STM32 mikrovezérlők programozása STM32CubeIDE környezetben

1. STM32CubeIDE alapok, I/O műveletek

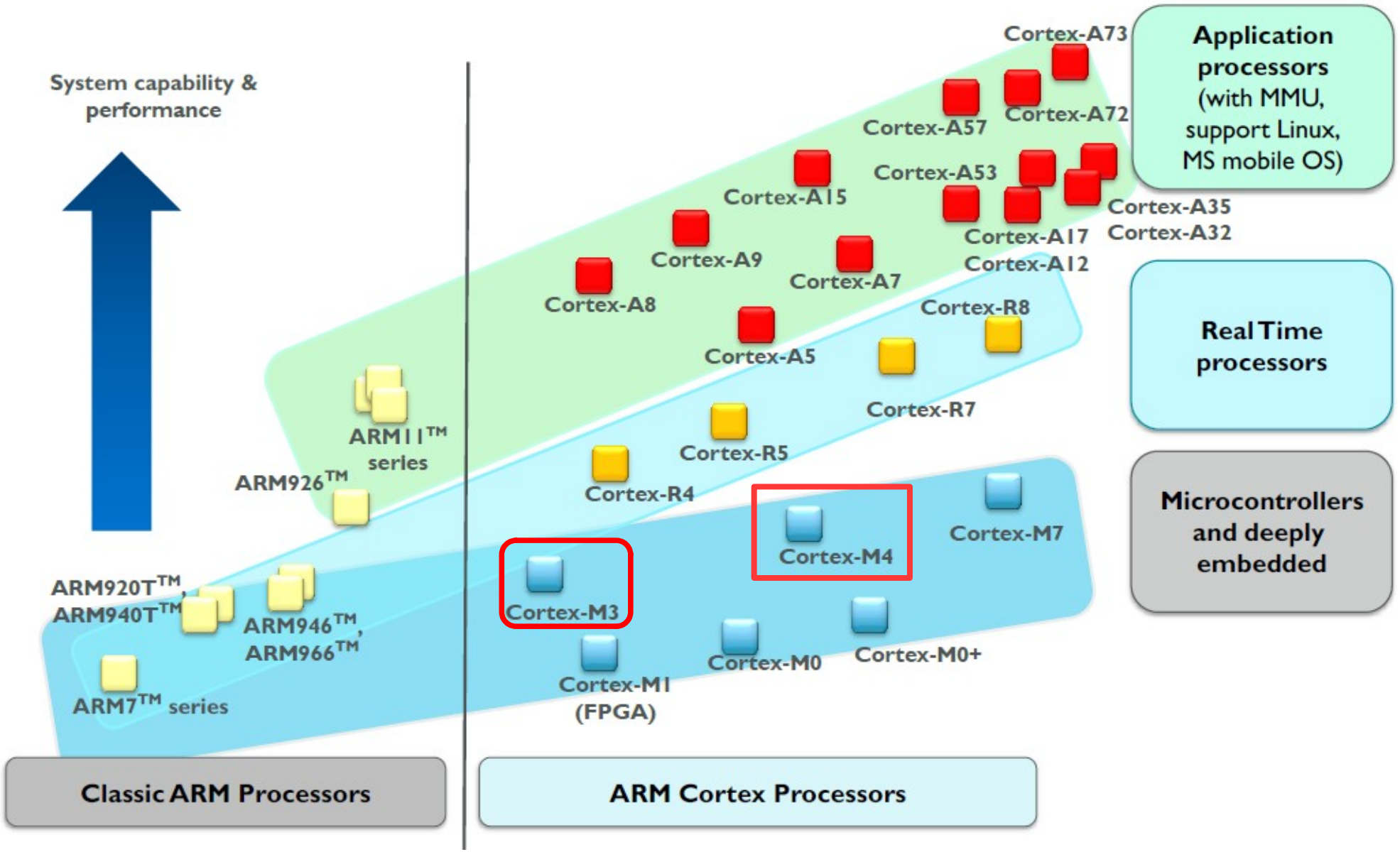
Felhasznált és ajánlott irodalom

- Joseph Yiu: **Cortex-M for Beginners**
- Joseph Yiu: **The Definitive Guide To The ARM Cortex-M3 and Cortex-M4**
- Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi: STM32 Arm Programming for Embedded Systems
- STMicroelectronics: **STM32CubeIDE Installation Guide**
- ST Microelectronics: **STM32CubeIDE quick start guide**
- EMCU: **First embedded program for STM32 mcu using STM32CubeIDE**

Adatlapok:

- **STM32F103C8 adatlap és termékinfo**
- **STM32F103 Family Reference Manual**
- **STM32F446RE adatlap és termékinfo**
- **STM32F446** Family Reference Manual

ARM processzorok

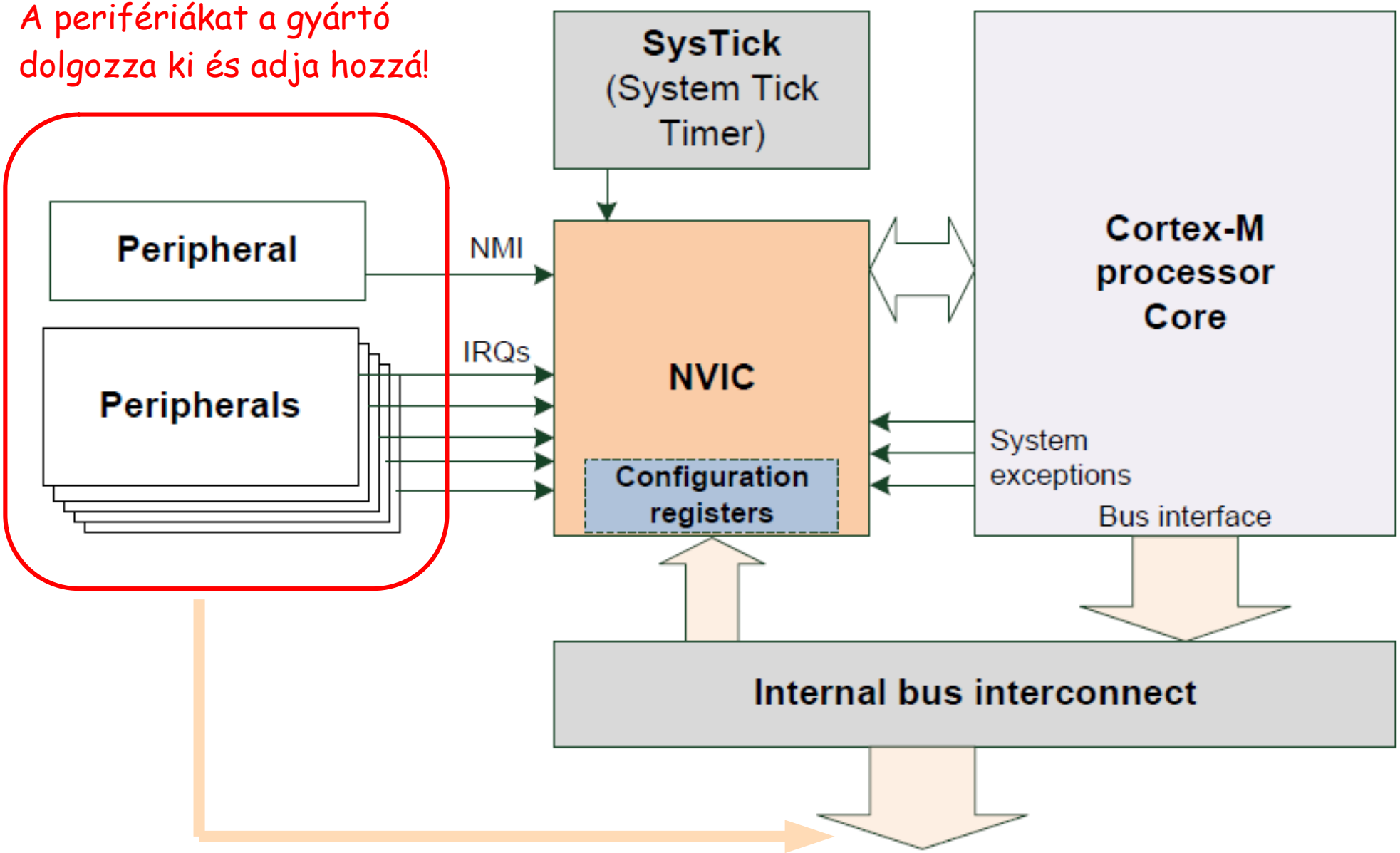


ARM Cortex-M mikrovezérlők

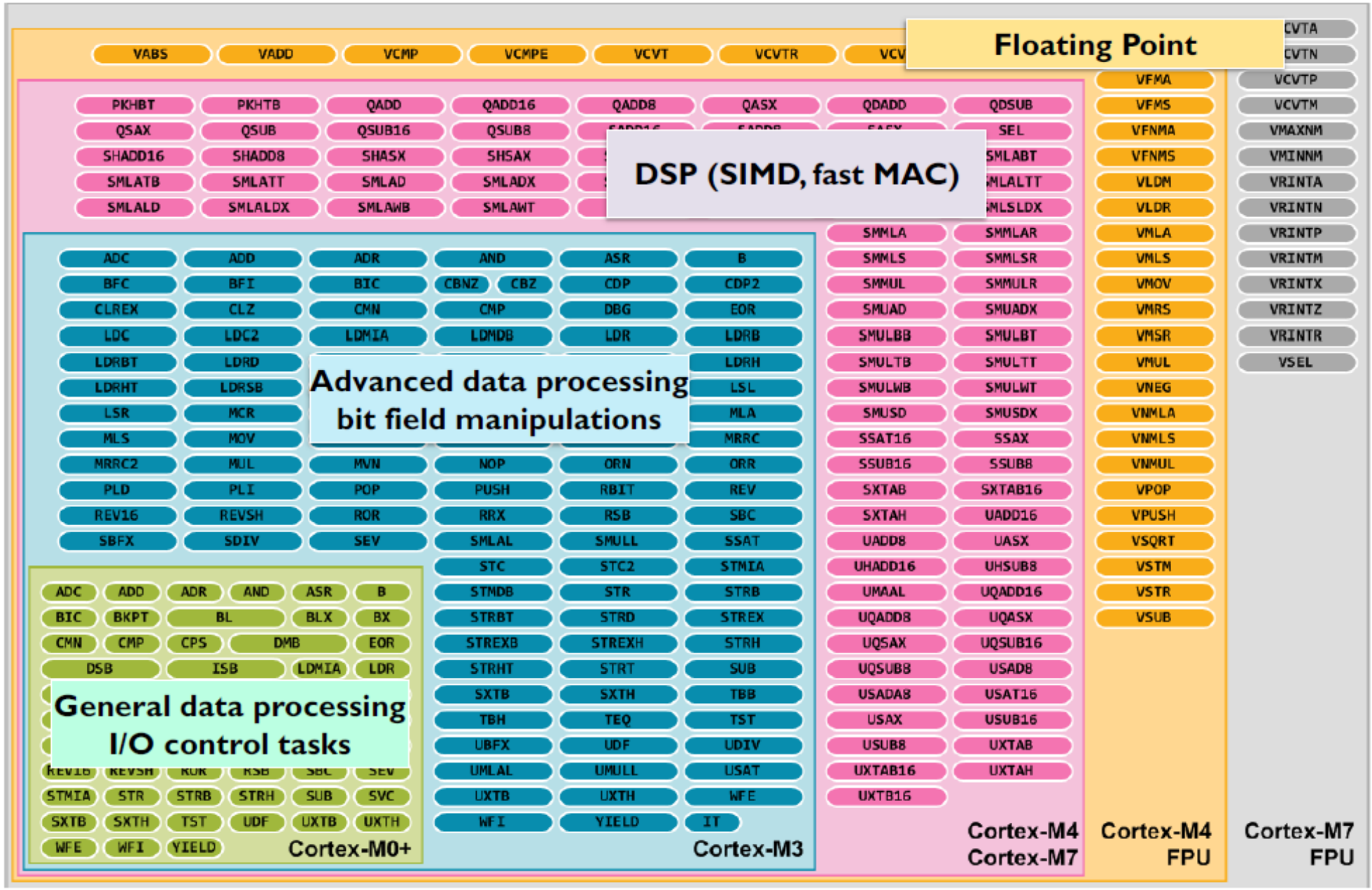
Cortex-M0	Egyszerű felépítésű (~12K kapu), kisfogyasztású processzor beágyazott alkalmazásokhoz
Cortex-M0+	Továbbfejlesztett Cortex-M0 processzor, ultra-kisfogyasztású, egyciklusú I/O, áthelyezhető vektortábla, stb.
Cortex-M1	FPGA-hoz kifejlesztett „szoftveres processzor” (Verilog). Utasításkészlete megegyezik a Cortex-M0-val
Cortex-M3	Hatékony mikrovezérlő, gazdag utasításkészlettel a komplex feladatokhoz. pl. Multiply-Accumulate (MAC) utasítások, hardveres osztás
Cortex-M4	Ugyanaz, mint a Cortex-M3, kibővítve további és hatékonyabb DSP utasításokkal. Opcionálisan egyszeres pontosságú lebegőpontos műveletekkel (Cortex-M4F)
Cortex-M7	Nagyteljesítményű mikrovezérlő CPU intenzív alkalmazásokhoz, duplapontosságú lebegőpontos utasítás, hatékonyabb memóriakezelés

ARM Cortex-M közös jellemzők

A perifériákat a gyártó dolgozza ki és adja hozzá!



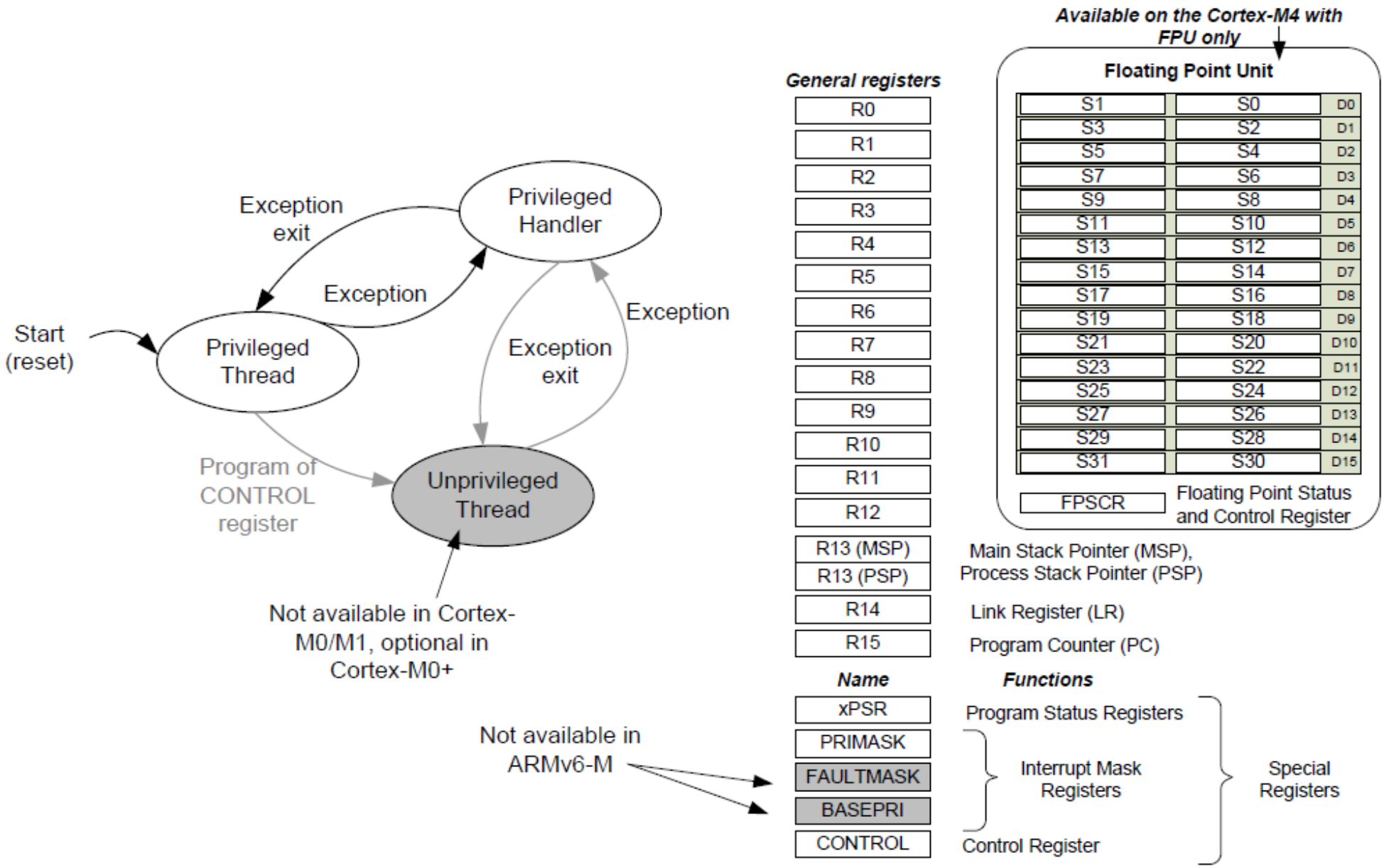
ARM Cortex-M utasításkészlet



ARM Cortex-M utasításkészlet

- A **Cortex-M0/M0+/M1** processzorok csak az alapvető I/O vezérlő műveletek és az általános adatfeldolgozási feladatokhoz alkalmas utasításkészlettel rendelkeznek (56 utasítás, melyek többnyire 16 bitesek – két utasítás egy szóban). Ez a készlet kevés logikai kapuval megvalósítható (~12k gate), de pl. a magasabb sorszámú regiszterek (R8-R12) elérése korlátozott.
- A **Cortex-M3 és M4** processzorok utasításkészlete jóval gazdagabb, számos 32 bites utasítást is nyújt, amelyekkel a felső regiszterek is használhatók. Emellett olyan utasításokkal is rendelkezik, mint:
  - Táblázatos elágaztató utasítások és feltételes végrehajtás
  - hardveres osztó utasítás
  - Szorzás és összegzés (MAC) utasítások
  - Különféle bitmanipulációs műveletek

Programozói modell



Kivételek és megszakítások

Exception Type	ARMv6-M	ARMv7-M	Vector Table	Vector address (initial)
255		Device Specific Interrupts 240/ 8 pri	Interrupt#239 vector [1]	0x000003FC
47	Device Specific Interrupts 32/ 4 pri		Interrupt#31 vector [1]	0x000000BC
17			Interrupt#1 vector [1]	0x00000044
16			Interrupt#0 vector [1]	0x00000040
15	SysTick	SysTick	SysTick vector [1]	0x0000003C
14	PendSV	PendSV	PendSV vector [1]	0x00000038
13	Not used	Not used	Not used	0x00000034
12		Debug Monitor	Debug Monitor vector [1]	0x00000030
11	SVC	SVC	SVC vector [1]	0x0000002C
10	Not used	Not used	Not used	0x00000028
9			Not used	0x00000024
8			Not used	0x00000020
7			SecureFault (ARMv8-M Mainline) [1]	0x0000001C
6		Usage Fault	Usage Fault vector [1]	0x00000018
5		Bus Fault	Bus Fault vector [1]	0x00000014
4		MemManage (fault)	MemManage vector [1]	0x00000010
3	HardFault	HardFault	HardFault vector [1]	0x0000000C
2	NMI	NMI	NMI vector [1]	0x00000008
1			Reset vector [1]	0x00000004
0			MSP initial value	0x00000000
	Cortex-M0	Cortex-M3		

Interrupt késedelem

- **Interrupt latency:** A programmegszakítás érvényesülésétől a megszakítást kiszolgáló eljárás első utasításáig eltelt órajelciklusok száma

	Interrupt latency (number of clock cycles)
Cortex-M0	16
Cortex-M0+	15
Cortex-M3	12
Cortex-M4	12
Cortex-M7	Typically 12, worst case 14

Mit nyújtanak az STM32 mikrovezérlők?

Fejlett perifériakészlet

Kedvező árfekvés

Skálázható termék portfolió

Program-könyvtárak

Fejlesztőeszközök gazdag választéka

Ultra-kis-fogyasztás

Fejlett központi egység

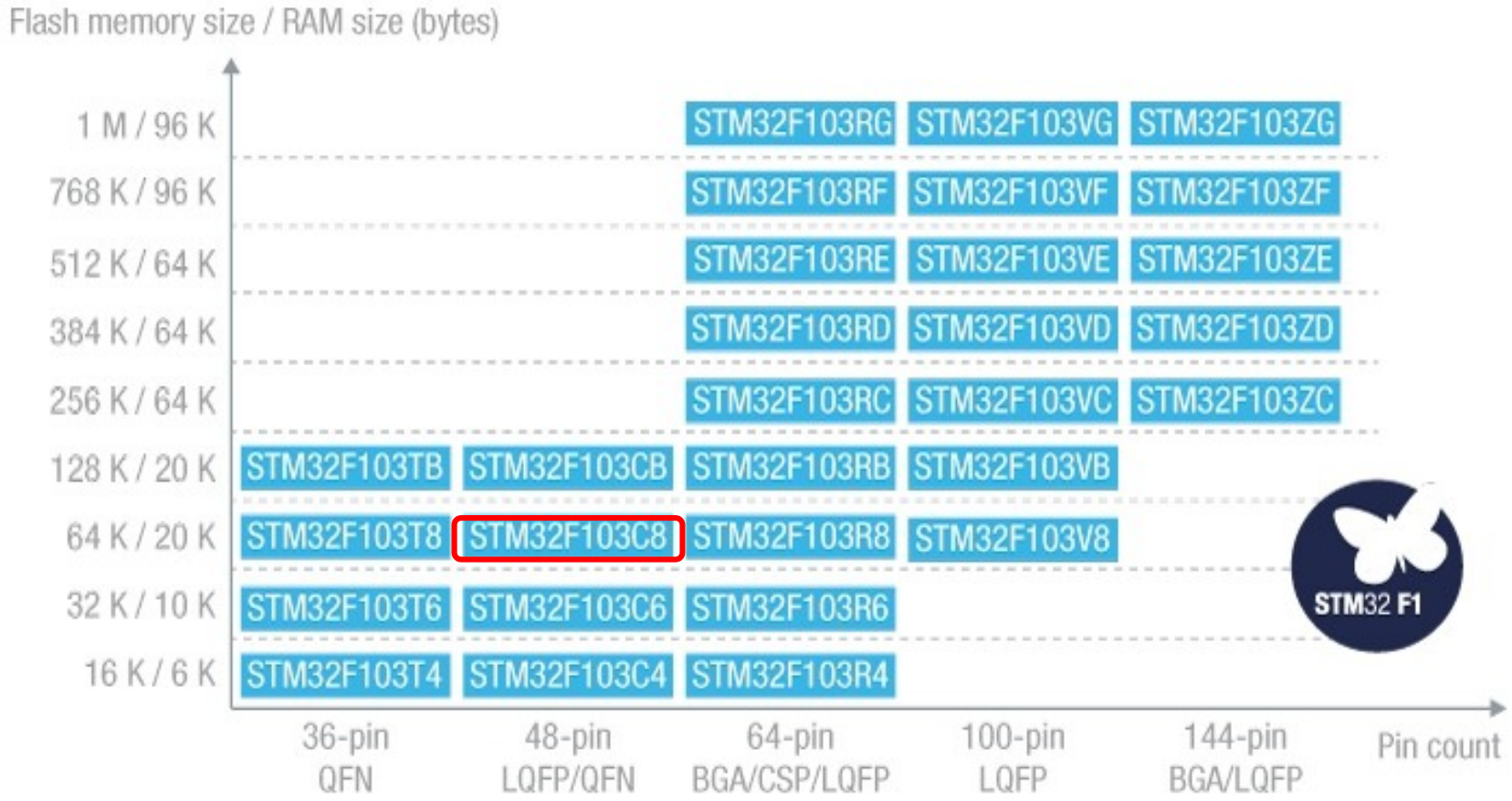
life.augmented

Mi az ST Cortex-M portfólió előnye?

- Elfelejthetjük a hagyományos 8/16/32 bites kategóriákat
- Minden célra találunk alkalmas architektúrát
- Kis fogyasztásra és könnyű felhasználásra optimalizált MCU-k

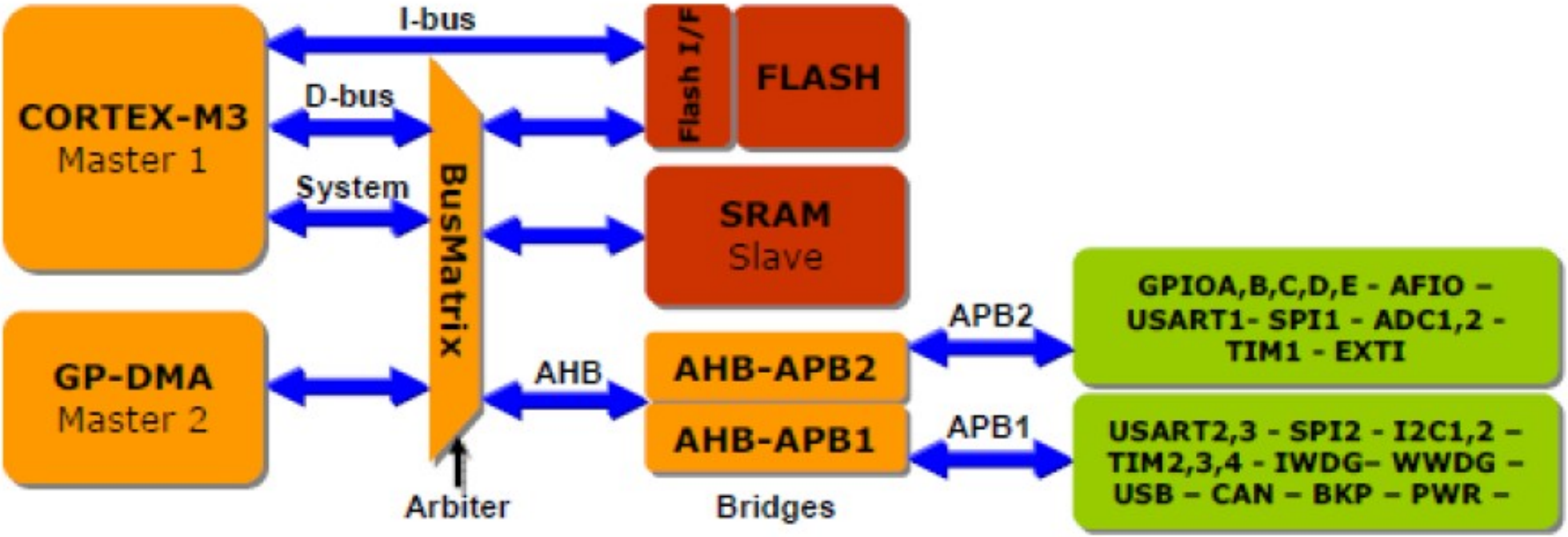
STM32F103 portfolio

- Az **ST Microelectronics** 32-bites mikrovezérlői a kisfogyasztású Cortex-M0 típusoktól a nagyteljesítményű Cortex-M7 típusig minden igénynek megfelelő skálázható termékcsaládot kínálnak
- **STM32F103C8 –** az **F103** (Cortex-M3) család kis lábszámú tagja

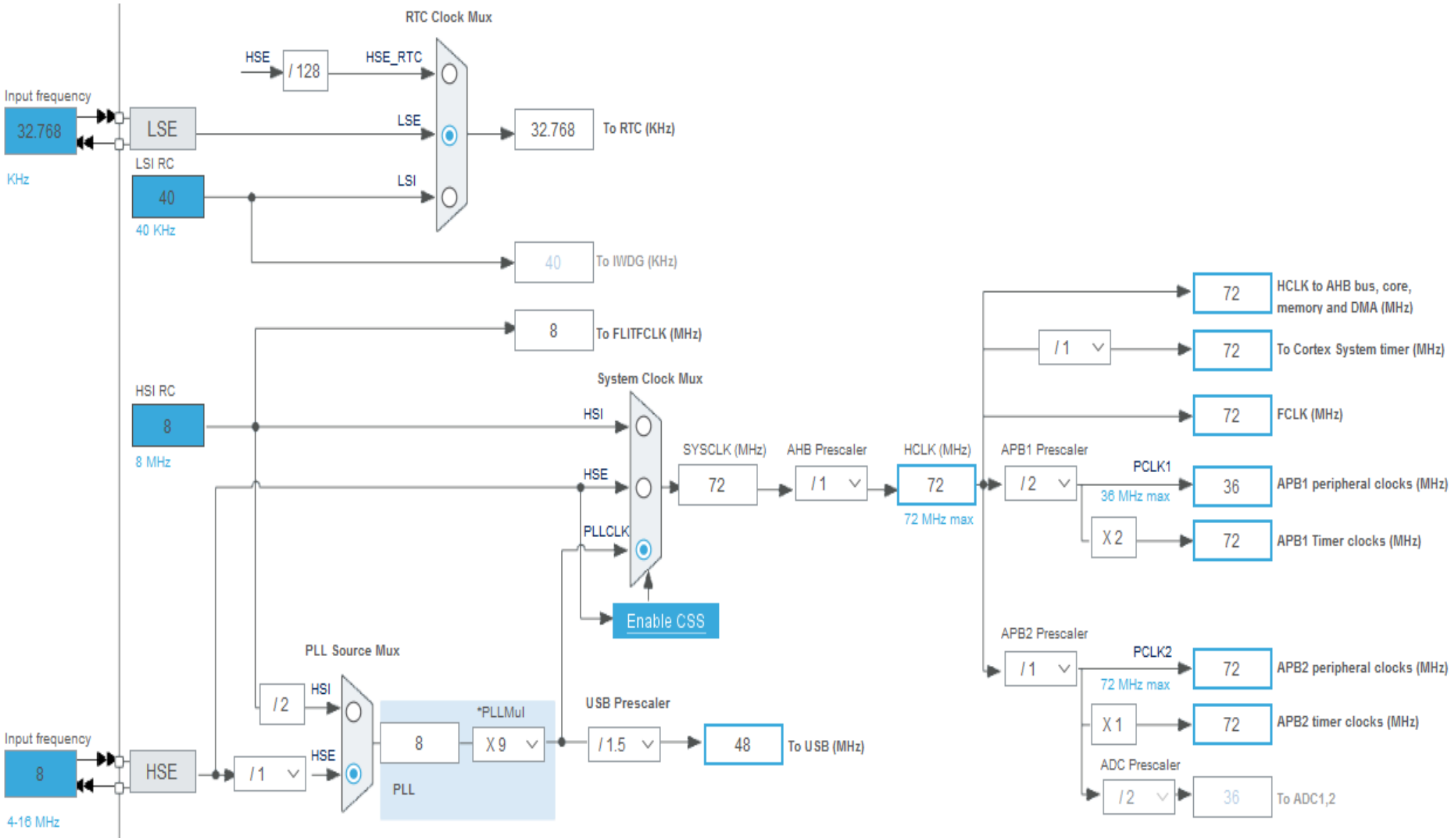


STM32F103 rendszerfelépítés

- Az adatáramlás a busz mátrixon keresztül történik
  - 4 master: I-bus, D-bus, System bus, GP-DMA bus
  - 3 slave: Flash memória, SRAM memória, **AHB** híd
- A perifériák az **APB1,** illetve **APB2** periféria buszokra vannak elosztva

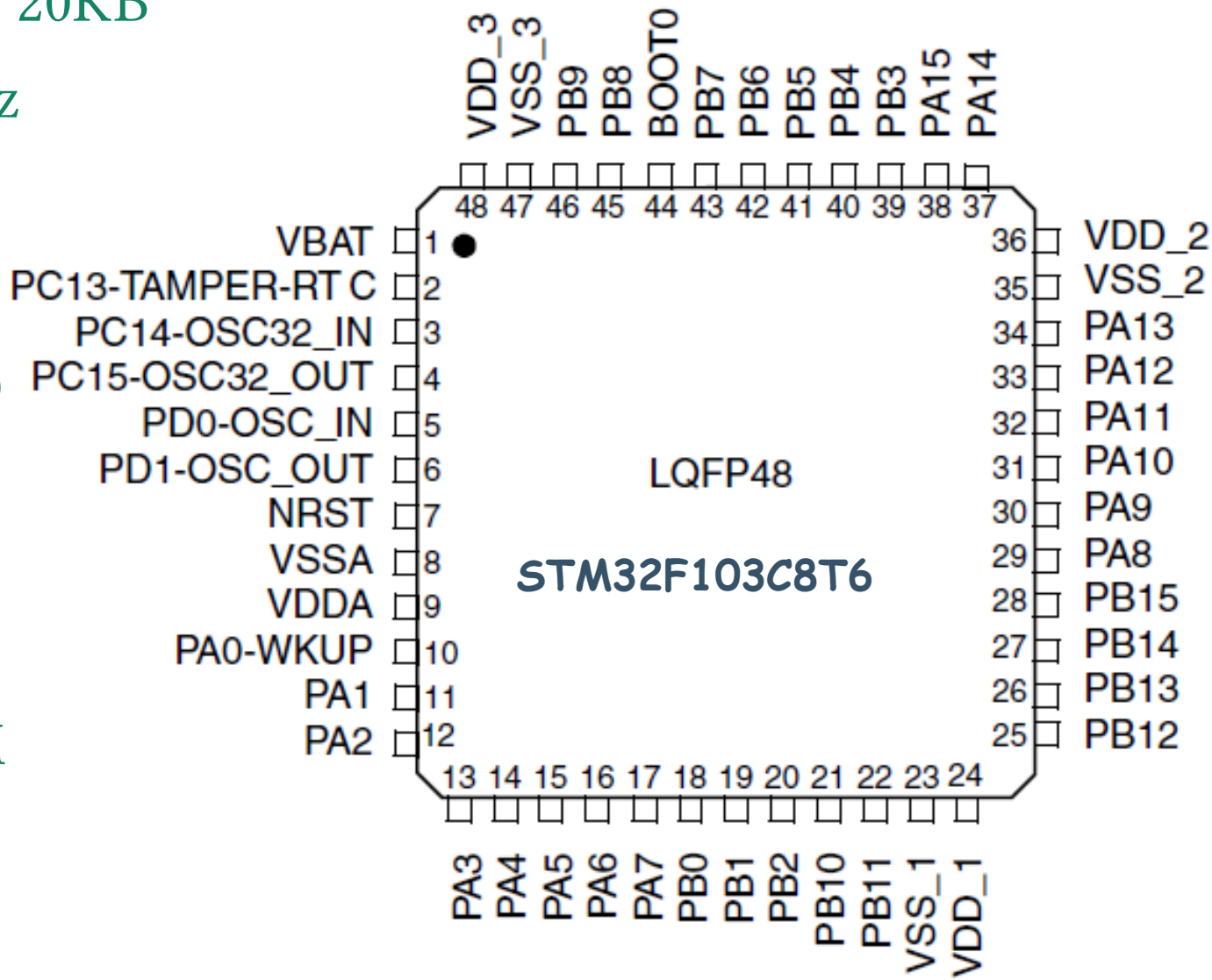


STM32F103 rendszer órajel források

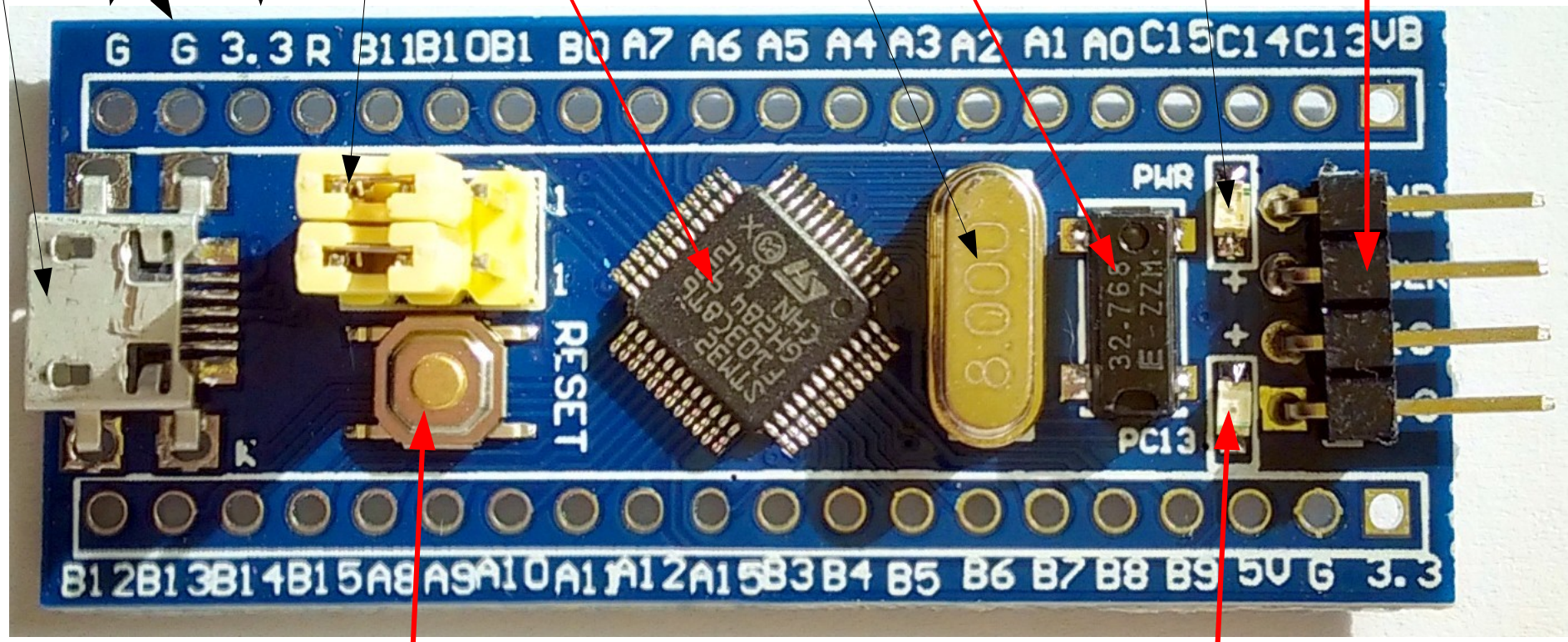


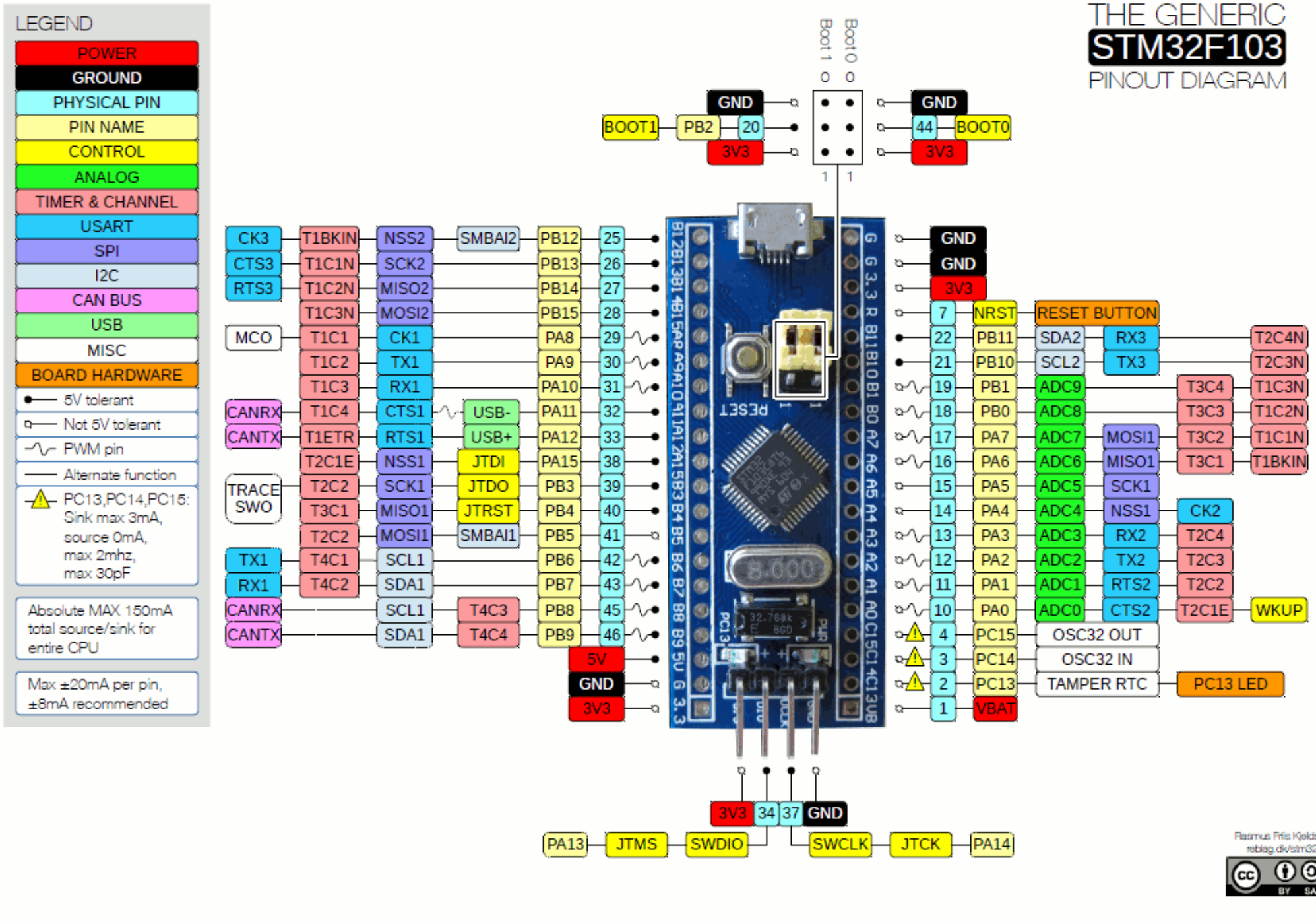
Az STM32F103C8T6 mikrovezérlő

- 32-bit ARM Cortex-M3 CPU
- Flash: 64KB / RAM: 20KB
- Órajel: max. 72 MHz
- Tápfesz.: 3,3 V
- Digitális I/O: 37
- Analóg bemenet: 10
- 2db 12 bites ADC
- USB, 2 SPI, 2 I2C, 3 UART, CAN
- 4 Timer, RTC, PWM



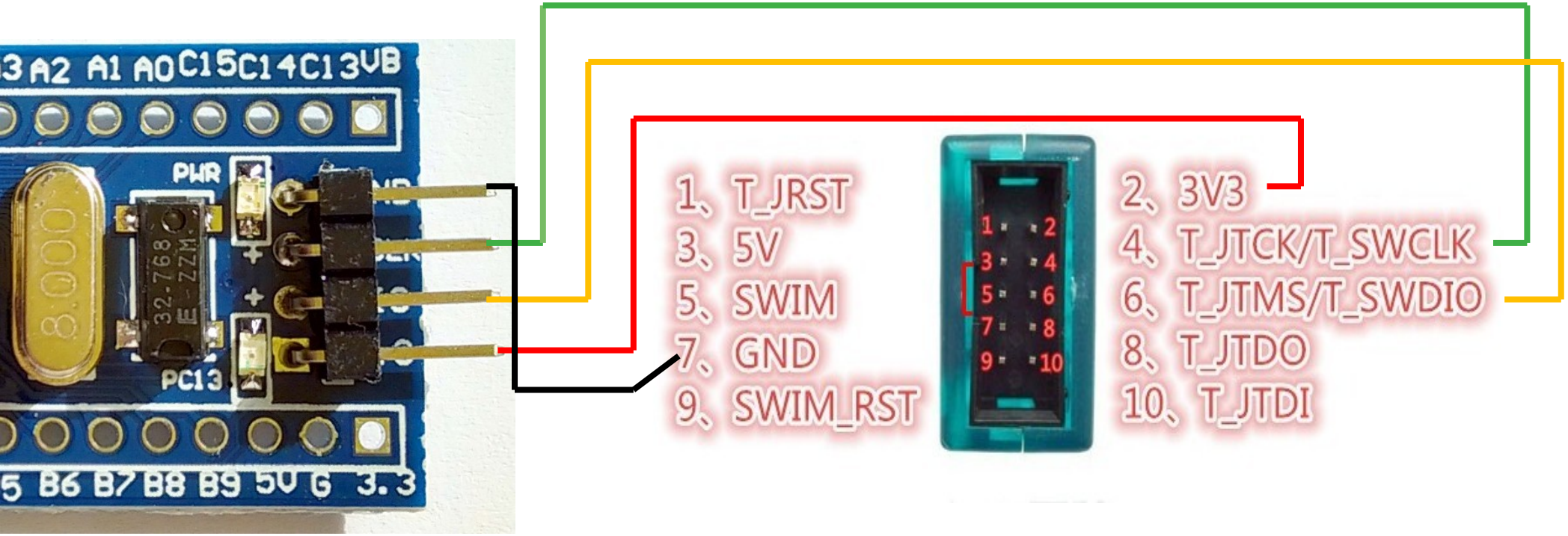
STM32F103C8 „Blue Pill” kártya




LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot1
Boot0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14
Rasmus Fris Kjeldsen
reblag.dk/stm32
V1.0

ST-link V2 J-tag programozó bekötése

- A képen látható ST-link klón SWIM (ST8), SWD (STM32) és JTAG (STM32) programozásra és nyomkövetésre is alkalmas
- Egy ellenállás beiktatásával SWO trace képessé tehető



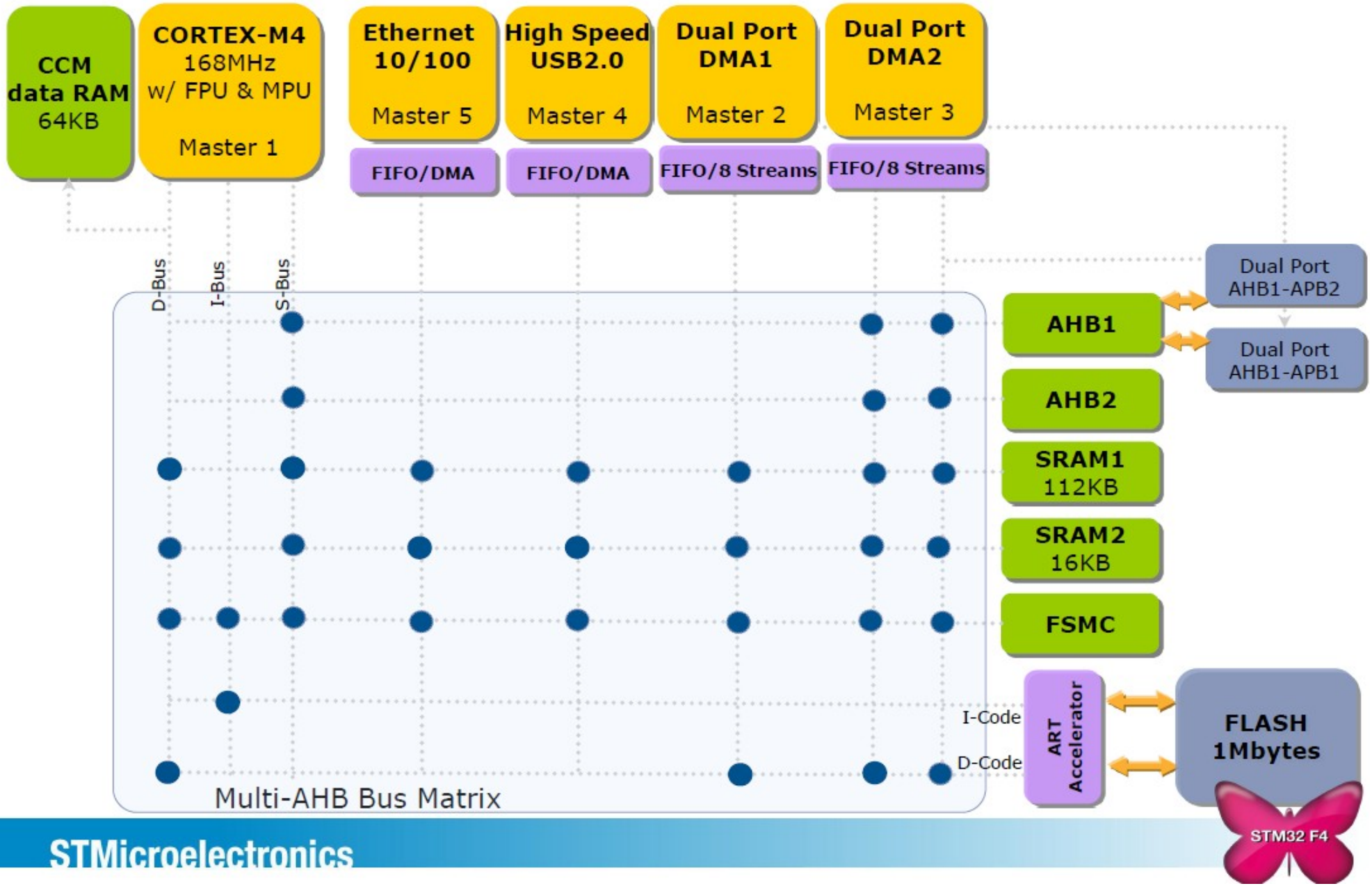
STM32 F4 termékcsalád portfolio

Product lines	F_{CPU} (MHz)	Flash (Kbytes)	RAM (KB)	Ethernet I/F IEEE 1588	2x CAN	Camera I/F	SDRAM I/F	Dual Quad-SPI	SAI	SPDIF RX	Chrom-ART Graphic Accelerator™	TFT LCD Controller	MIPI DSI
Advanced lines													
STM32F469[2]	180	512 K to 2056 K	384	•	•	•	•	•	•		•	•	•
STM32F429[2]	180	512 K to 2056 K	256	•	•	•	•		•		•	•	
STM32F427[2]	180	1024 K to 2056 K	256	•	•	•	•		•		•		
Foundation lines													
STM32F446	180	256 K to 512 K	128		•	•	•	•	•	•			
STM32F407[2]	168	512 K to 1024 K	192	•	•	•							
STM32F405[2]	168	512 K to 1024 K	192		•								

STM32 F4 termékcsalád portfolio

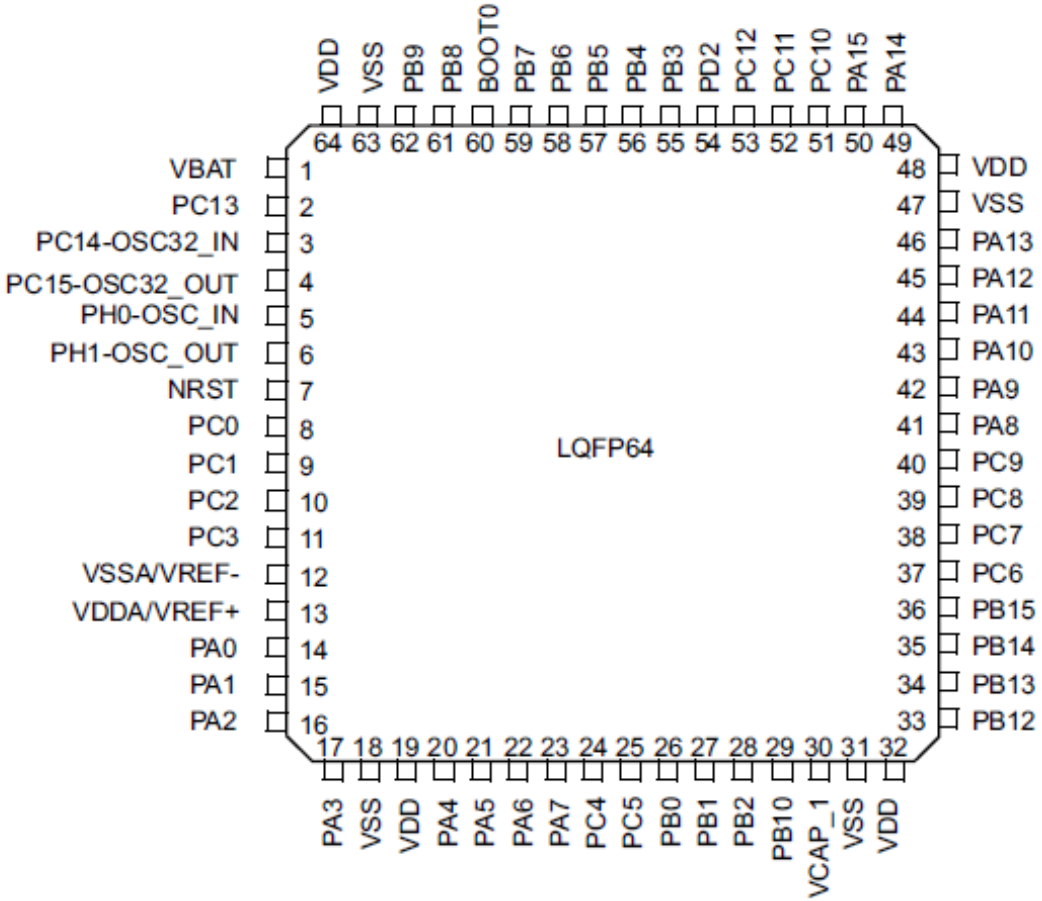
Product lines	F_{CPU} (MHz)	Flash (Kbytes)	RAM (KB)	RUN current (µA/MHz)	STOP current (µA)	Small package (mm)	FSMC (NOR/ PSRAM/LCD support	QSPI	DFSDM	DAC	TRNG	DMA Batch Aquisition Mode	USB 2.0 OTG FS
Access lines													
STM32F401	84	128 K to 512 K	up to 96	Down to 128	Down to 10	Down to 3x3							•
STM32F410	100	64 K to 128 K	32	Down to 89	Down to 6	Down to 2.553x 2.579				•	•	BAM	-
STM32F411	100	256 K to 512 K	128	Down to 100	Down to 12	Down to 3.034x 3.22						BAM	•
STM32F412	100	512 K to 1024 K	256	Down to 112	Down to 18	Down to 3.653x 3.651	•	•	•		•	BAM	• +LPM[1]
STM32F413[2]	100	1024 K to 1536 K	320	Down to 115	Down to 18	Down to 3.951x 4.039	•	•	•	•	•	BAM	• +LPM[1]

STM32F4 és a Multi-AHB buszmátrix

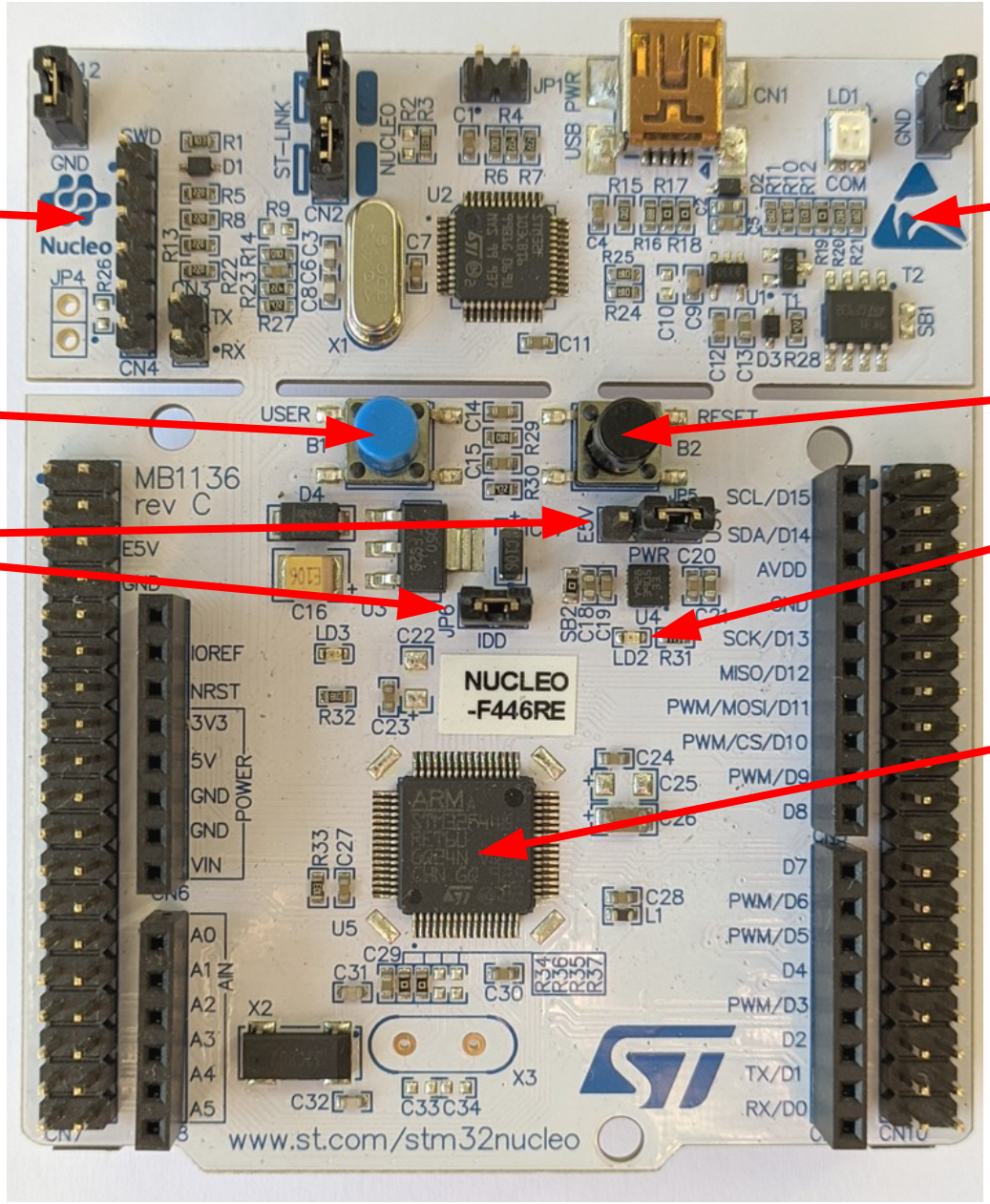


Az STM32F446RET6 mikrovezérlő

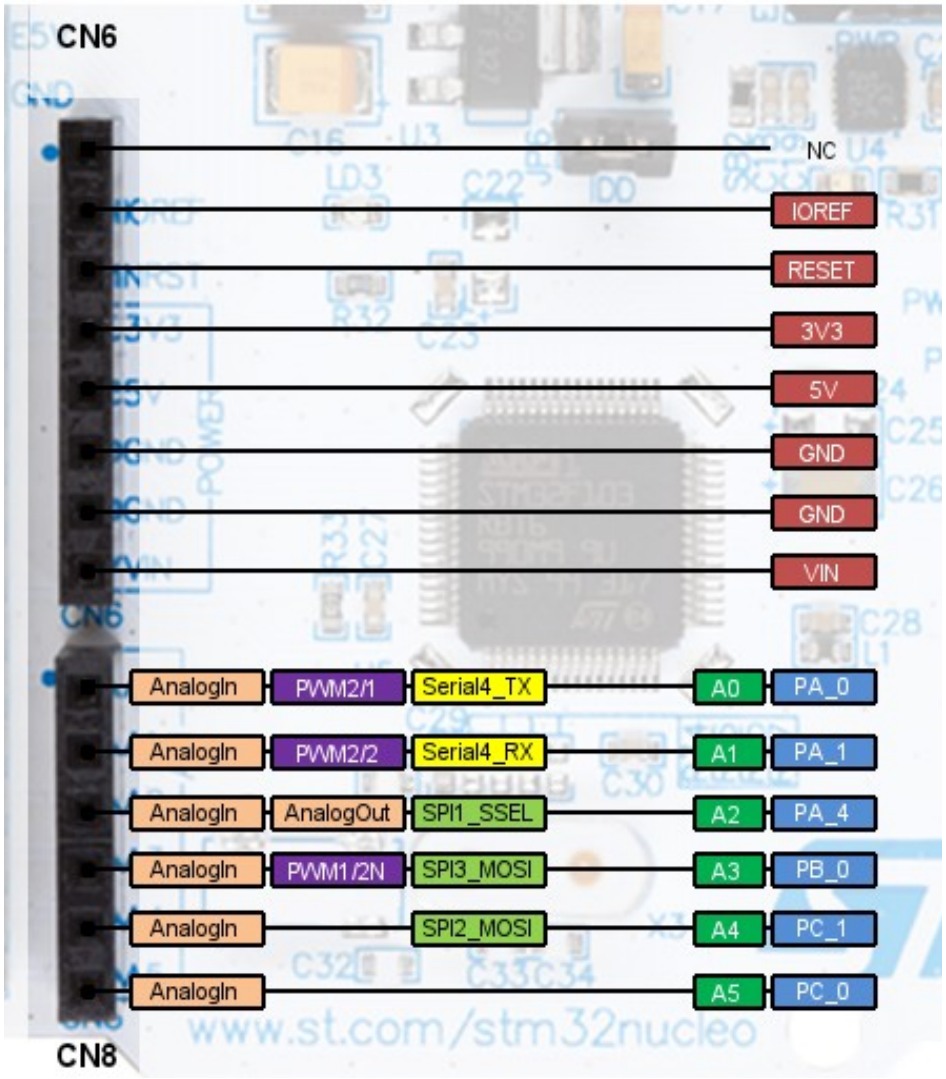
- 32-bit ARM Cortex-M4F CPU
- Flash: 512KB / RAM: 128KB
- Órajel: max. 180 MHz
- Tápfesz.: 3,3 V (1.8 V – 3.6 V)
- Digitális I/O: 50
- Analóg bemenet: 16
- 3db 12 bites ADC
- 2 db 12 bites DAC
- USB FS/HS, 4 SPI, 4 I2C,
  4 USART, 2 UART, 2 CAN
- 10+2+2 Timer
- RTC, SDIO

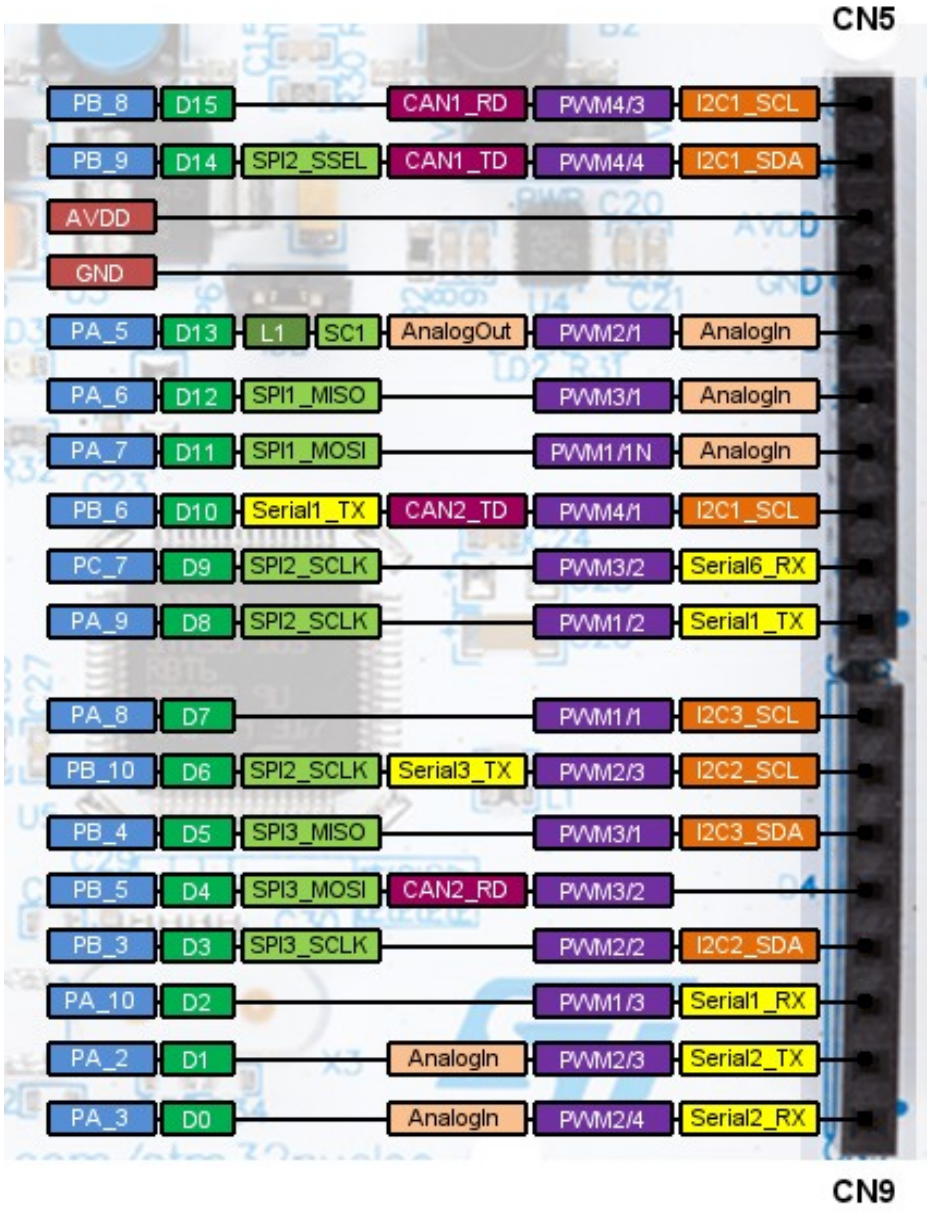


A NUCLEO-F446RE fejlesztői kártya

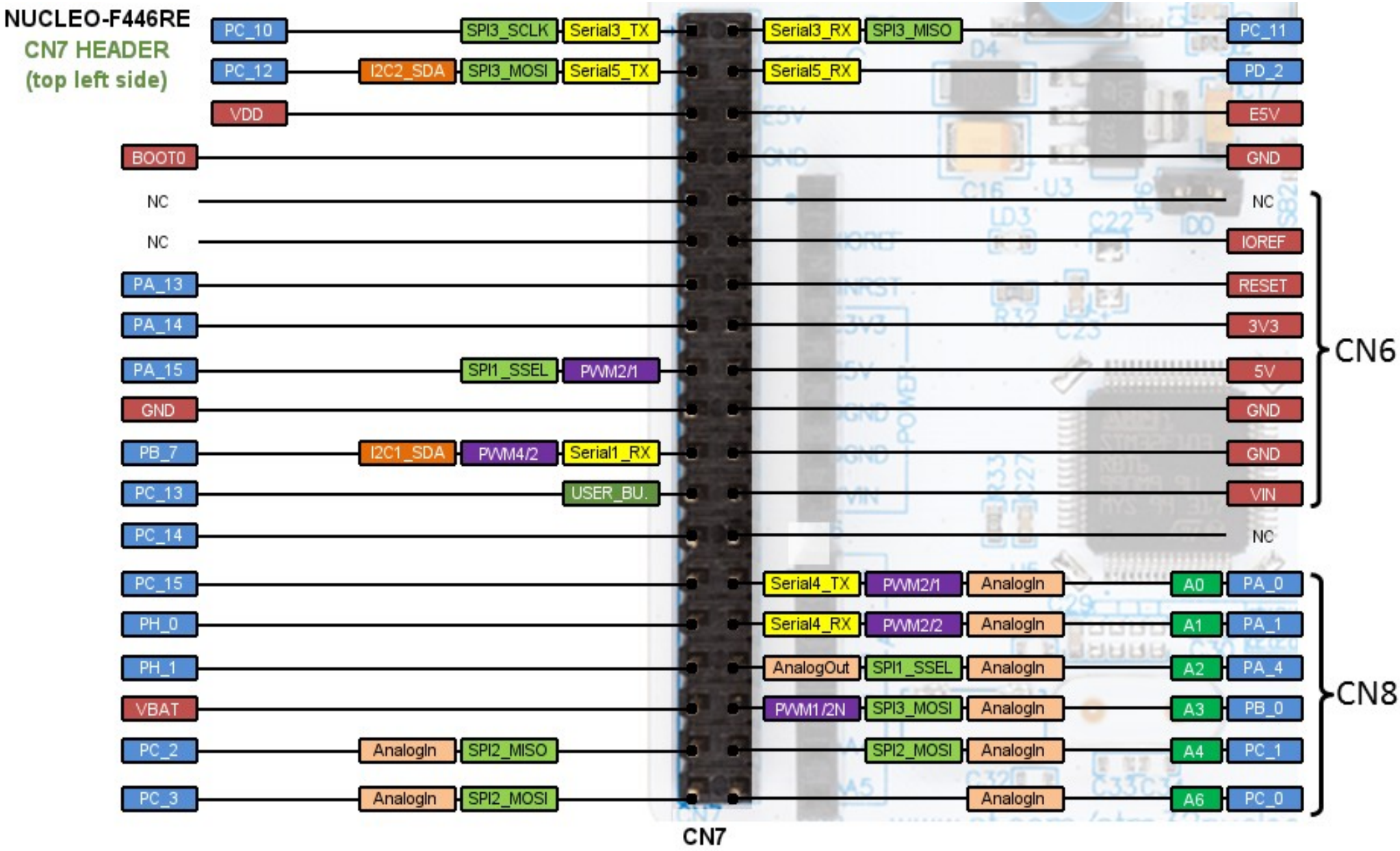


Arduino kompatibilis kivezetések

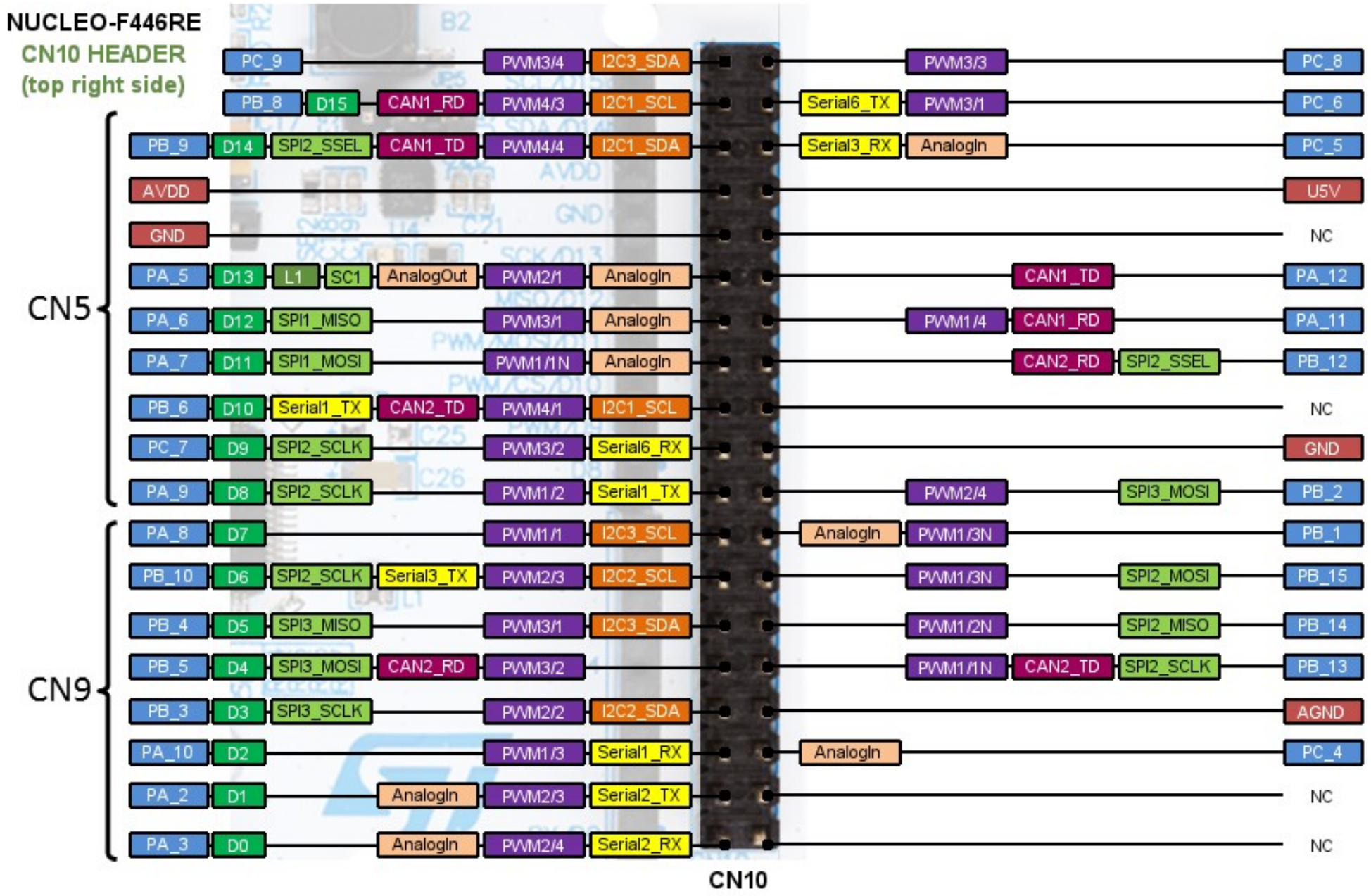




Morpho kivezetések - CN7



Morpho kivezetések - CN10



Lab01 projektek: GPIO portok kezelése

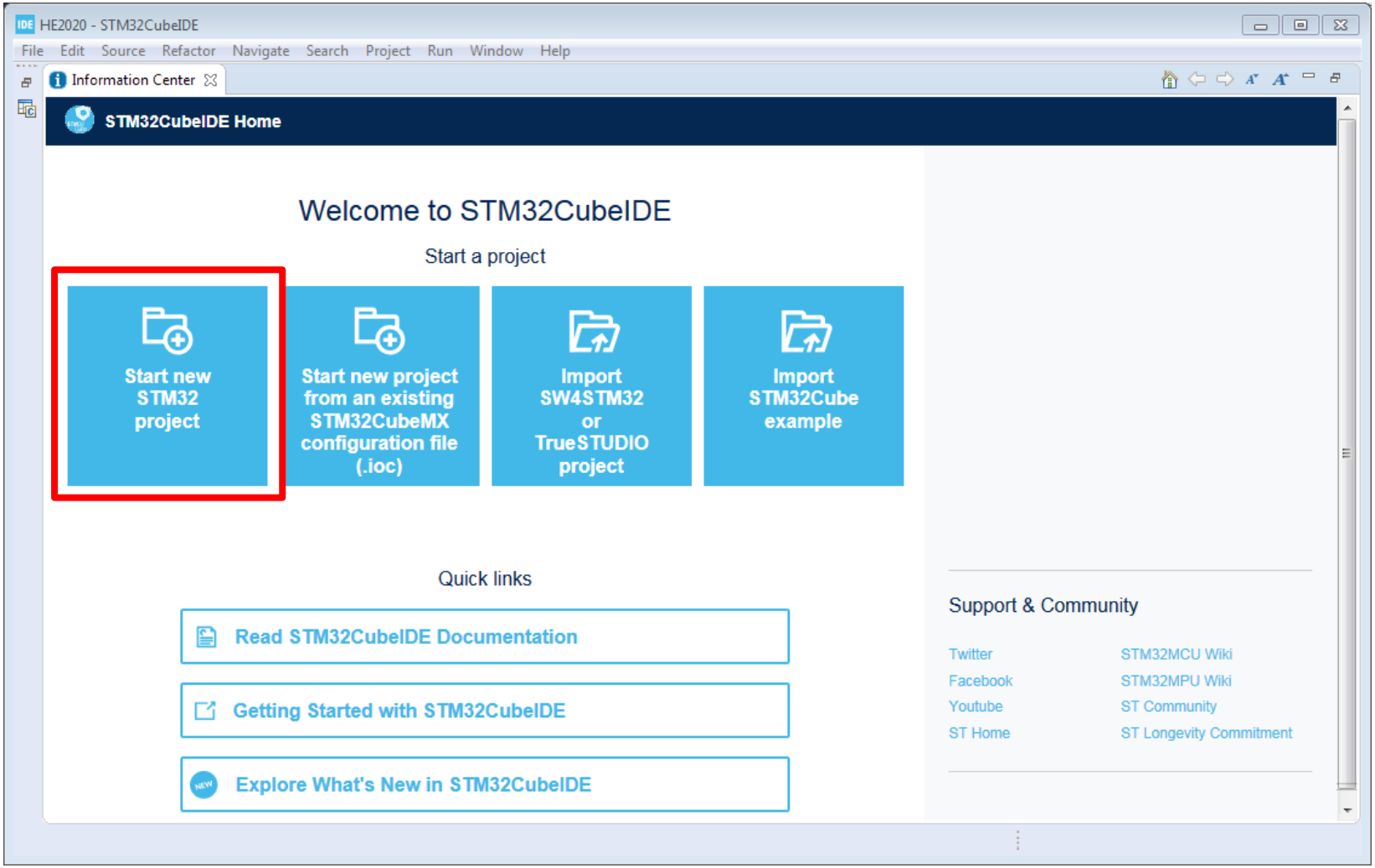
- **Lab01/F103_blinky:** A beépített LED (**PC13**) villogtatása a „Blue pill” kártyán (STM32F103)
- **Lab01/F446RE_blinky:** A beépített LED (**PA5**) villogtatása a Nucleo-F446RE kártyán (STM32F446RE)
- **Lab01/F103_button:** A beépített LED kapcsolgatása a **PB0** bemenetre kötött nyomógombbal a „Blue pill” kártyán (STM32F103)
- **Lab01/F446RE_button:** A beépített LED kapcsolgatása a **PC13** bemenetre kötött beépített nyomógombbal a Nucleo-F446RE kártyán (STM32F446RE)

Lab01/F103_blinky projekt

- A beépített LED villogtatása a „Blue pill” kártyán (STM32F103), a következő tevékenységet ismételgetve:
  - Felkapcsoljuk a LED-t, majd várunk 50 ms-ot
  - Lekapcsoljuk a LED-et, majd várunk 2500 ms-ot
- A beépített LED katódját a „Blue pill” kártyán a **C** port 13. bitje vezérli. A LED lehúzáskor (alacsony kimeneti szint) világít
- A **PC13** kimenetre **LED** névvel hivatkozzunk!
- A kártyán egy 8 MHz-es és egy 32 kHz-es kvarc rezonátor is be van kötve (HSE, ill. LSE)
- A rendszer órajelét a 8 MHz-es HSE órajelből PLL-lel állítsuk elő, s a SYSCLK = 72 MHz-es maximális frekvenciát állítsuk be!
- A periféria buszok órajele APB1 = 36 MHz, APB2 = 72 MHz legyen!

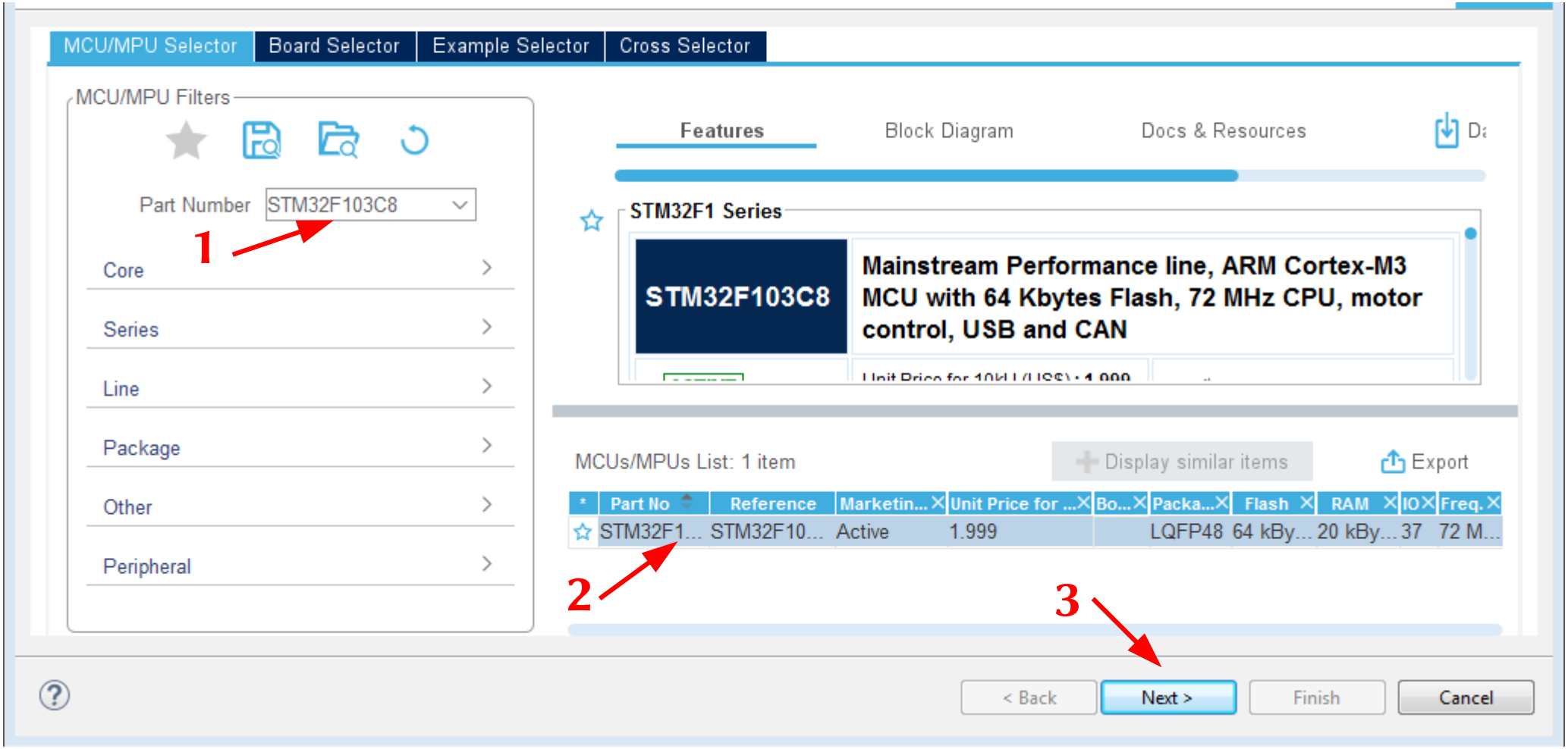
Új projekt létrehozása (LED villogtatás)

- Az **STM32CubeIDE** indítása után kattintsunk az első gombra!
- A projekt neve legyen pl. **F103_blinky**



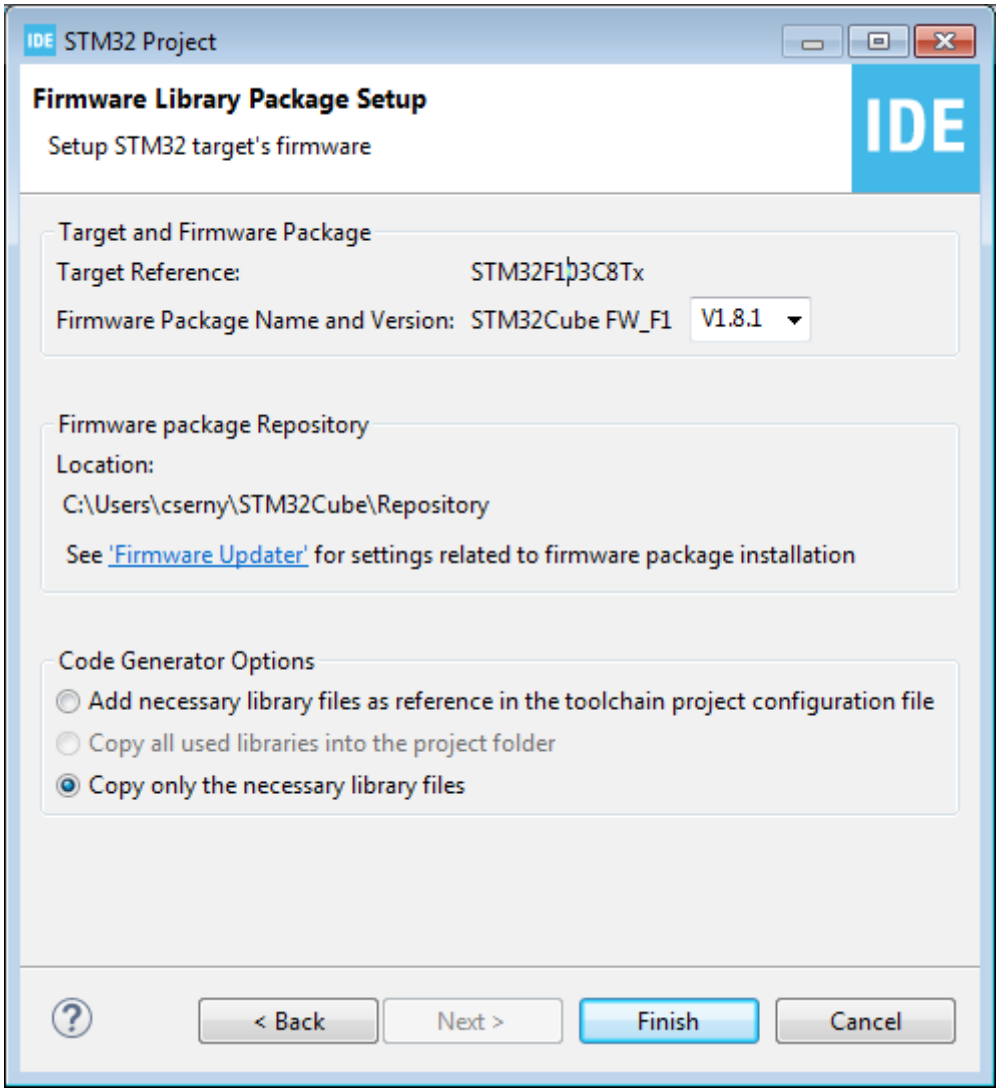
Az MCU kiválasztása

- Az MCU típusát a **Part Number** legördülő menüből kereshetjük ki, vagy írhatjuk be. Elsőként válasszuk az **STM32F103C8** típust!
- A jobboldali listában kattintsunk rá, majd a **Next** gombra



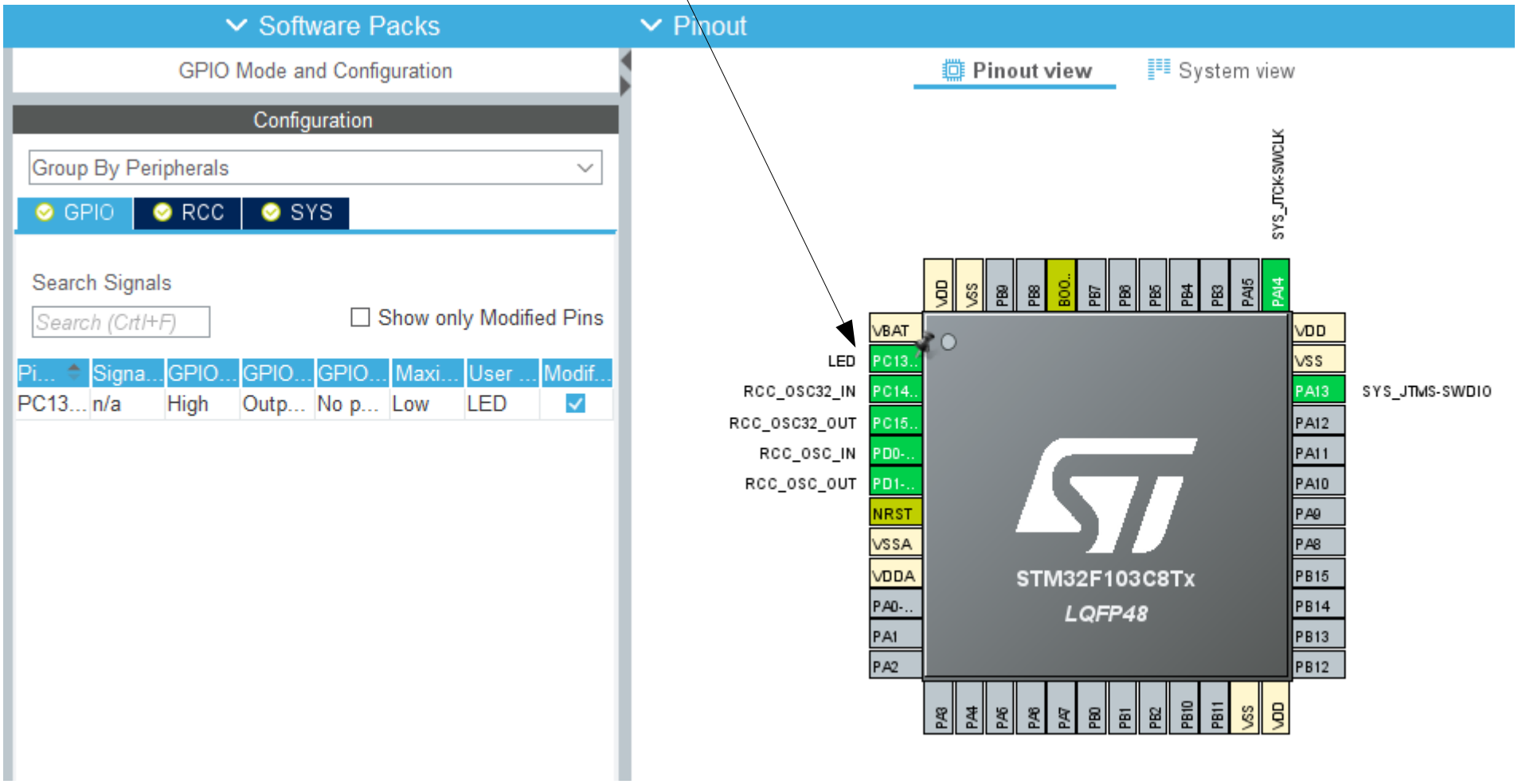
Firmware csomag kiválasztása és opciói

- A **Firmware Package** kiválasztásával nem kell foglalkozni, ha az automatikusan kiválasztott legfrissebb verzióval dolgozunk
- A legalsó opció bejelölésével azt kértük, hogy csak a szükséges fájlokat másoljuk be a projekt könyvtárba
- A fájlok hivatkozásként történő becsatolása helytakarékosabb megoldás volna, de rontja a hordozhatóságot!
- Végül kattintsunk a **Finish** gombra!



A Device Configuration Tool használata

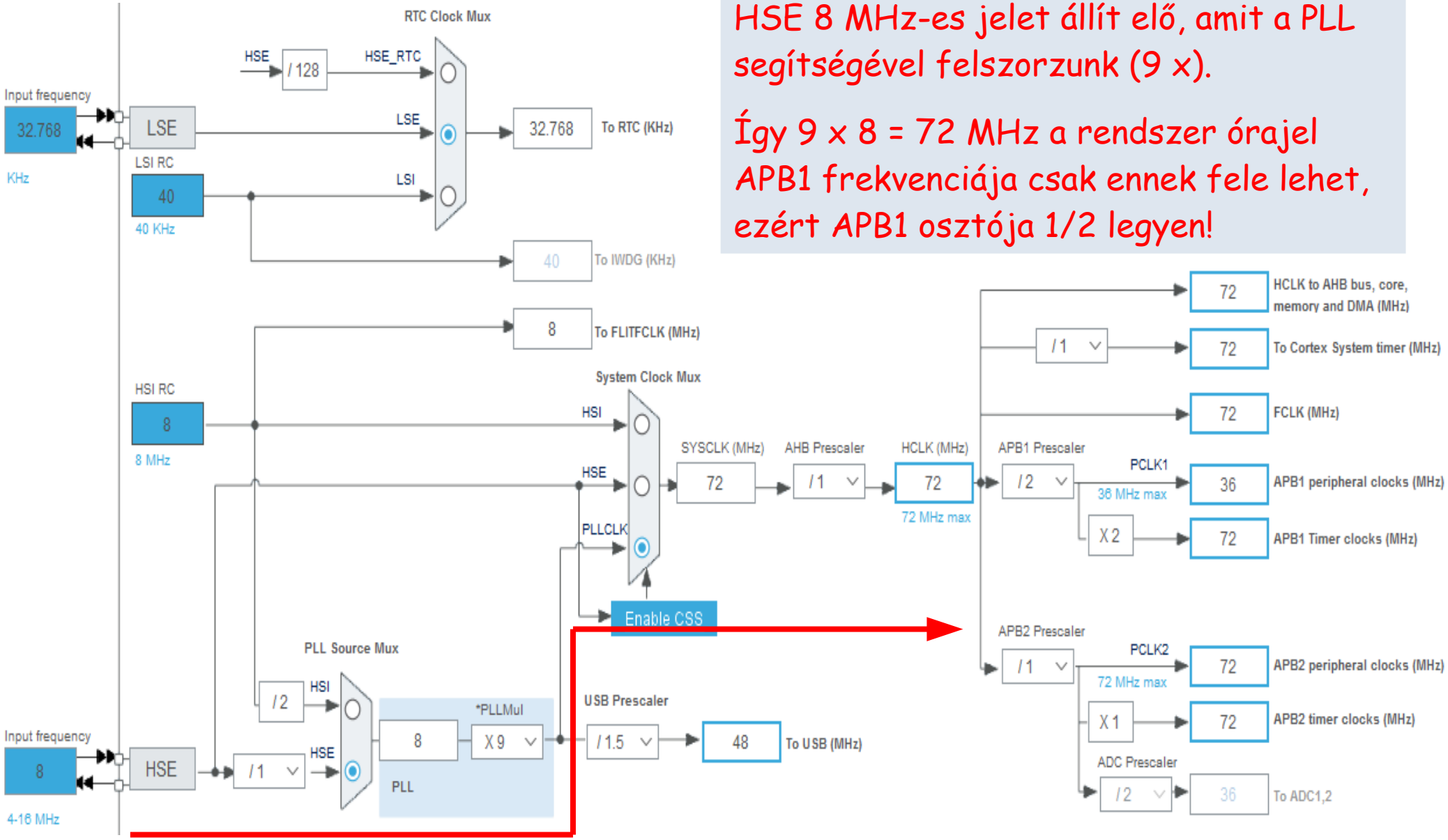
- A **System Core / SYS** Debug eszköze legyen **Serial Wire**
- **RCC** modulnál **HSE** és **LSE** legyen **Crystal/ceramic resonator**
- A **PC13 GPIO** kivezetés a LED, ezt konfiguráljuk kimenetnek



STM32F103 órajelek konfigurálása

HSE 8 MHz-es jelet állít elő, amit a PLL segítségével felszorzunk (9 x).

Így 9 x 8 = 72 MHz a rendszer órajel APB1 frekvenciája csak ennek fele lehet, ezért APB1 osztója 1/2 legyen!



Kódgenerálás és módosítás

- A **Project Manager** lapon most ne változtassunk semmit!
- Kattintsunk a **Project** menü **Generate Code** pontjára!
- A generált projektben **main.c** végtelen ciklusát még ki kell egészítenünk a LED-et kapcsolgató és a késleltető parancsokkal

```c
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  while (1) {
    HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET); // LED on
    HAL_Delay(50);                                           // 50 ms
    HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);   // LED off
    HAL_Delay(2500);                                         // 2500 ms
  }
}
```

A LED elérhetőségét main.h definiálja:

```c
60  /* Private defines -----------------
61  #define LED_Pin GPIO_PIN_13
62  #define LED_GPIO_Port GPIOC
```

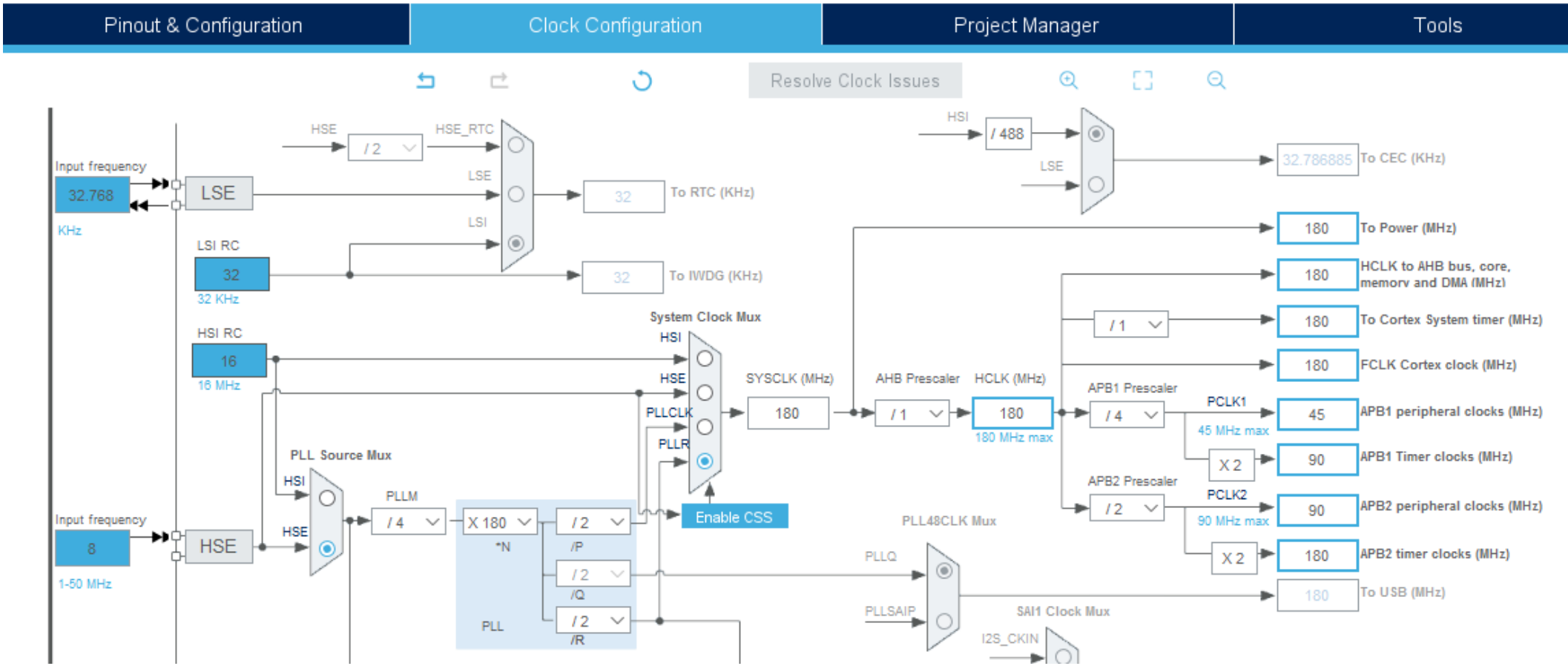
A LED lehúzásra világít!

Lab01/F44RE_blinky projekt

- A beépített LED villogtatása a **NUCLEO-F446RE** kártyán (STM32F446RE), a következő tevékenységet ismételgetve:
  - Felkapcsoljuk a LED-t, majd várunk 50 ms-ot
  - Lekapcsoljuk a LED-et, majd várunk 2500 ms-ot
- A beépített LED anódját a NUCLEO kártyán a **A** port 5. bitje vezérli. A LED felhúzáskor (magas kimeneti szint) világít
- A **PA5** kimenetre **LED** névvel hivatkozzunk!
- A kártya a programozó felől kap külső, 8 MHz-es órajelet (HSE) és egy 32 kHz-es kvarc rezonátor is van a kártyán (LSE)
- A rendszer órajelét a 8 MHz-es HSE órajelből PLL-lel állítsuk elő, s a SYSCLK = 180 MHz-es maximális frekvenciát állítsuk be!
- A periféria buszok órajele APB1 = 45 MHz, APB2 = 90 MHz legyen!

Lab01/F446RE_blinky projekt

- Az előzőhöz hasonlóan készült a **Lab01/F446RE_blinky** projekt
- MCU: **STM32F446RE**, a LED az **A5** kimenetre csatlakozik
- HSE órajele: Bypass (a programozótól jön) és 8 MHz
- SYSCLK: 180 MHz, APB2: 90 MHz, APB1: 45 MHz lehet legfeljebb



A generált kód kiegészítése (main.c)

- A generált projektben **main.c** végtelen ciklusát itt is ki kell egészítenünk a LED-et kapcsolgató és a késleltető parancsokkal
- A LED elérhetőségét main.h így definiálja:

```
#define LED_Pin GPIO_PIN_5
#define LED_GPIO_Port GPIOA
```

```
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  while (1) {
    HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);   // LED on
    HAL_Delay(50);                                           // 50 ms
    HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET); // LED off
    HAL_Delay(2500);                                         // 2500 ms
  }
}
```

A LED felhúzásra világít!

Amire ügyelnünk kell ...

- Az általunk beírt sorok mindig egy **USER CODE BEGIN** és **USER CODE END** virtuális zárójelpár közé kerüljenek, különben elvesznek egy újabb konfigurációmódosítás és projekt újragenerálás során!
- Példa: egy LED villogtató program részlete (NUCLEO-F446RE)

```
 97   /* Infinite loop */
 98   /* USER CODE BEGIN WHILE */
 99   while (1)
100   {
101     /* USER CODE END WHILE */
102
103     /* USER CODE BEGIN 3 */
104       HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
105       HAL_Delay(250);
106   }
107   /* USER CODE END 3 */
108 }
```

- Az **.ioc** kiterjesztésű állomány írja le a konfigurációt, ez az **STM32Cube MX** programmal közvetlenül is megnyitható

STM32F103 GPIO portok regiszterei

- A GPIO portokat mindig 32 bites egységként kell címezni, de csak az alsó 16 bitjük van implementálva.
- Minden GPIO porthoz az alábbi 32 bites regiszterek tartoznak:
- **GPIOx_CRL** – konfigurációs és mód bitek 0..7 (offset: 0x00 RW)
- **GPIOx_CRH** – konfigurációs és mód bitek 8..15 (offset: 0x04 RW)
- **GPIOx_IDR** – bemeneti adatregiszter (offset: 0x08 R)
- **GPIOx_ODR** – kimeneti adatregiszter (offset: 0x0C RW)
- **GPIOx_BSRR** – bit törlés/beállítás regiszter (offset: 0x10 W)
- **GPIOx_BRR** – bit törlés regiszter (offset: 0x14 W)
- **GPIOx_LCKR** – konfiguráció rögzítés regiszter (offset: 0x18 RW)

STM32F103 port kivezetések konfigurálása

- A portok alsó és a felső 8 bitjének üzemmódját a **GPIOx_CRL**, illetve **GPIOx_CRH** regiszterekben állíthatjuk be (L: low, H: high)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Például **GPIOC_CRH** címe: `0x40011004`

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 21*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

Mode bits

00: fenntartott
01: max. 10 MHz
10: max. 2 MHz
11: max 50 MHz

Beépített LED-hez (PORTC 13. bit)

CNF: 00

MODE: 10 kell!

STM32F446 GPIO portok regiszterei

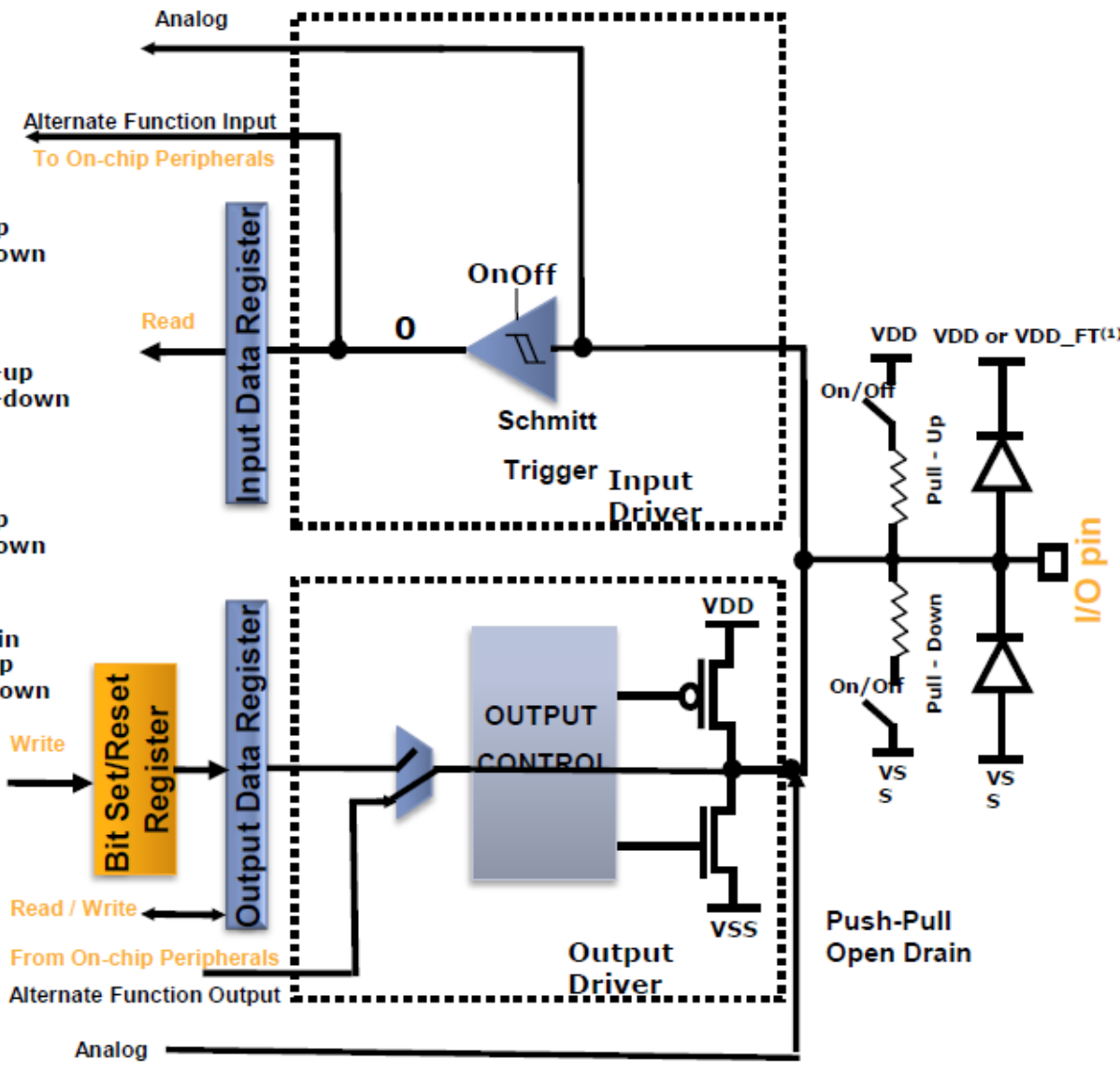
Minden GPIO porthoz az alábbi 32 bites regiszterek tartoznak:

- **GPIOx_MODER** – üzemmód (**00:** IN, **01:** OUT, **10:** AF, **11:** analog)
- **GPIOx_OTYPER** – kimenettípus választó regiszter (**0:** PP, **1:** OD)
- **GPIOx_OSPEEDER** – sebesség (**00:** low, **01:** medium, **10:** fast, **11:** high)
- **GPIOx_PUPDR** – le/felhúzás (**00:** no, **01:** PUP, **10:** PDN, **11:** reserved)
- **GPIOx_IDR** – bemeneti adatregiszter
- **GPIOx_ODR** – kimeneti adatregiszter
- **GPIOx_BSRR** – bit törlés/beállítás regiszter
- **GPIOx_BRR** – bit törlés regiszter
- **GPIOx_LCKR** – konfiguráció rögzítés regiszter
- **GPIOx_AFRL** – alternatív funkció választó bitek (0..7)
- **GPIOx_AFRH** – alternatív funkció választó bitek (8..15)

STM32F446 port kivezetések konfigurálása

MODER(i) [1:0]	OTYPER(i) [1:0]	PUPDR(i) [1:0]	I/O configuration
01	0	0 0	Output Push Pull
		0 1	Output Push Pull with Pull-up
		1 0	Output Push Pull with Pull-down
	1	0 0	Output Open Drain
		0 1	Output Open Drain with Pull-up
		1 0	Output Open Drain with Pull-down
10	0	0 0	Alternate Function Push Pull
		0 1	Alternate Function PP Pull-up
		1 0	Alternate Function PP Pull-down
	1	0 0	Alternate Function Open Drain
		0 1	Alternate Function OD Pull-up
		1 0	Alternate Function OD Pull-down
00	x	0 0	Input floating
		0 1	Input with Pull-up
		1 0	Input with Pull-down
11	x	x	Analog mode

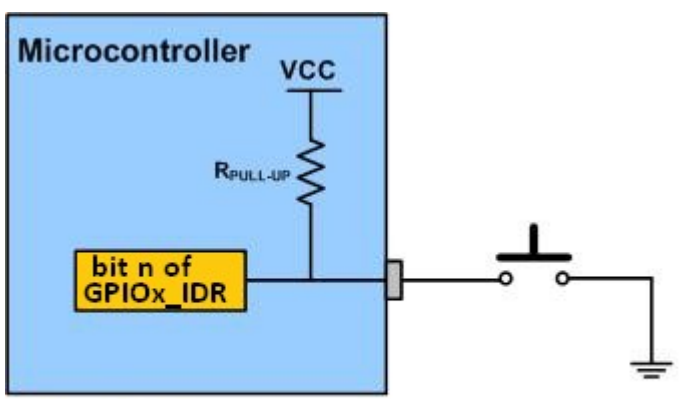
* In output mode, the I/O speed is configurable through OSPEEDR register: 2MHz, 25MHz, 50MHz or 100 MHz



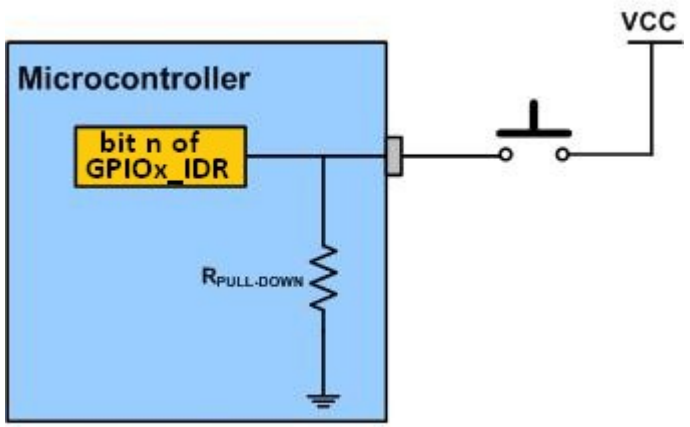
(1) VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.

Nyomógomb állapotának beolvasása

- A nyomógombot kétféle módon is használhatjuk digitális bemenetek vezérlésére: lehúzásra, vagy felhúzásra



(a) Using Pull-up



(b) Using Pull-down

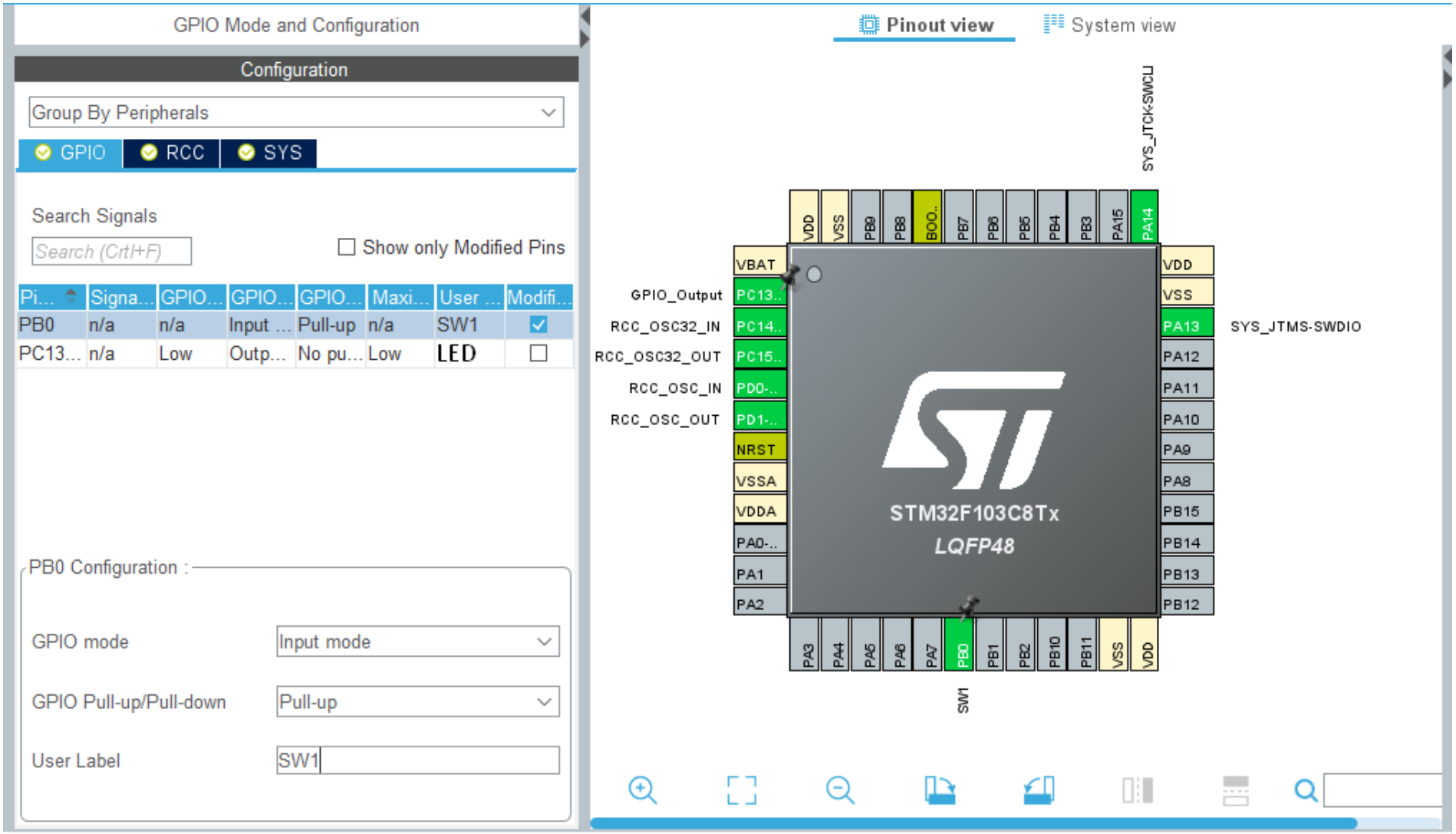
- Esetünkben a nyomógombot az **SW1** kivezetés és a közös pont (**GND**) közé kötjük, a nyomógomb megnyomásakor a bemenet alacsony szintre kerül (földre zár)
- **A hardver bekötés ez lesz:**
  SW1 = **PB0** (F103), illetve **PC13** (F446)
- A nyomógomb elengedett állapotában külső vagy belső felhúzásnak kell gondoskodnia róla, hogy a bemenet magas szinten legyen, s ne „lebegjen”. Mi most a **belső felhúzást** fogjuk használni

Lab01/F103_button projekt

- A beépített LED-et vezéreljük egy nyomógomb segítségével a **„Blue pill”** kártyán (STM32F103C8): A LED csak akkor világítson, amikor a nyomógomb le van nyomva!
- A beépített LED a **PC13** kimenet alacsony állapotában világít
- A nyomógomb, amelyre **SW1** néven fogunk hivatkozni a **PB0** bemenet és **GND** közé legyen kötve, lenyomáskor **GND**-hez zár. Használjunk belső felhúzást!
- A kártyán egy 8 MHz-es és egy 32 kHz-es kvarc rezonátor is be van kötve (HSE, ill. LSE)
- A rendszer órajelét a 8 MHz-es HSE órajelből PLL-lel állítsuk elő, s a SYSCLK = 72 MHz-es maximális frekvenciát állítsuk be!
- A periféria buszok órajele APB1 = 36 MHz, APB2 = 72 MHz legyen!

Lab01/F103_button projekt

- A projektet úgy konfiguráljuk, mint a LED villogtatásnál, csak a **PB0** GPIO kivezetést is konfiguráljuk (Input, pullup, SW1 név)



Lab01/F103_button projekt

- A generált projektben **main.c** forráskódját ki kell egészítenünk egy változó deklarálásával és a while ciklus teendőivel
- A ciklusba egy pergésmentesítő késleltetést is elhelyeztünk, bár itt ennek nincs jelentősége

```
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
volatile int button;

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  while (1) {
      button = HAL_GPIO_ReadPin(SW1_GPIO_Port,SW1_Pin);
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, button);
      HAL_Delay(20);
  }
}
```

Lab01/F446RE_button projekt

- A beépített LED-et vezéreljük a beépített nyomógomb segítségével a **NUCLEO-F446RE** kártyán (STM32F446RE): A LED csak akkor világítson, amikor a nyomógomb le van nyomva!
- A beépített LED a **PA5** kimenet magas állapotában világít
- A nyomógomb, amelyre **SW1** néven fogunk hivatkozni a **PC13** bemenet és **GND** közé van kötve, lenyomáskor **GND**-hez zár. Használjunk belső felhúzást!
- A kártya a programozó felől kap külső, 8 MHz-es órajelet (HSE) és egy 32 kHz-es kvarc rezonátor is van a kártyán (LSE)
- A rendszer órajelét a 8 MHz-es HSE órajelből PLL-lel állítsuk elő, s a SYSCLK = 180 MHz-es maximális frekvenciát állítsuk be!
- A periféria buszok órajele APB1 = 45 MHz, APB2 = 90 MHz legyen!

Lab01_F44RE_button projekt

- A projektet úgy konfiguráljuk, mint a LED villogtatásnál, csak a **PC13** GPIO kivezetést is be kell állítanunk (input, pullup, SW1 név)
- A generált projektben **main.c** forráskódját ki kell egészítenünk egy változó deklarálásával és a while ciklus teendőivel

```
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
volatile int button;

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  while (1) {
      button = HAL_GPIO_ReadPin(SW1_GPIO_Port,SW1_Pin);
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, !button);
      HAL_Delay(20);
  }
}
```

Negáció SW1 negatív logikája miatt!