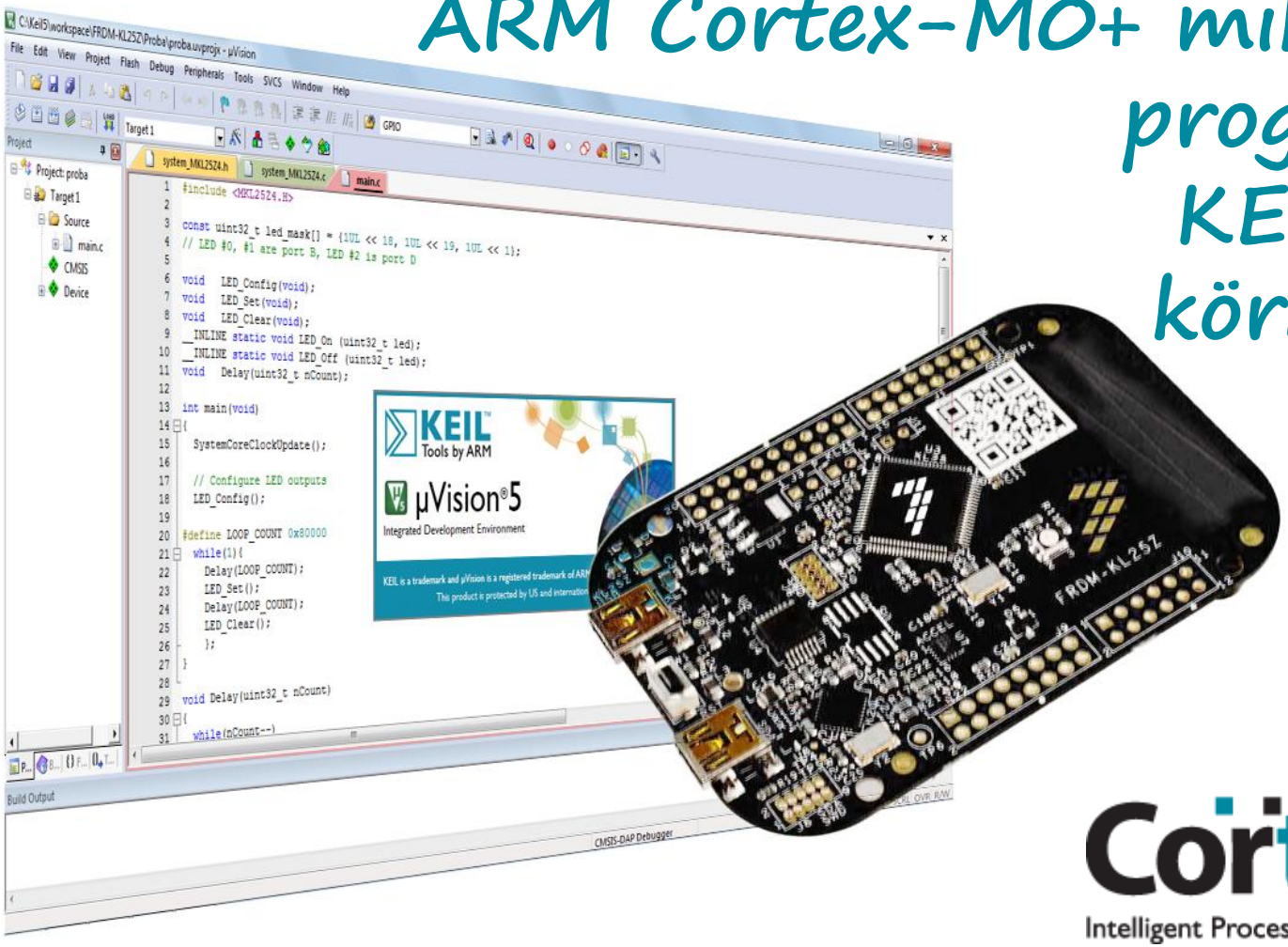
ARM Cortex-M0+ mikrovezérlő programozása KEIL MDK 5 környezetben



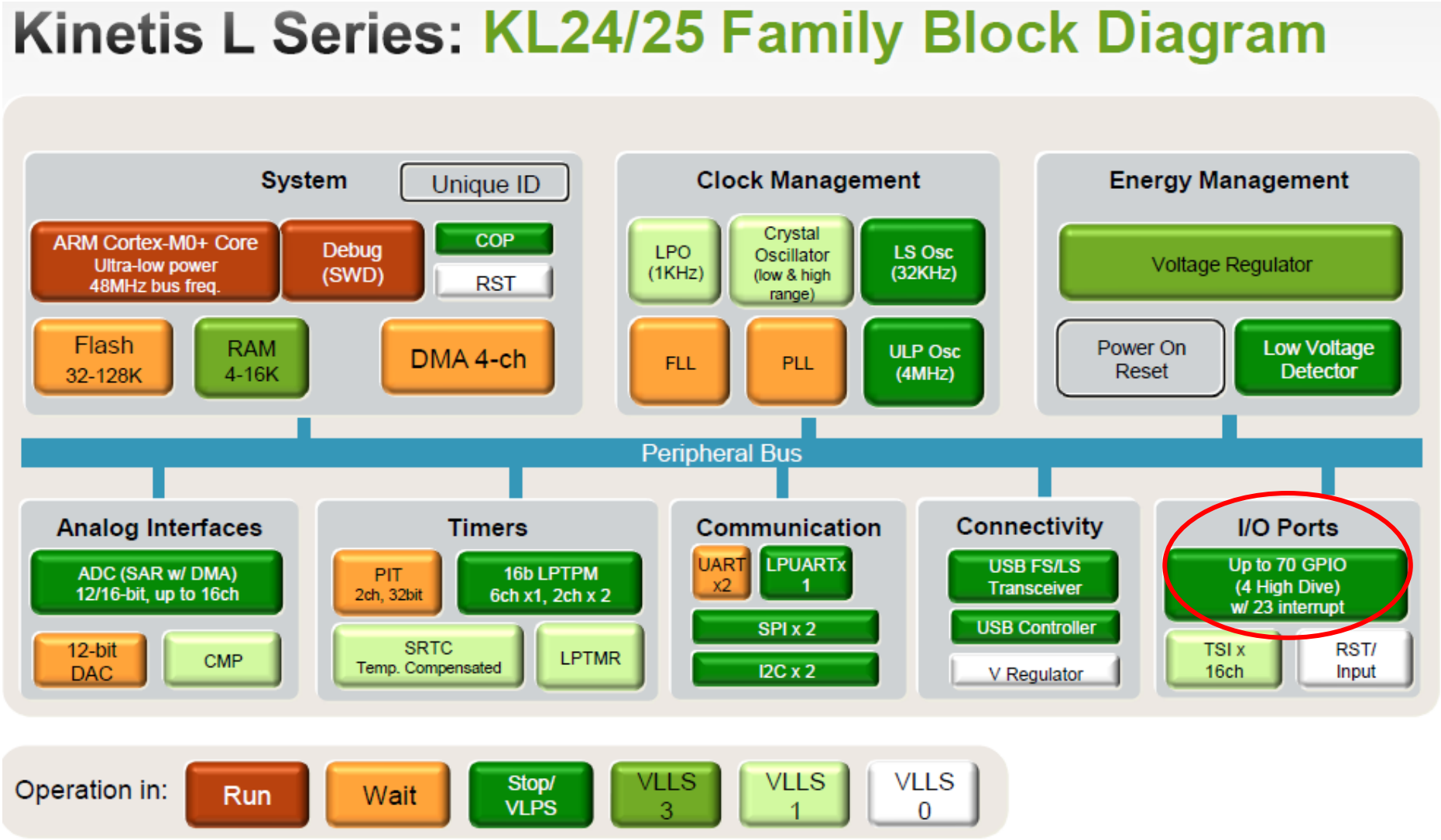
Cortex
Intelligent Processors by ARM

1. Digitális ki- és bemenetek használata

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**
- Trevor Martin: **The Designer's Guide to the Cortex-M Processor Family**
- 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)
- Freescale: **MKL25Z128VLK4 MCU datasheet**
- Freescale: **KL25 Sub-Family Reference Manual**
- Freescale: **FRDM-KL25Z User Manual**

Emlékeztető: MCU blokkvázlat

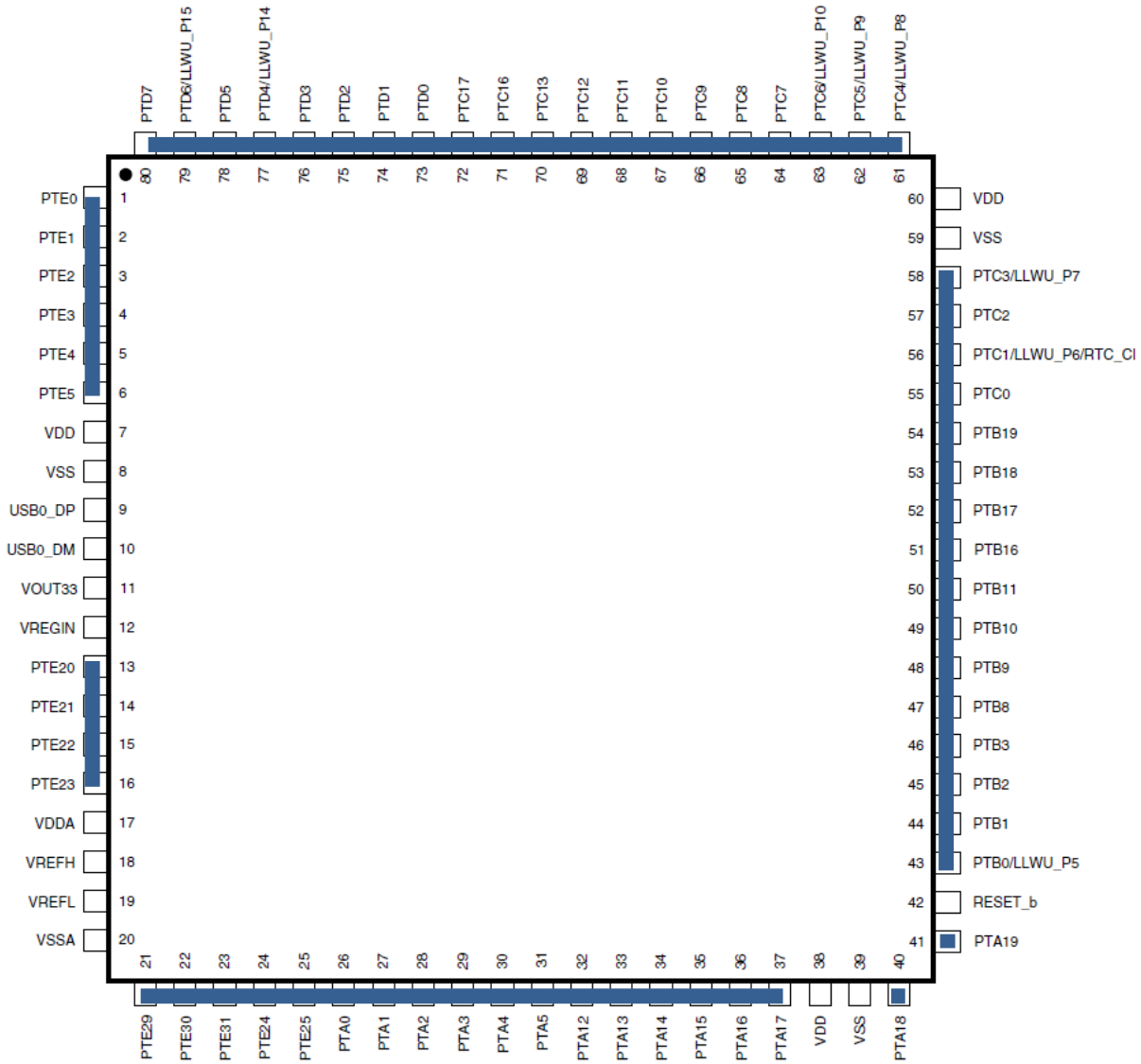


Általános célú digitális I/O (GPIO)

A mikrovezérlők legegyszerűbb perifériái az általános célú digitális ki- és bemenetek, amelyeket csoportokba rendezve (**portok**) kezelhetjük.

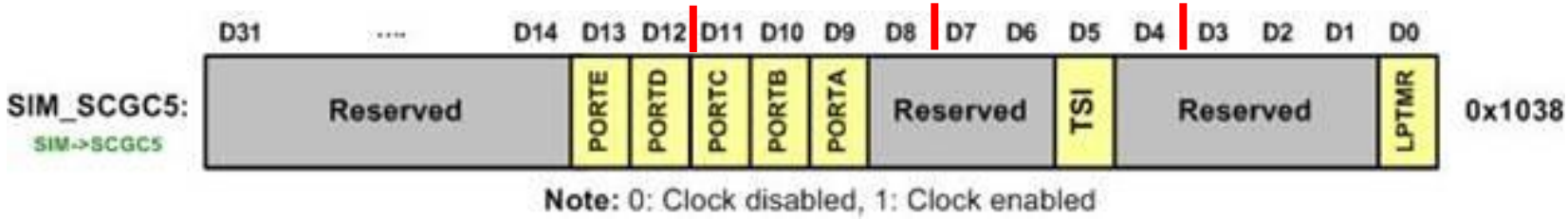
MKL25Z128VLK4 mikrovezérlő

- 80 kivezetése közül 66 lehet GPIO ki/bemenet
- 5 db 32 bites portba vannak szervezve (PortA, PortB, PortC, PortD, PortE)
- A portoknak nincs minden bitje implementálva.



A GPIO portok engedélyezése

A mikrovezérlő *Rendszer Integrációs Moduljának* (SIM) **SIM_SCGC5** nevű regisztere segítségével engedélyezhetjük a GPIO portok órajelét (enélkül nem működnek), vagy tilthatjuk le a nem használt portokat, az energiafelhasználás minimalizálása érdekében.
(Bővebben lásd a **KL25 Sub-Family Reference Manual** 12. fejezetében!)



SIM_SCGC5 címe = 0x40048038

Bit	Port	
13	PORTE	Az **E** port letiltása (0), vagy engedélyezése (1)
12	PORTD	A **D** port letiltása (0), vagy engedélyezése (1)
11	PORTC	A **C** port letiltása (0), vagy engedélyezése (1)
10	PORTB	A **B** port letiltása (0), vagy engedélyezése (1)
9	PORTA	Az **A** port letiltása (0), vagy engedélyezése (1)

Port kivezetések üzemmód vezérlése

Az egyes kivezetések üzemmódjának beállítása a **PORT***x*_**PCR***n* (Port Pin Control) speciális funkciójú regiszterek segítségével történik. Itt *x* a port jele (A – E), *n* pedig a porton belüli bitsorszám (0 – 31).

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.

A felső 16 bit a kivezetéshez tartozó megszakítást/DMA kérelemet konfigurálja, az **alsó 16 bit** pedig a kivezetés funkciójának (analóg, digitális GPIO , speciális, chip specifikus periféria funkció) kiválasztására és egyéb tulajdonságok beállítására szolgál.

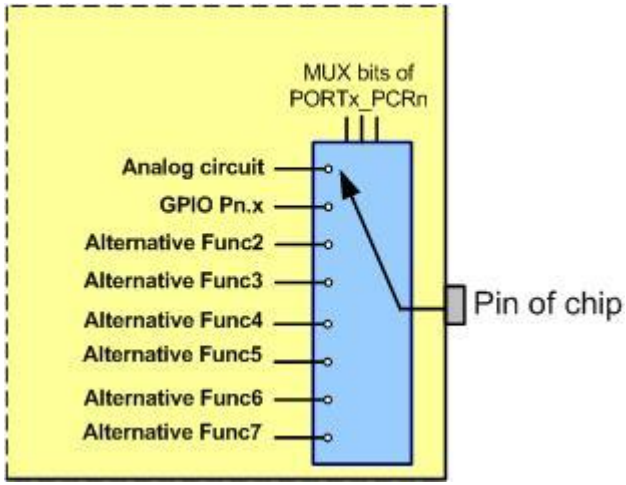
PS – pull select (0: pull-down, 1: pull-up)
PE – pull enable (0: disable, 1: enable)
SRE – slew rate enable (0: fast, 1: slow)
PFE – passive filter enable (0: disable, 1: enable)
DSE – Drive Strength Enable (0: low, 1: high)
MUX – multiplexer select (0: analog, 1: GPIO, 2: Alt2, 3: Alt3, 4: Alt4, 5: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

Alternatív funkciók kiválasztása

Minden kivezetés többféle funkcióval rendelkezik. Példa gyanánt a **PTA0 – PTA4** lábak lehetséges funkcióit soroljuk fel a FRDM-KL25Z Pinouts dokumentum alapján.

Pin	Alt 0	Alt 1	Alt 2	Alt 3	Alt 4	Alt 5	Alt 6	Alt 7
PTA0	TSI0_CH1	PTA0	-	FTM0_CH5	-	-	-	SWD_CLK
PTA1	TSI0_CH2	PTA1	UART0_RX	FTM2_CH0				
PTA2	TSI0_CH3	PTA2	UART0_TX	FTM2_CH1				
PTA3	TSI0_CH4	PTA3	I2C1_SCL	FTM0_CH0	-	-	-	SWD_DIO
PTA4	TSI0_CH5	PTA4	I2C1_SDA	FTM0_CH1				NMI_b

A megfelelő **PORT***x***_PCR***n* regiszter **MUX** bitcsoportjával – mint egy fokozatkapcsolóval – választhatjuk ki, hogy az adott kivezetés melyik funkciót szolgálja ki.



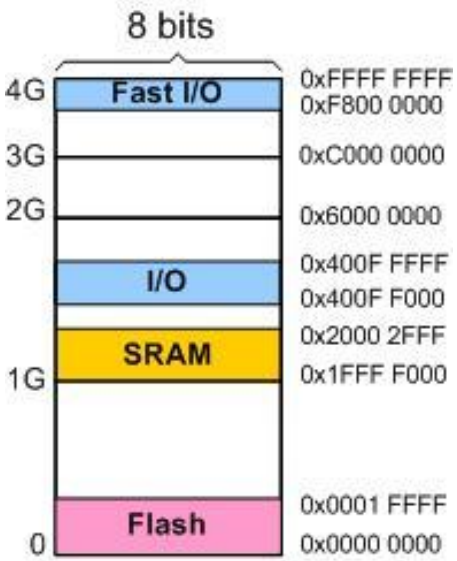
Például:

```
PORTA_PCR3 = 0x100        // PTA3 GPIO módba kerül

PORTA_PCR3 = 0x200        // PTA3 I2C módba kerül
```

Megjegyzés: A hivatkozást így is írhatjuk: PORTA->PCR[3]

A GPIO portok elérése



Az **MKL25Z128VLK4** mikrovezérlő **GPIO** portjai általános perifériaként az APB buszon keresztül a 0x400F F000 – 0x400F F13F címtartományban érhetők el. Ez a hozzáférés azonban nem hatékony, az elérés általában 2 órajelciklust igényel).

A gyorsabb (egyciklusú) elérés az **ARM Cortex-M0+** mikrovezérlők speciális IOPORT vezérlőjén keresztül történik, amit a Freescale Fast GPIO-nak (**FGPIO**) nevezett el. Ennek elérési címtartománya: 0xF80F F000 – 0xF80F F13F

Az I/O portok GPIO és FGPIO báziscímei:

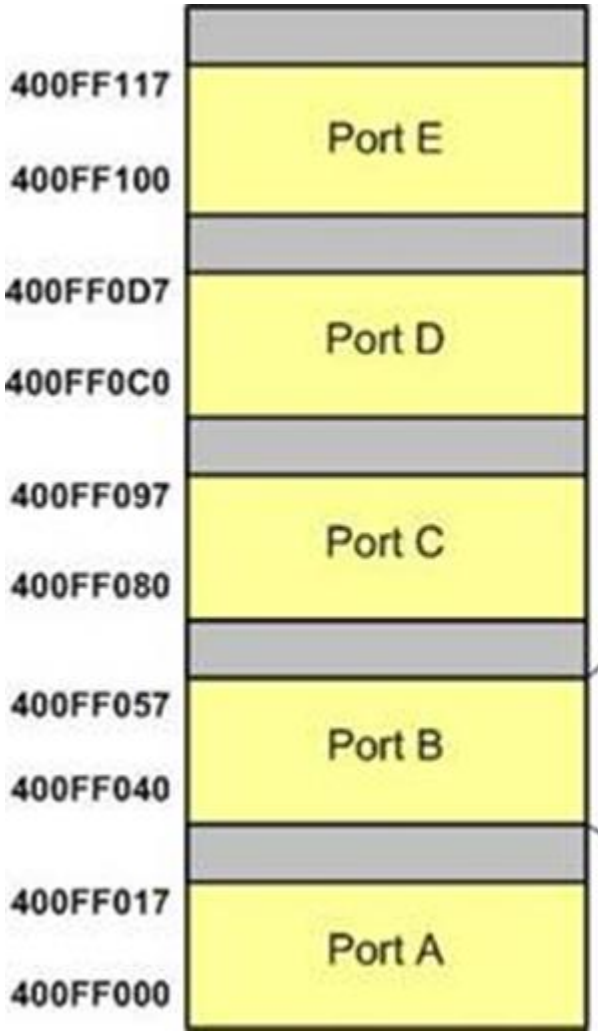
GPIO Port A (APB): 0x400F F000
GPIO Port B (APB): 0x400F F040
GPIO Port C (APB): 0x400F F080
GPIO Port D (APB): 0x400F F0C0
GPIO Port E (APB): 0x400F F100

FGPIO Port A (AHB): 0xF80F F000
FGPIO Port B (AHB): 0xF80F F040
FGPIO Port C (AHB): 0xF80F F080
FGPIO Port D (AHB): 0xF80F F0C0
FGPIO Port E (AHB): 0xF80F F100

GPIO portok regiszterkészlete

PDOR – Kimeneti adatregiszter
PSOR – 1-be állító regiszter
PCOR – 0-ba törlő regiszter
PTOR – bit átbillentő regiszter
PDIR – bemeneti adatregiszter
PDDR – adatáramlási irány (0: bemenet, 1: kimenet)

Ekvivalens művelet:
PDOR ← data
PDOR ← PDOR | data
PDOR ← PDOR & ~data
PDOR ← PDOR ^ data



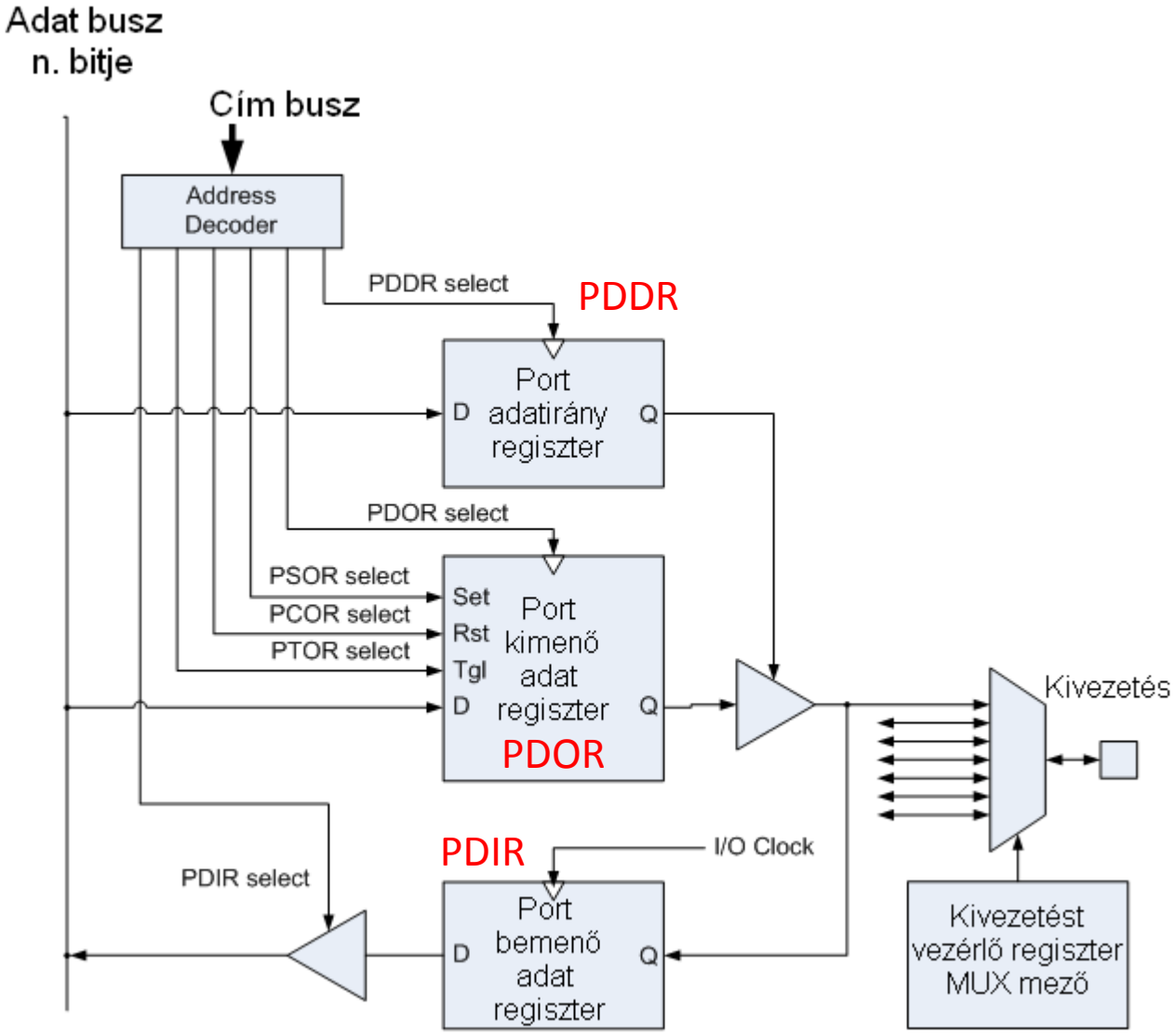
Offset		Absolute Address
0014	**GPIOB_PDDR** Port Data Direction Register	400FF054
0010	**GPIOB_PDIR** Port Data Input Register	400FF050
000C	**GPIOB_PTOR** Port Toggle Output Register	400FF04C
0008	**GPIOB_PCOR** Port Clear Output Register	400FF048
0004	**GPIOB_PSOR** Port Set Output Register	400FF044
0000	**GPIOB_PDOR** Port Data Output Register	400FF040

GPIO kivezetés áramköre

- **Konfiguráció**
  - Adatáramlás iránya
  - Multiplexer
- **Adat**
  - Kimenet (különböző lehetőségek a hozzáférésre)
  - Bemenet

Megjegyzések:

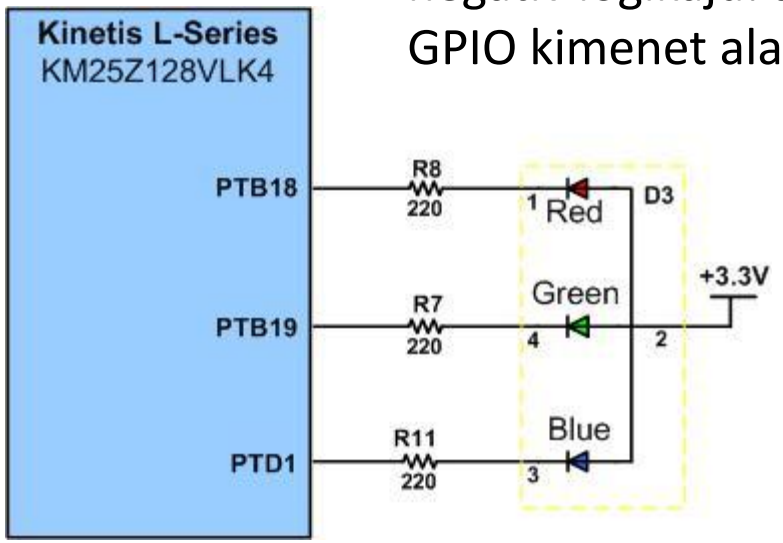
- Az adatirány valójában nem fordítható meg, csak letiltjuk a kimenetet.
- A digitális bemenet minden digitális MUX módban rendelkezésre áll.



Egy GPIO kivezetés áramköri vázlata.

LED vezérlés elve

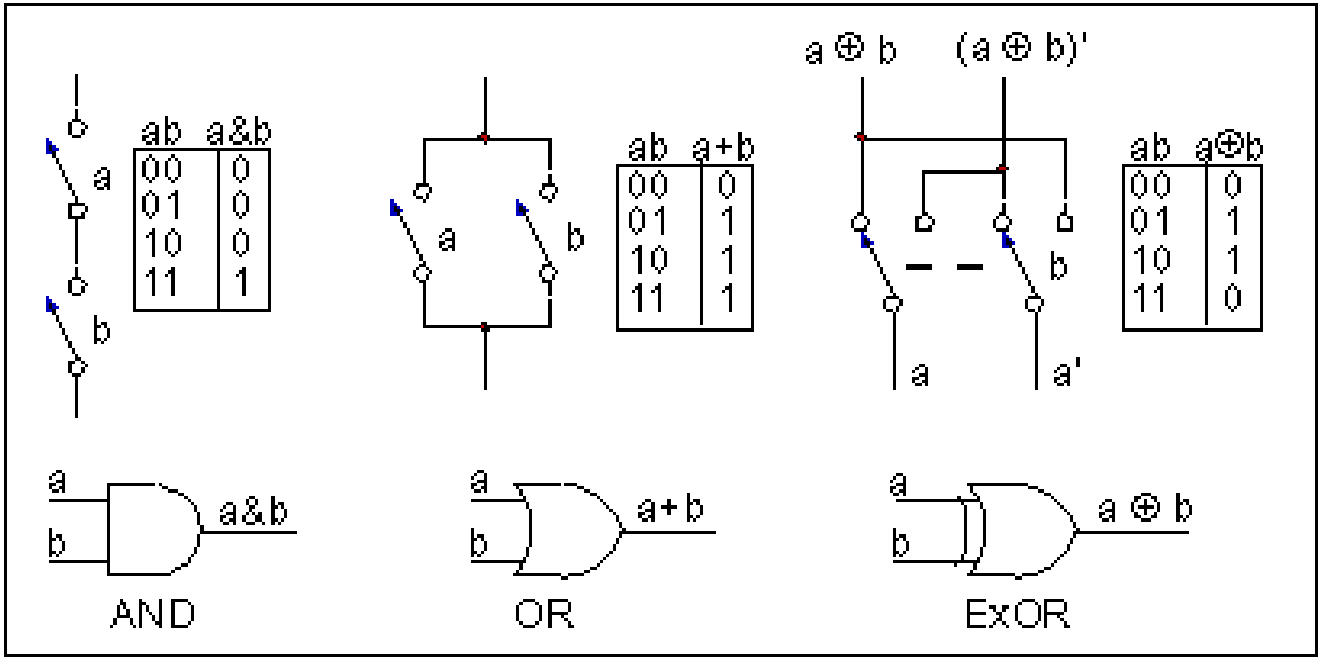
A **FRDM-KL25Z** kártyán a közös anódú RGB LED miatt a vezérlés negatív logikájú: akkor világít a LED, amikor a hozzá kapcsolt GPIO kimenet alacsony szintre húzza a katódot.



Típus: CREE CLV1A-FKB-CJ1M1F1BB7R4S3 **adatlap:** CLV1A-FKB_Rev5.pdf
Max. áram: 25 mA **Hullámhossz:** 619~624 (R), 520~540 (G), 460~480 (B)
Nyitófeszültség @ 20 mA: 2.0 V (R), 3.2 V (G), 3.2 V (B)
Fényerő: 450 – 900 mCd (R), 710 – 1400 mCd (G), 224 – 450 mCd (B)

Emlékeztető: bitműveletek a C nyelvben

A C viszonylag hardverközeli nyelv, ezért fontos szerepe van a bitműveletekenek, amikor az azonos helyiértékű bitek között végzünk logikai műveleteket. Ezeket a műveleteket használhatjuk bitcsoportok törlésére, 1-be állítására, vagy logikai negálására.
A jobbra/balra léptetéssel pedig gyorsan és hatékonyan oszthatunk/szorozhatunk 2 hatványaival.



Bitwise Operators

& (bitwise and)
| (bitwise or)
^ (bitwise xor)
~ (bitwise not)
<< (bitshift left)
>> (bitshift right)

Y = a & b; Y = a | b; Y = a ^ b;

Összetett műveletek

A C nyelv tömörségére jellemző, hogy az **`A = A + 1;`** vagy **`A = A * valami;`** helyett így is írhatjuk: **`A++;`** vagy **`A *= valami;`**

Compound Operators

++	(increment)	a++ ➡	a = a + 1;
--	(decrement)	a-- ➡	a = a - 1;
+=	(compound addition)	a += b; ➡	a = a + b;
-=	(compound subtraction)	a -= b; ➡	a = a - b;
*=	(compound multiplication)	a *= b; ➡	a = a * b;
/=	(compound division)	a /= b; ➡	a = a / b;
&=	(compound bitwise and)	a &= b; ➡	a = a & b;
|=	(compound bitwise or)	a |= b; ➡	a = a | b;

Megjegyzés: az **`a++`** és **`a--`** postfix műveletek, tehát ha értékadás jobboldalán vagy logikai relációban szerepelnek, azok kiértékelésénél az ***a*** változó régi értéke lesz felhasználva, s csak utána történik az inkrementálás/dekrementálás.
Ha az új (megnövelt/csökkentett) értéket akarjuk felhasználni, akkor a művelet prefix megfelelőjét kall használni: **`++a`**, vagy **`--a`**.

Bit vagy bitcsoport 1-be állítása

(BIT6 + BIT0)

P1DIR |= 0x41; // 0x41 = 0100_0001b

Jelentése: P1DIR = P1DIR | 0x41;

Elv: x | 1 = 1

Példa:

```
0 x 1 0   1 1 0 x   kiindulási érték
0 1 0 0   0 0 0 1   bitmaszk
---------------------------------
0 1 1 0   1 1 0 1   eredmény
```

Bit vagy bitcsoport 0-ba törlése

(BIT6 + BIT0)

P1DIR &= ~0x41; // ~0x41 = 1011_1110b

Jelentése: P1DIR = P1DIR & 0xDE;

Elv: x & 0 = 0

Példa:

```
0 x 1 0   1 1 0 x   kiindulási érték
1 0 1 1   1 1 1 0   bitmaszk
---------------------------------
0 0 1 0   1 1 0 0   eredmény
```

Bit vagy bitcsoport átbillentése

(BIT6 + BIT0)

P1DIR ^= 0x41; // 0x41 = 0100_0001b

Jelentése: P1DIR = P1DIR ^ 0x41;

Elv: x ^ 1 = ~x x ^ 0 = x

Példa:

```
0 1 1 0   1 1 0 0   kiindulási érték
0 1 0 0   0 0 0 1   bitmaszk
---------------------------------
0 0 1 0   1 1 0 1   eredmény
```

Mintapélda: Program2_1

A kártyára épített zöld LED (PTB19) villogtatása. A használt regiszterek címét mi definiáljuk.
Forrás: Mazidi et al., **Freescale ARM Cortex-M Embedded Programming**

```
#define SIM_SCGC5    (*((volatile unsigned int*)0x40048038))
#define PORTB_PCR19  (*((volatile unsigned int*)0x4004A04C))
#define GPIOB_PDDR   (*((volatile unsigned int*)0x400FF054))
#define GPIOB_PDOR   (*((volatile unsigned int*)0x400FF040))

int main (void) {
    void delayMs(int n);                // Előre deklaráljuk a delayMs() függvényt
    SIM_SCGC5 |= 0x400;                 // Port B órajelének engedélyezése (bit10=1)
    PORTB_PCR19 = 0x100;                // PTB19 pin legyen GPIO (MUX = 1, Alt1 mode)
    GPIOB_PDDR |= 0x80000;              // PTB19 legyen kimenet (bit 19 = 1)
    while (1) {                         // Végtelen ciklus...
        GPIOB_PDOR &= ~0x80000;         // Zöld LED felkapcsolása (bit19 = 0)
        delayMs(500);                   // 500 ms várakozás
        GPIOB_PDOR |= 0x80000;          // Zöld LED lekapcsolása   (bit19 = 1)
        delayMs(500);                   // 500 ms várakozás
    }
}
void delayMs(int n) {                   // Késleltető függvény
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++); // Alapértelmezett órajelhez (20.97152 MHz)
}
```

Mintapélda: Program2_4

A kártyára épített zöld LED (PTB19) villogtatása. A regiszterek eléréséhez használt struktúrákat a Keil **MKL25Z4.H** nevű fejléc állománya definiálja számunkra, amit be kell csatolnunk.
A program forrása: Mazidi et al., **Freescale ARM Cortex-M Embedded Programming**

Megjegyzések: 1. **PTB** csupán egy alias név a **GPIOB**-hez.
2. A Keil **MKL25Z4.h** fejléc állománya többféle szintaxis használatát is támogatja, ezért például `PORTB_PCR19` helyett `PORTB->PCR[19]`, `SIM_SCG5` helyett pedig `SIM->SCG5` is írható!

```
#include <MKL25Z4.h>
int main (void) {
    void delayMs(int n);                // Előre deklaráljuk a delayMs() függvényt
    SIM->SCGC5 |= 0x400;                // Port B órajelének engedélyezése (bit10=1)
    PORTB->PCR[19] = 0x100;             // PTB19 pin legyen GPIO (MUX = 1, Alt1 mode)
    PTB->PDDR |= 0x80000;               // PTB19 legyen kimenet (bit 19 = 1)
    while (1) {                         // Végtelen ciklus...
        PTB->PDOR &= ~0x80000;          // Zöld LED felkapcsolása (bit19 = 0)
        delayMs(500);                   // 500 ms várakozás
        PTB->PDOR |= 0x80000;           // Zöld LED lekapcsolása   (bit19 = 1)
        delayMs(500);                   // 500 ms várakozás
    }
}
void delayMs(int n) {                   // Késleltető függvény
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++);     // Alapértelmezett órajelhez (20.97152 MHz)
}
```

MKL25Z4.h (részletek)

Nézzünk bele az **MKL25Z4.h** állományba! Többek között ezeket találjuk benne:

```
/* GPIO - Register Layout Typedef */
typedef struct {
  __IO uint32_t PDOR;     // Port Data Output Register,    offset: 0x0
  __O  uint32_t PSOR;     // Port Set Output Register,     offset: 0x4
  __O  uint32_t PCOR;     // Port Clear Output Register,   offset: 0x8
  __O  uint32_t PTOR;     // Port Toggle Output Register,  offset: 0xC
  __I  uint32_t PDIR;     // Port Data Input Register,     offset: 0x10
  __IO uint32_t PDDR;     // Port Data Direction Register, offset: 0x14
} GPIO_Type

#define GPIOB_BASE        (0x400FF040u)
#define GPIOB             ((GPIO_Type *)GPIOB_BASE)
#define PTB               GPIOB

/** PORT - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCR[32];  // Pin Control Register n, offset: 0x0, step: 0x4
  __O  uint32_t GPCLR;    // Global Pin Control Low Register, offset: 0x80
  __O  uint32_t GPCHR;    // Global Pin Control High Register, offset: 0x84
  uint8_t RESERVED_0[24];
  __IO uint32_t ISFR;     // Interrupt Status Flag Register, offset: 0xA0
} PORT_Type

#define PORTB_BASE        (0x4004A000u)
#define PORTB             ((PORT_Type *)PORTB_BASE)
```

Mintapélda: Program2_5

A kártyára épített zöld LED (PTB19) villogtatása. A kimenet vezérléséhez most használjuk a PSOR, PCOR, PTOR bitmanipulációs regisztereket!

A program forrása: Mazidi et al., **Freescale ARM Cortex-M Embedded Programming**

```
#include <MKL25Z4.h>
int main (void) {
    void delayMs(int n);                // Előre deklaráljuk a delayMs() függvényt
    SIM->SCGC5 |= 0x400;                // Port B órajelének engedélyezése (bit10=1)
    PORTB->PCR[19] = 0x100;             // PTB19 pin legyen GPIO (MUX = 1, Alt1 mode)
    PTB->PDDR |= 0x80000;               // PTB19 legyen kimenet (bit 19 = 1)
    while (1) {                         // Végtelen ciklus...
        PTB->PCOR = 0x80000;            // Zöld LED felkapcsolása (bit19 = 0)
        delayMs(500);                   // 500 ms várakozás
        PTB->PSOR = 0x80000;            // Zöld LED lekapcsolása   (bit19 = 1)
        delayMs(500);                   // 500 ms várakozás
    }
}
void delayMs(int n) {                   // Késleltető függvény
    int i, j;
    for(i = 0 ; i < n; i++)
        for (j = 0; j < 3500; j++); // Alapértelmezett órajelhez (20.97152 MHz)
}
```

Egyszerűbben is lehet:

```
while (1) {PTB->PTOR = 0x80000; delayMs(500); }
```

Mintapélda: Program2_7

Az RGB LED alapszín kombinációinak körbeléptetése.

A program forrása: Mazidi et al., **Freescale ARM Cortex-M Embedded Programming**

```
#include <MKL25Z4.h>

int main (void) {
  void delayMs(int n);
  int counter = 0;
  SIM->SCGC5 |= 0x1400;          // Port B és D engedélyezése (bit 10 és bit 12)
  PORTB->PCR[18] = 0x100;        // PTB18 legyen GPIO (MUX = 1)
  PORTB->PCR[19] = 0x100;        // PTB19 legyen GPIO (MUX = 1)
  PORTD->PCR[1]  = 0x100;        // PTD1 legyen GPIO  (MUX = 1)
  PTB->PDDR |= 0xC0000;          // PTB18 és PTB19 legyen kimenet (bit18 és 19=1)
  PTD->PDDR |= 0x02;             // PTD1 legyen kimenet (bit1 = 1)
  while (1) {
    if (counter & 1) PTB->PCOR = 0x40000; // Piros LED: BE (bit18 = 0)
      else PTB->PSOR = 0x40000; // Piros LED: KI           (bit18 = 1)
    if (counter & 2) PTB->PCOR = 0x80000; // Zöld LED: BE  (bit19 = 0)
      else PTB->PSOR = 0x80000; // Zöld LED: KI            (bit19 = 1)
    if (counter & 4) PTD->PCOR = 0x02;    // Kék LED: BE   (bit1  = 0)
      else PTD->PSOR = 0x02;    // Kék LED: KI             (bit1  = 0)
    counter++;
    delayMs(500);
  }
}
```

A **delayMs()** függvény itt nem részletezett definíciója megegyezik a korábbiakkal!

Kódolási stílusok

```
SIM->SCGC5 |= 0x1400; // Port B és D engedélyezése (bit 10 és bit 12)
```

Bit 10 és bit 12 beállítása: **0000 0000 0000 0000 0001 0100 0000 0000** binárisan vagy **0x0000 1400** hexadecimálisan

A bitek kézi számolgatása helyett dolgoztassuk meg a fordítót!

```
m = (1UL << 12) | (1UL << 10);
```

(UL az unsigned long típusjele!)

```
SIM->SCGC5 |= m;
```

Nem rossz, de próbáljuk még olvashatóbbá tenni a kódot!

1. Definiáljunk konstansokat a portvezérlő bitekhez!

```
#define b_port (10)
#define d_port (12)
```

2. Definiáljunk makrót a megadott helyiértékű bit kijelöléséhez!

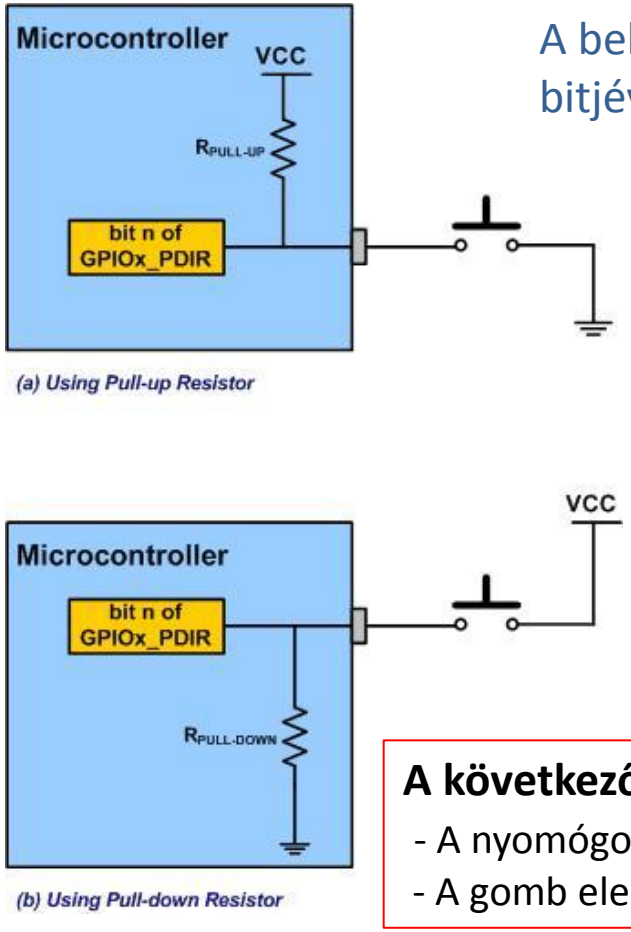
```
#define MASK(x) (1 << x)
```

3. A portok engedélyezése:

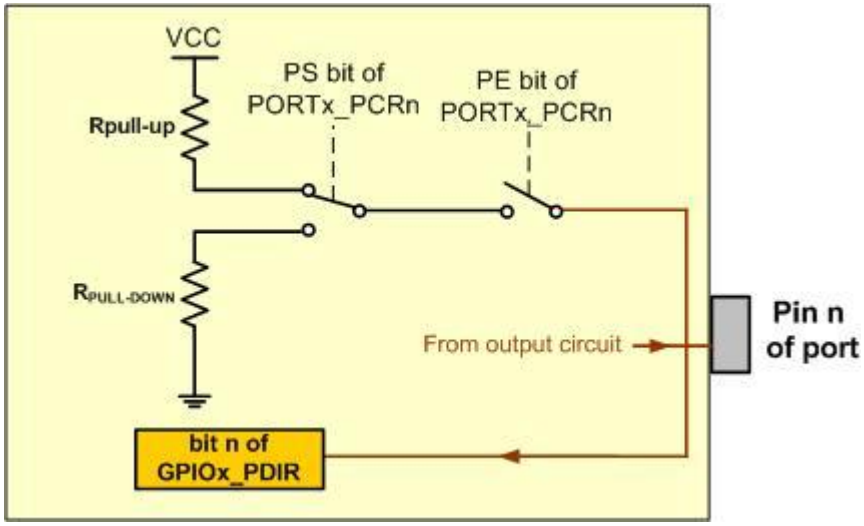
```
SIM->SCGC5 |= MASK(b_port) | MASK(d_port);
```

Nyomógomb állapotának beolvasása

A **FRDM-KL25Z** kártya nem tartalmaz felhasználói nyomógombot, viszont a bővítő csatlakozók felhasználásával tudunk külső nyomógombot csatlakoztatni. A csatlakoztatás módjától függően használhatjuk a belső felhúzást vagy lehúzás lehetőségét.



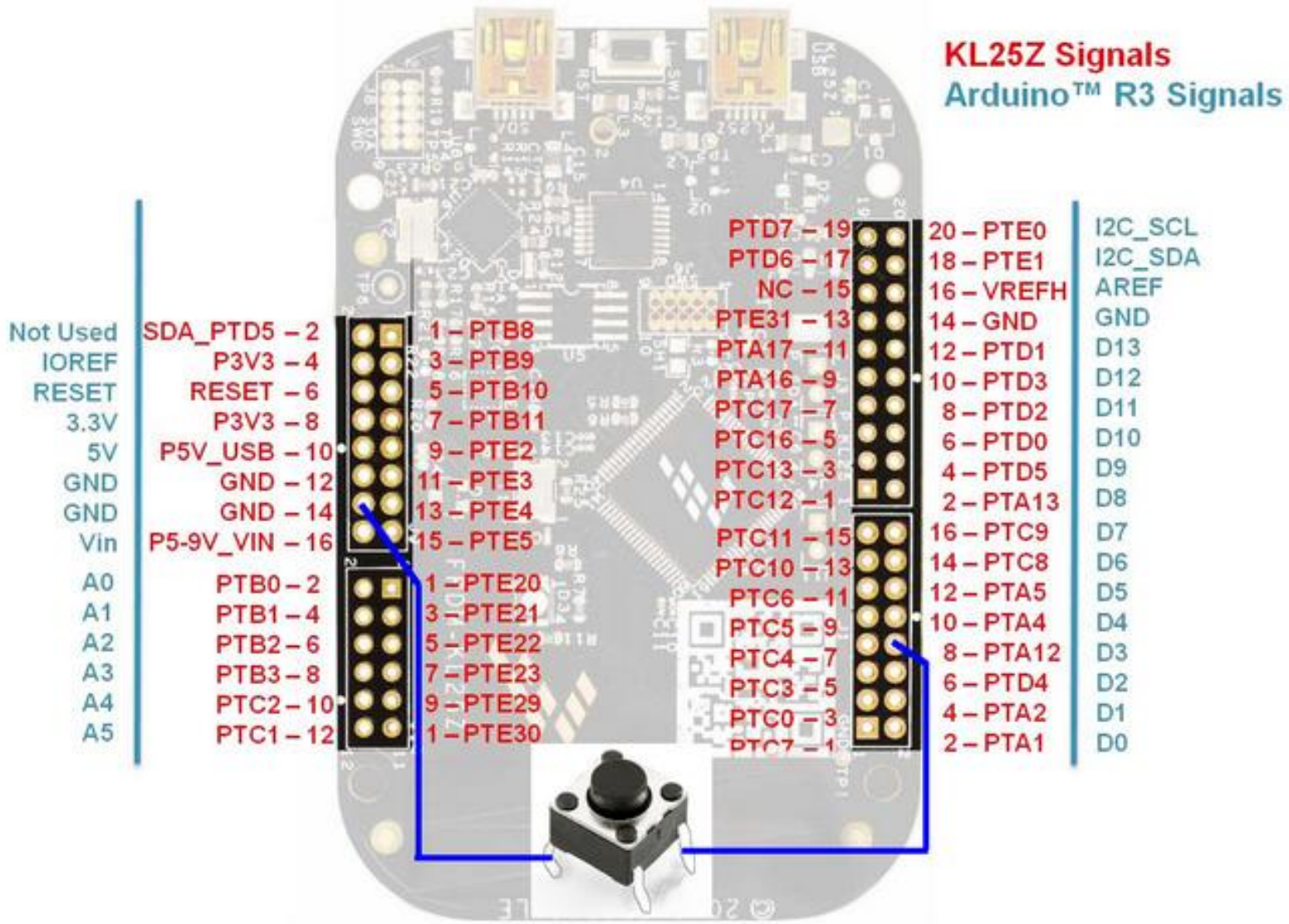
A belső fel- vagy lehúzást a megfelelő **PORT*x*_PCR*n*** regiszter **PE** bitjével tudjuk engedélyezni, és a **PS** bitjével tudjuk kiválasztani.



A következő mintaprogramban mi az *a.)* esetetet fogjuk használni:

- A nyomógomb lenyomáskor földre húzza a bemenetet.
- A gomb elengedésekor a belső felhúzás magas szintre húzza a bemenetet.

A nyomógomb bekötési vázlata



Mintapélda: Program2_8

LED vezérlése nyomógombbal. Ha a **PTA12** bemenet és a GND közé kötött nyomógombot lenyomjuk, a zöld LED világítani kezd. Ha a nyomógombot felengedjük, a LED kialszik.
A **PTA12** bemenet felhúzásához bekapcsoljuk a belső felhúzást.
A program forrása: Mazidi et al., **Freescale ARM Cortex-M Embedded Programming**

Az eredeti programban szereplő **PTA1** bemenet helyett mi **PTA12**-re kötjük a nyomógombot!

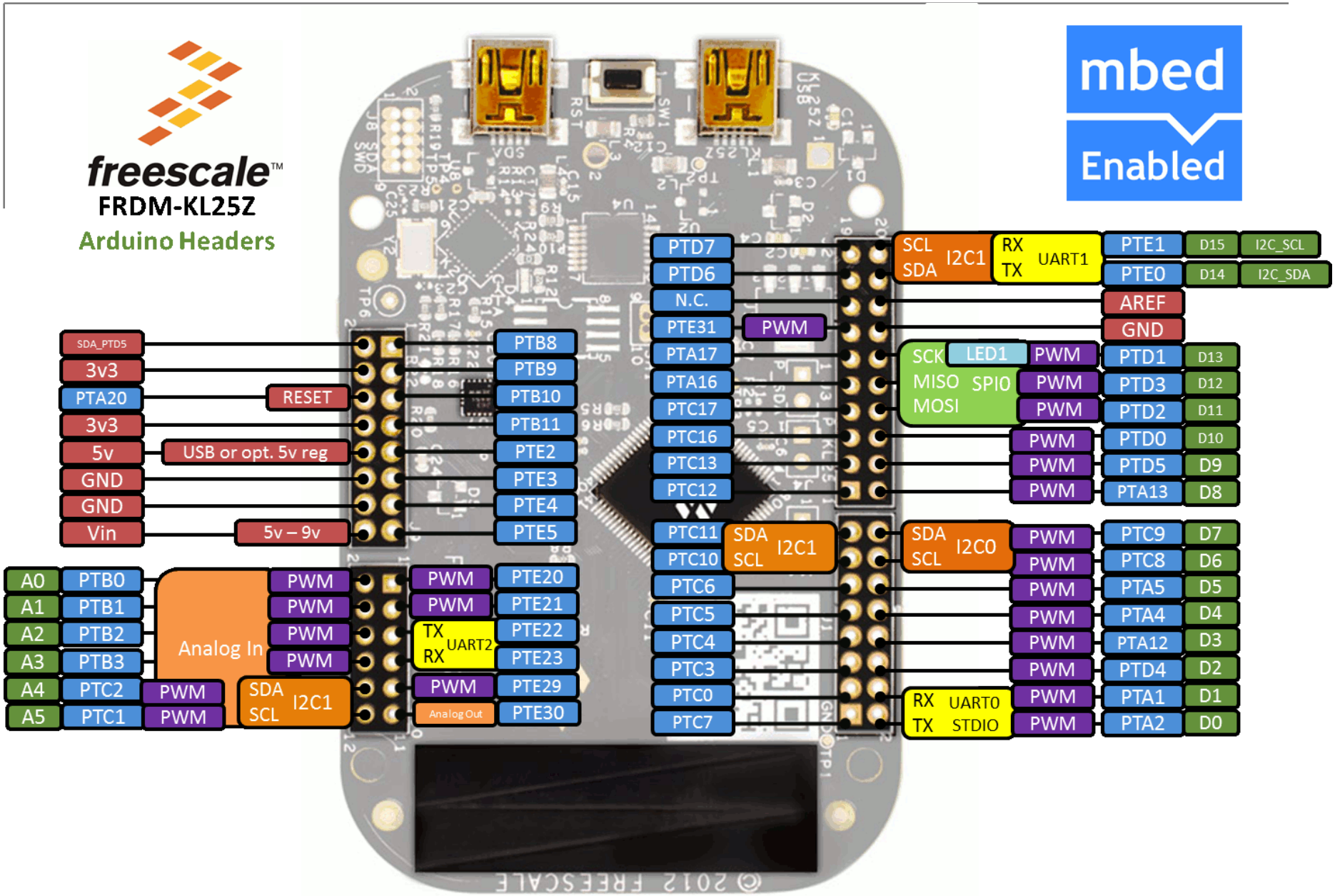
Összevonhatók!

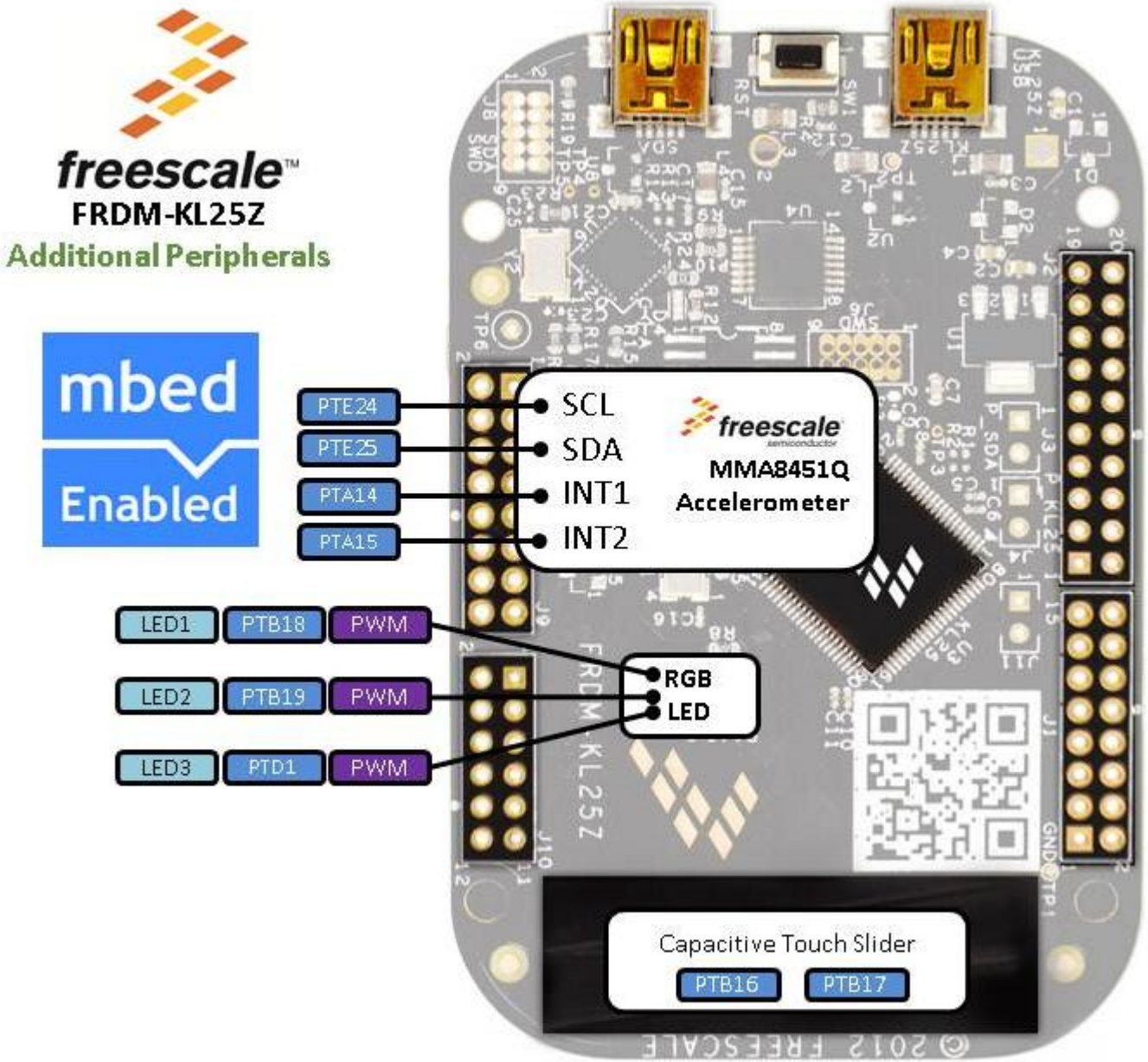
```
#include <MKL25Z4.h>

int main (void) {
    SIM->SCGC5 |= 0x400;          // B port engedélyezése
    PORTB->PCR[19] = 0x100;       // PTB19 legyen GPIO (MUX = 1)
    PTB->PDDR |= 0x80000;         // PTB19 legyen kimenet (Zöld LED vezérlés)

    SIM->SCGC5 |= 0x200;          // A port engedélyezése
    PORTA->PCR[12] = 0x103;       // PTA12: GPIO, felhúzással (PE=1,PS=1, MUX=1)
    PTA->PDDR &= ~0x1000;         // PTA12 legyen bemenet (PDDR bit 12 = 0)

    while (1) {
        if (PTA->PDIR & 0x1000)   // Gomblenyomás figyelése
            PTB->PSOR = 0x80000;  // Felengedve: LED KI
        else
            PTB->PCOR = 0x80000;  // Lenyomva: LED be
    }
}
```


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