítás engedélyezése
    TPM0->SC |= 0x08;               // Számlálás engedélyezése
    NVIC->ISER[0] |= 0x00020000;    // IRQ17 engedélyezése (ISER[0] bit17)
    SIM->SCGC6 |= 0x02000000;       // TPM1 engedélyezése
    TPM1->SC = 0;                   // Konfigurálás idejére letiltjuk
    TPM1->SC = 0x07;                // Előosztó /128 legyen
    TPM1->MOD = 0x7FFF;             // Modulo a maximum fele legyen!
    TPM1->SC |= 0x40;               // Megszakítás engedélyezése
    TPM1->SC |= 0x08;               // Számlálás engedélyezése
    NVIC->ISER[0] |= 0x00040000;    // IRQ18 engedélyezése (ISER[0] bit18)
    __enable_irq();                 // Megszakítások globális engedélyezése
    while (1) {
        PTD->PTOR = 0x02;           // A kék LED átbillentése
        delayMs(1500);
    }
}
```

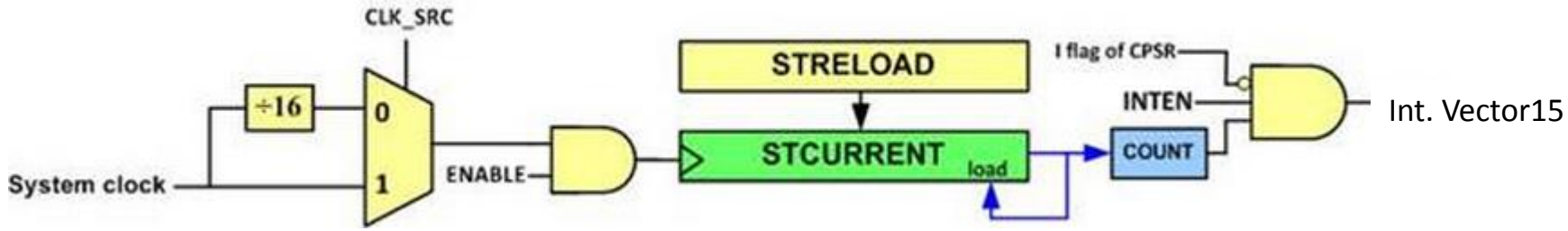
TPM0 65536 * 128/20,97 MHz = 400 ms időközönként ad megszakítást
TPM1 csak 32k-ig számol, ezért feleannyi (200 ms) időközönként kelt megszakítást.

Program6_5: TPM0 és TPM1 megszakítások

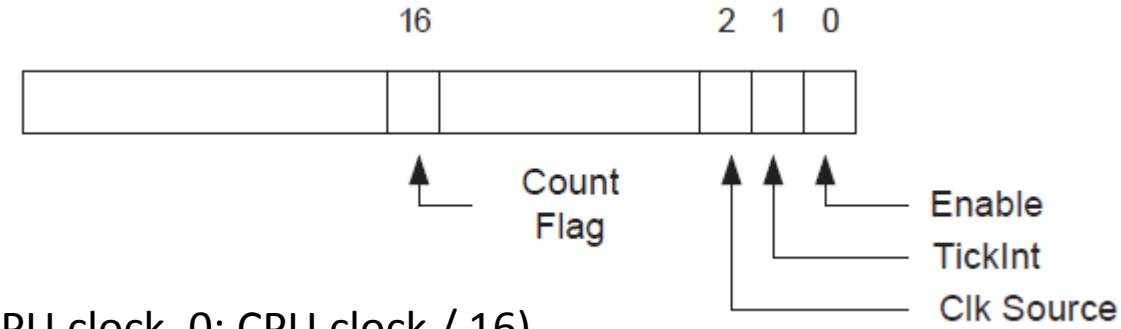
```
//----------------------------------
// TPM0 megszakítások kiszolgálása
//----------------------------------
void TPM0_IRQHandler(void) {      // TPM0 a piros LED-et villogtatja
    PTB->PTOR = 0x40000;          // A piros LED átbillentése
    TPM0->SC |= 0x80;             // A túlcsordulásjelző bit törlése
}

//----------------------------------
// TPM1 megszakítások kiszolgálása
//----------------------------------
void TPM1_IRQHandler(void) {      // TPM1 a zöld LED-et villogtatja
    PTB->PTOR = 0x80000;          // A zöld LED átbillentése
    TPM1->SC |= 0x80;             // A túlcsordulásjelző bit törlése
}

//-------------------------------------------------------------------
// Késleltető függvény alapértelmezett órajelhez (20.97152 MHz)
//-------------------------------------------------------------------
void delayMs(int n) {
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++);
}
```

SysTick megszakítások

Egy másik hasznos megszakítási forrás a **SysTick** időzítő.



A **SysTick->CTRL** vezérlő és állapotjelző regiszter bitjeinek kiosztása:



COUNTFLAG – alulcsordulás jelző

CLKSOURCE – órajel választás (1: CPU clock, 0: CPU clock / 16)

TICKINT – alulcsorduláskor megszakítás engedélyezés (1: enged, 0: tilt)

ENABLE – Számlálás engedélyezés (1: enged, 0: tilt)

Megjegyzések:

- **SysTick** a rendszer kivételek közé van sorolva, ezért engedélyezését és prioritását nem kezelik az **NVIC** regiszterei. Az **SCB->SHP[1]** regiszter 31:24. bitjei szabják meg a prioritását.
- A megszakításjelző bitet **nem kell törölni** a megszakítás kiszolgálásakor.

Program6_6: SysTick megszakítás

A **SysTick** idozítot úgy konfiguráljuk, hogy 1 Hz gyakoriséggal adjon megszakítást. Ehhez a /16 osztású alternatív órajelet kell választanunk. A megszakításban a piros LED állapotát átbillentjük.

```
#include <MKL25Z4.H>

int main (void) {
  __disable_irq();                // A megszakítások globális tiltása
  SIM->SCGC5 |= 0x400;            // PORTB engedélyezése
  PORTB->PCR[18] = 0x100;         // PTB18 GPIO módba
  PTB->PDDR |= 0x40000;           // PTB18 kimenet legyen
  SysTick->LOAD = 20970000/16-1;// Újratöltéshez: 1s alatti óraütések száma - 1
  SysTick->VAL  = 0;              // Kezdéshez alaphelyzetbe állunk
  SysTick->CTRL = 3;              // SysTick megszakítás engedélyezése,
                                  // alternatív órajel választása
  __enable_irq();                 // Megszakítások globális engedélyezése

  while (1) {                     // Megszakításra várunk
  }
}

void SysTick_Handler(void) {
  PTB->PTOR = 0x40000;            // A piros LED átbillentése
}
```

Program6_7: egymásba ágyazott megszakítások vizsgálata

Ebben a programban azt vizsgáljuk, hogy a magasabb prioritású megszakítás hogyan szakítja meg, illetve blokkolja az alacsonyabb prioritású megszakítást. A program megszakítás-kiszolgáló függvényeiben látható késleltetések természetesen normális programokban kerülendők, itt is csupán a kísérletezéshez használtuk ezeket.

TPM1 1 Hz-es gyakorisággal kelt megszakítást. A megszakításban a piros LED-et bekapcsoljuk, majd 360 ms késleltetés után kikapcsoljuk.

TPM2 10 Hz gyakorisággal kelt megszakítást. A megszakításban a kék LED-et bekapcsoljuk, majd 20 ms késleltetés után kikapcsoljuk. A kék LED-del együtt a **PTD2** kimenetet is billegtetjük, oszcilloszkópos vizsgálathoz.

Ha TPM1 prioritása nagyobb, akkor **TPM2** megszakításait blokkolja: a kék LED nem villog, amikor a piros LED világít.

Ha pedig TPM2 prioritása magasabb, akkor a **TPM1** megszakítások kiszolgálását szakítják félbe TPM2 megszakításai, ekkor a kék LED folyamatosan villog.

Program6_7: egymásba ágyazott megszakítások vizsgálata

```
#include "MKL25Z4.h"
void Timer1_init(void);
void Timer2_init(void);
void delayMs(int n);

int main (void) {
    __disable_irq();              // Megszakítások globális tiltása
    SIM->SCGC5 |= 0x400;          // Port B engedélyezése
    SIM->SCGC5 |= 0x1000;         // Port D engedélyezése
    PORTB->PCR[18] = 0x100;       // PTB18 legyen GPIO (MUX = 1)
    PTB->PDDR |= 0x40000;         // PTB18 legyen kimenet (bit18)
    PTB->PSOR = 0x40000;          // Piros LED ki
    PORTD->PCR[1] = 0x100;        // PTD1 legyen GPIO  (MUX = 1)
    PTD->PDDR |= 0x02;            // PTD1 legyen GPIO (MUX = 1)
    PORTD->PCR[2] = 0x100;        // PTD2 is legyen GPIO módban
    PTD->PDDR |= 0x04;            // PTD2 is legyen kimenet
    PTD->PSOR = 0x06;             // Kék LED és PTD2 ki
    Timer1_init();
    Timer2_init();
    __enable_irq();               // Megszakítások globális engedélyezése

    while(1) { }                  // Megszakításra várunk
}
```

Program6_7: egymásba ágyazott megszakítások vizsgálata

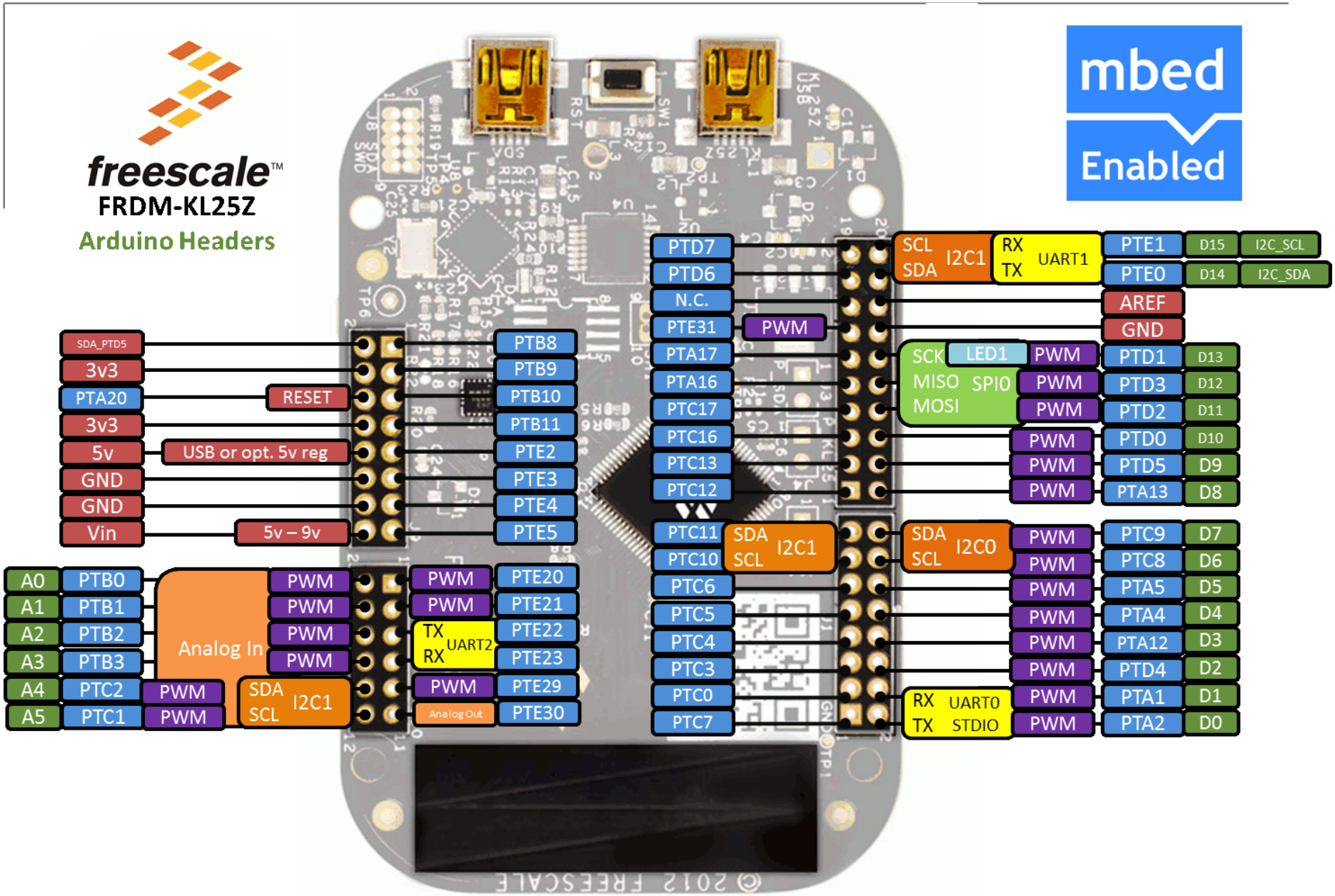
```
void TPM1_IRQHandler(void) {
    PTB->PCOR = 0x40000;          // Piros LED be (active LOW
    delayMs(350);
    PTB->PSOR = 0x40000;          // Piros LED ki
    TPM1->SC |= 0x80;             // Túlcsordulásjelzo bit törlése
}

void TPM2_IRQHandler(void) {
    PTD->PCOR = 0x06;             // Kék LED és PTD2 be (active LOW)
    delayMs(50);
    PTD->PSOR = 0x06;             // Kék LED és PTD2 ki
    TPM2->SC |= 0x80;             // Túlcsordulásjelzo bit törlése
}

//--- TPM1 és TPM2 megszakításainak prioritása 0 és 3 közötti szám lehet.
#define PRIOTPM1 2U               // TPM1 megszakítás prioritása
#define PRIOTPM2 3U               // TPM2 megszakítás prioritása

//--- Késlelteto függvény alapértelmezett órajelhez (20.97152 MHz)
void delayMs(int n) {
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++);
}
```

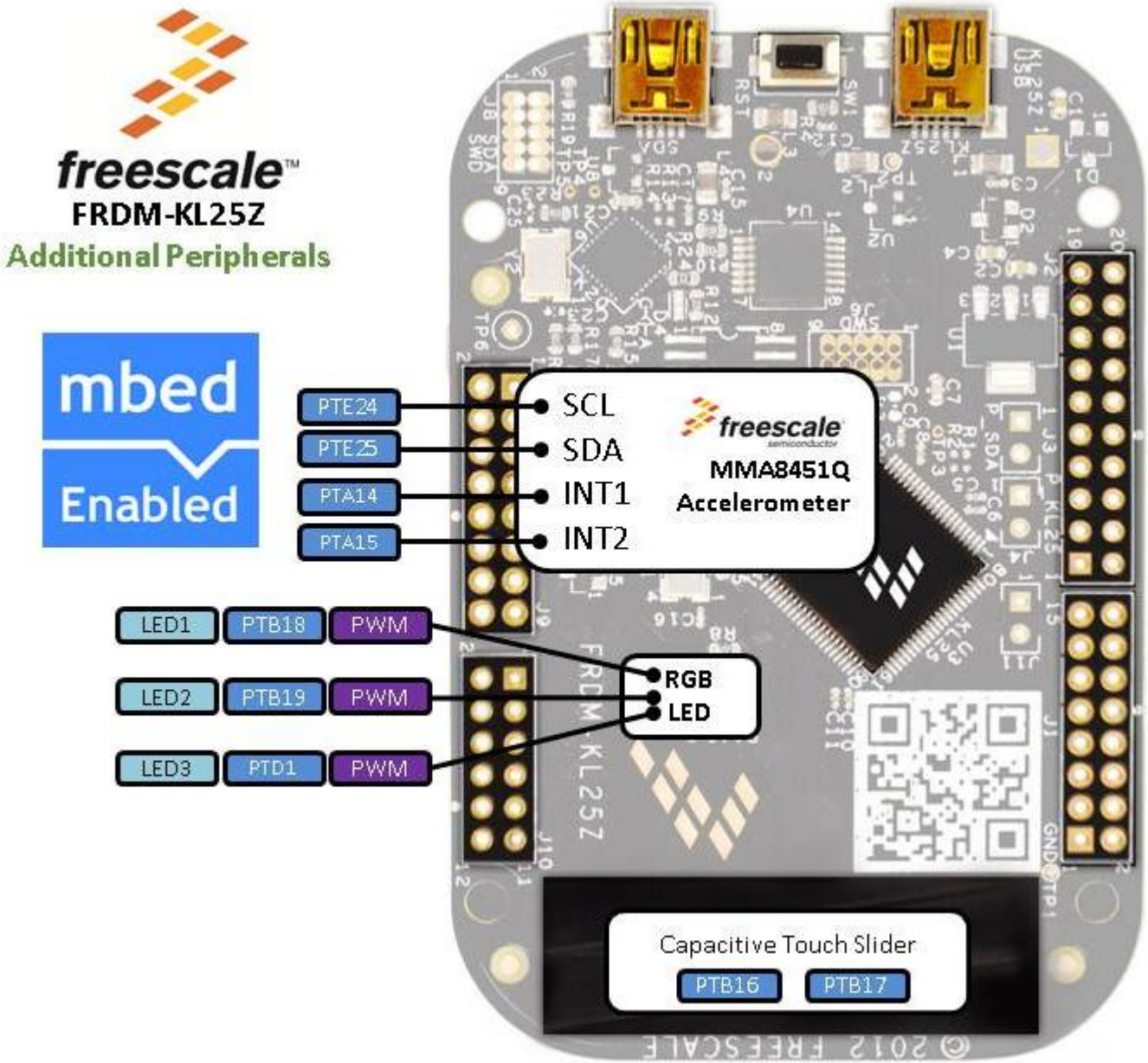
Ezt változtassuk meg, pl. 1-re, és figyeljük meg a változást!

Program6_7: egymásba ágyazott megszakítások vizsgálata

```
void Timer1_init(void) {
    SIM->SCGC6 |= 0x02000000;     // TPM1 engedélyezése
    SIM->SOPT2 |= 0x03000000;     // MCGIRCLK legyen a közös TPM órajel

    TPM1->SC = 0;                 // Konfiguráláshoz letiltjuk
    TPM1->MOD = 32767;            // 1s alatti óraütések száma - 1
    TPM1->SC |= 0x80;             // Túlcsordulásjelzo bit törlése
    TPM1->SC |= 0x48;             // Számlálás indítása, megszakítás engedélyezése
    // A bitek számolgatása helyett a CMSIS NVIC API függvényeket használjuk
    NVIC_SetPriority(TPM1_IRQn,PRIOTPM1); // TPM1 megszakítás prioritás beállítása
    NVIC_EnableIRQ(TPM1_IRQn);    // TPM1 megszakítások engedélyezése
}

void Timer2_init(void) {
    SIM->SCGC6 |= 0x04000000;     // TPM2 engedélyezése
    TPM2->SC = 0;                 // Konfiguráláshoz letiltjuk
    TPM2->MOD = 3276;             // modulo = 32768/10 -1
    TPM2->SC |= 0x80;             // Túlcsordulásjelzo bit törlése
    TPM2->SC |= 0x48;             // Számlálás indítása, megszakítás engedélyezése
     // A bitek számolgatása helyett a CMSIS NVIC API függvényeket használjuk
    NVIC_SetPriority(TPM2_IRQn,PRIOTPM2); // TPM2 megszakítás prioritás beállítása
    NVIC_EnableIRQ(TPM2_IRQn);    // TPM1 megszakítások engedélyezése
}
```

Az **MCGIRCLK** órajel használatához definiálni kell az **RTE system_MKL25Z4.h** állományban a **CLOCK_SETUP** makrót 0 értékkel!


freescale
FRDM-KL25Z
Arduino Headers
mbed
Enabled
SDA_PTD5
3v3
PTA20
3v3
5v
GND
GND
Vin
RESET
USB or opt. 5v reg
5v – 9v
PTB8
PTB9
PTB10
PTB11
PTE2
PTE3
PTE4
PTE5
A0 PTB0
A1 PTB1
A2 PTB2
A3 PTB3
A4 PTC2
A5 PTC1
Analog In
PWM
SDA SCL I2C1
PWM
PWM
PWM
PWM
PWM
PWM
PTE20
PTE21
TX RX UART2
PTE22
PTE23
PWM
PTE29
Analog Out
PTE30
PTD7
PTD6
N.C.
PTE31
PWM
PTA17
PTA16
PTC17
PTC16
PTC13
PTC12
PTC11
PTC10
SDA SCL I2C1
PTC6
PTC5
PTC4
PTC3
PTC0
PTC7
SCL SDA I2C1
RX TX UART1
PTE1 D15 I2C_SCL
PTE0 D14 I2C_SDA
AREF
GND
SCK MISO MOSI SPI0
LED1
PWM
PWM
PWM
PTD1 D13
PTD3 D12
PTD2 D11
PWM
PWM
PWM
PTD0 D10
PTD5 D9
PTA13 D8
SDA SCL I2C0
PWM
PWM
PWM
PWM
PWM
PWM
PWM
PWM
PTC9 D7
PTC8 D6
PTA5 D5
PTA4 D4
PTA12 D3
PTD4 D2
PTA1 D1
PTA2 D0
RX TX UART0 STDIO
© 2012 FREESCALE


freescale
FRDM-KL25Z
Additional Peripherals
mbed
Enabled
PTE24
PTE25
PTA14
PTA15
SCL
SDA
INT1
INT2
freescale
MMA8451Q
Accelerometer
LED1
PTB18
PWM
LED2
PTB19
PWM
LED3
PTD1
PWM
RGB
LED
Capacitive Touch Slider
PTB16
PTB17