STM32F4 Cube HAL

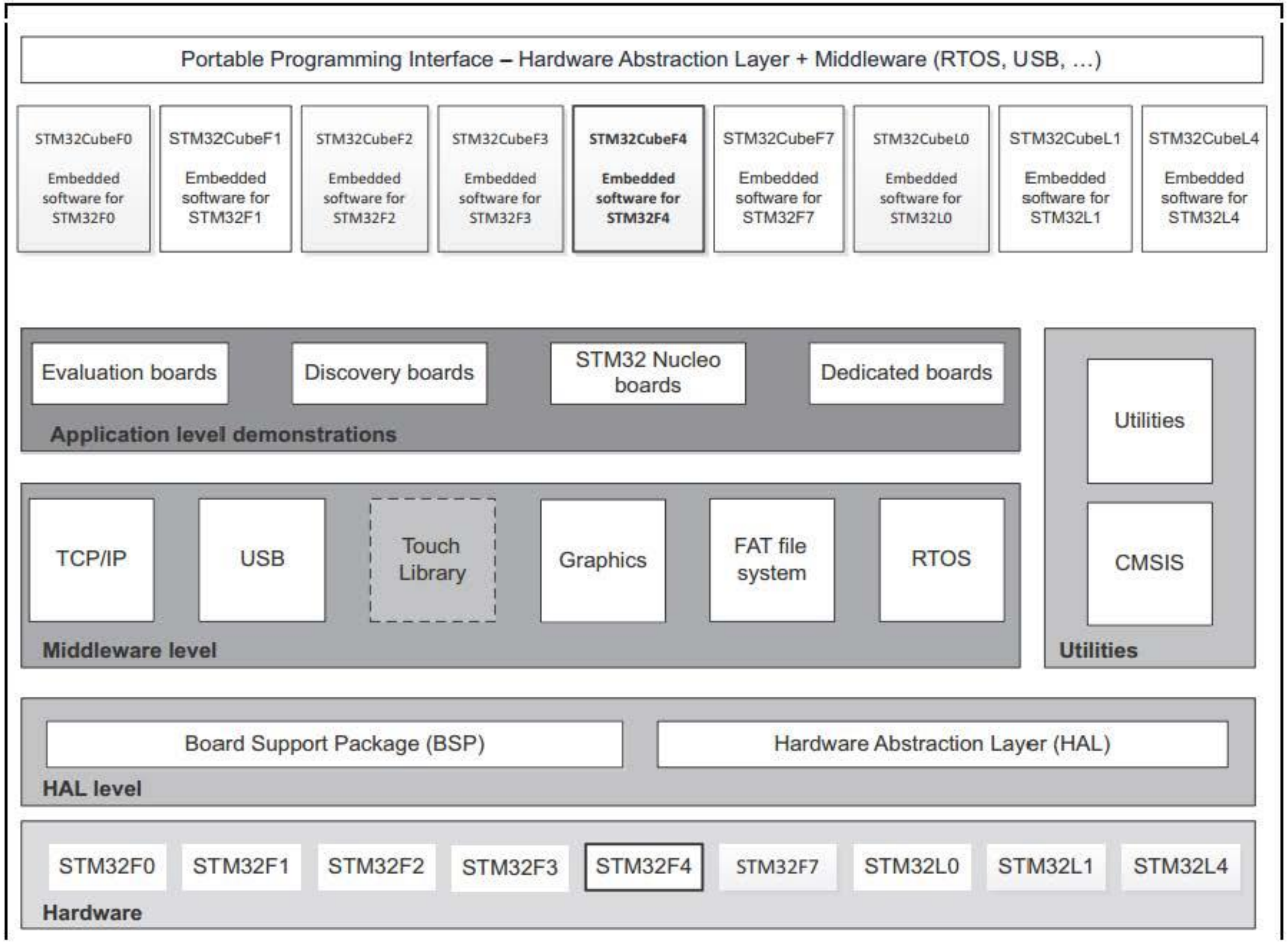
Introduction To Embedded Systems Programming

Amirhossein Shahshahani

WINTER 2018

Some notes

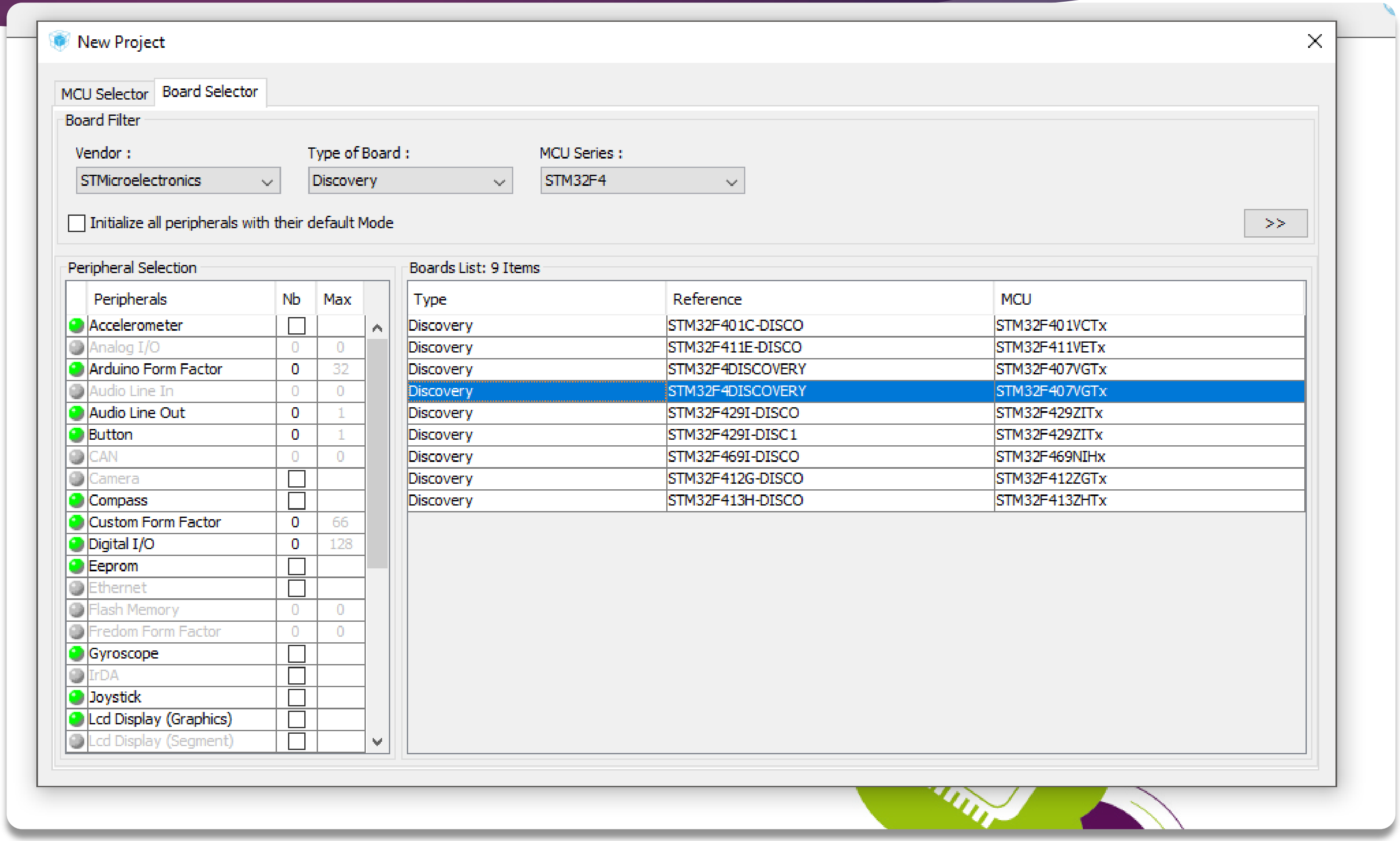
- These slides only present a small but main portion of the lecture. Due to the nature of the course, you need to go over datasheets, example codes and driver files. We cannot present those in slides.
- Some parts of assignments are based on students decisions and comprehension to choose a parameter, as a real engineering work.
- When using the STM32F4 Discovery board, under **NO CIRCUMSTANCES** you are to use, configure or touch the pins **PORTA 13, PORTA 14** and **PORTB3**.
- In Section 4.12 of the Doc_10-Disocvery Kit F4 Rev.C. pdf, you will see that those pins are reserved for the serial wire debug (SWD) functionality. That is, these pins are used to program and debug the board.



Portable Programming Interface – Hardware Abstraction Layer + Middleware (RTOS, USB, …)
STM32CubeF0 Embedded software for STM32F0
STM32CubeF1 Embedded software for STM32F1
STM32CubeF2 Embedded software for STM32F2
STM32CubeF3 Embedded software for STM32F3
STM32CubeF4 Embedded software for STM32F4
STM32CubeF7 Embedded software for STM32F7
STM32CubeL0 Embedded software for STM32L0
STM32CubeL1 Embedded software for STM32L1
STM32CubeL4 Embedded software for STM32L4
Evaluation boards
Discovery boards
STM32 Nucleo boards
Dedicated boards
Application level demonstrations
TCP/IP
USB
Touch Library
Graphics
FAT file system
RTOS
Middleware level
Utilities
CMSIS
Utilities
Board Support Package (BSP)
Hardware Abstraction Layer (HAL)
HAL level
STM32F0
STM32F1
STM32F2
STM32F3
STM32F4
STM32F7
STM32L0
STM32L1
STM32L4
Hardware

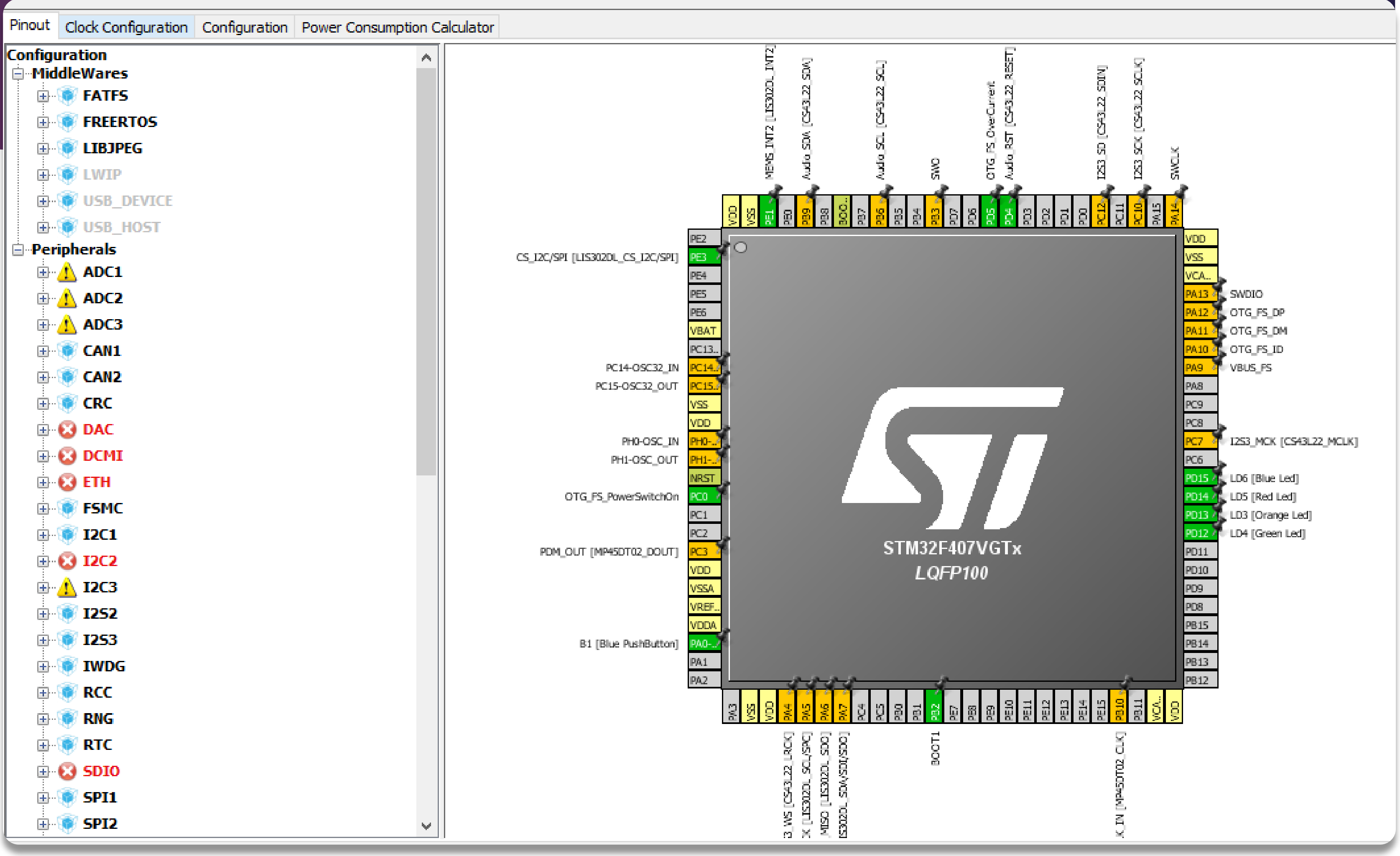
STM32CubeMx

- This software is a tool to make a base project with all peripheral configurations.
- You can select a board and enable configurations that are compatible and usable with your board. So, choose the STM32F407VGTx



Pinout

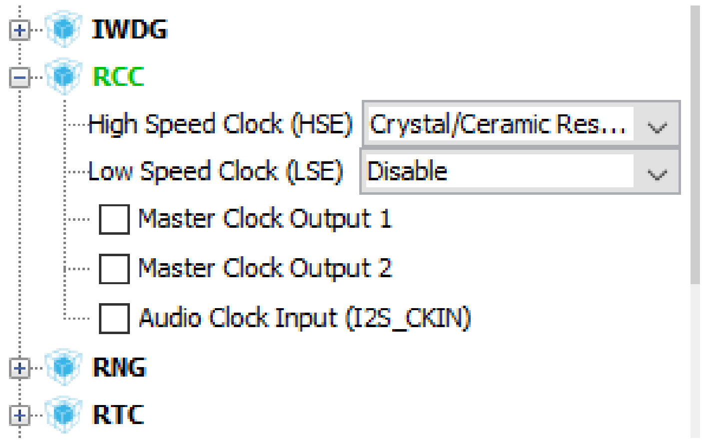
- You will see all GPIO pins reserved as Orange which are not recommended to be changed, since they are connected to other external components.
- Greens are changeable and grays are not assigned to any peripheral yet.



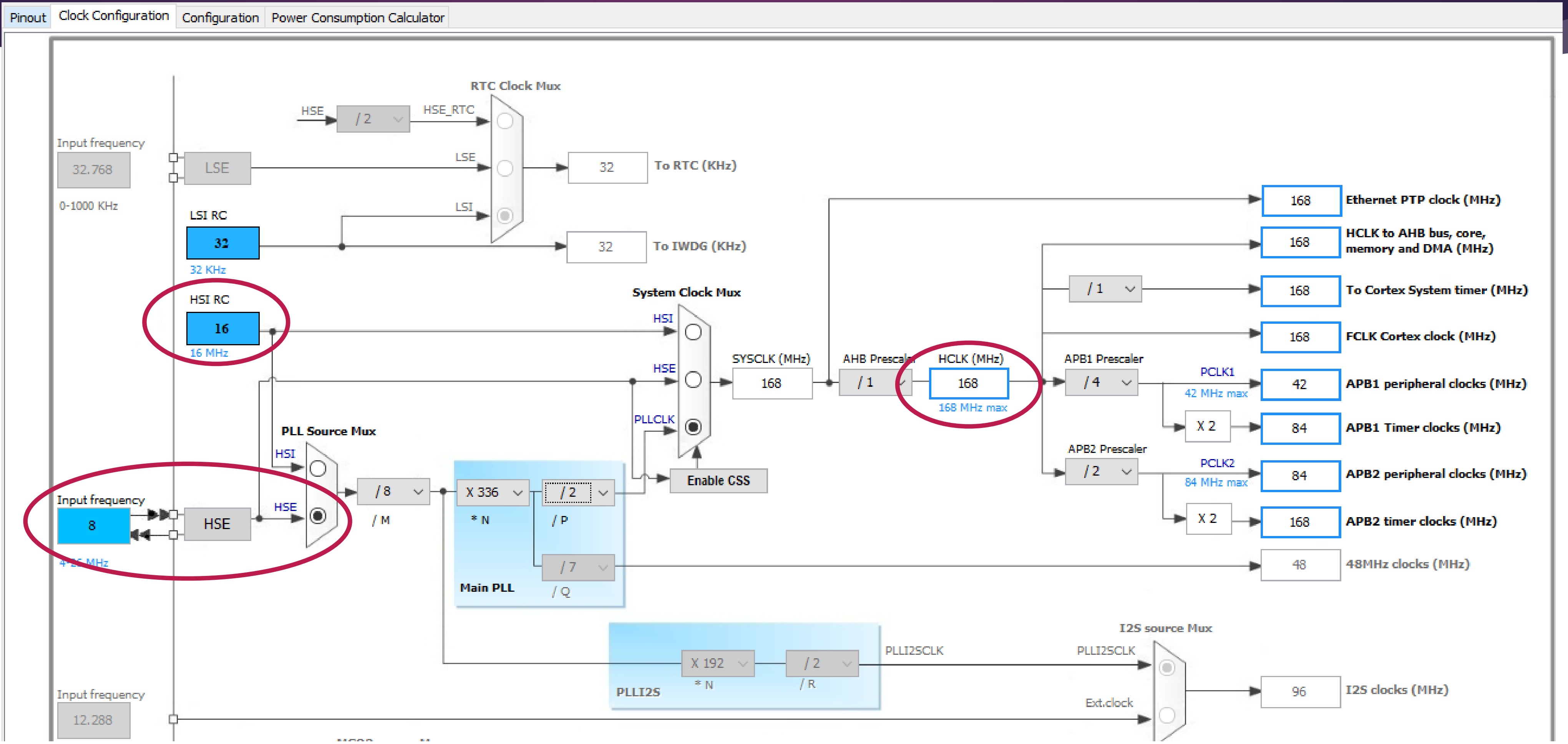
Clock tree

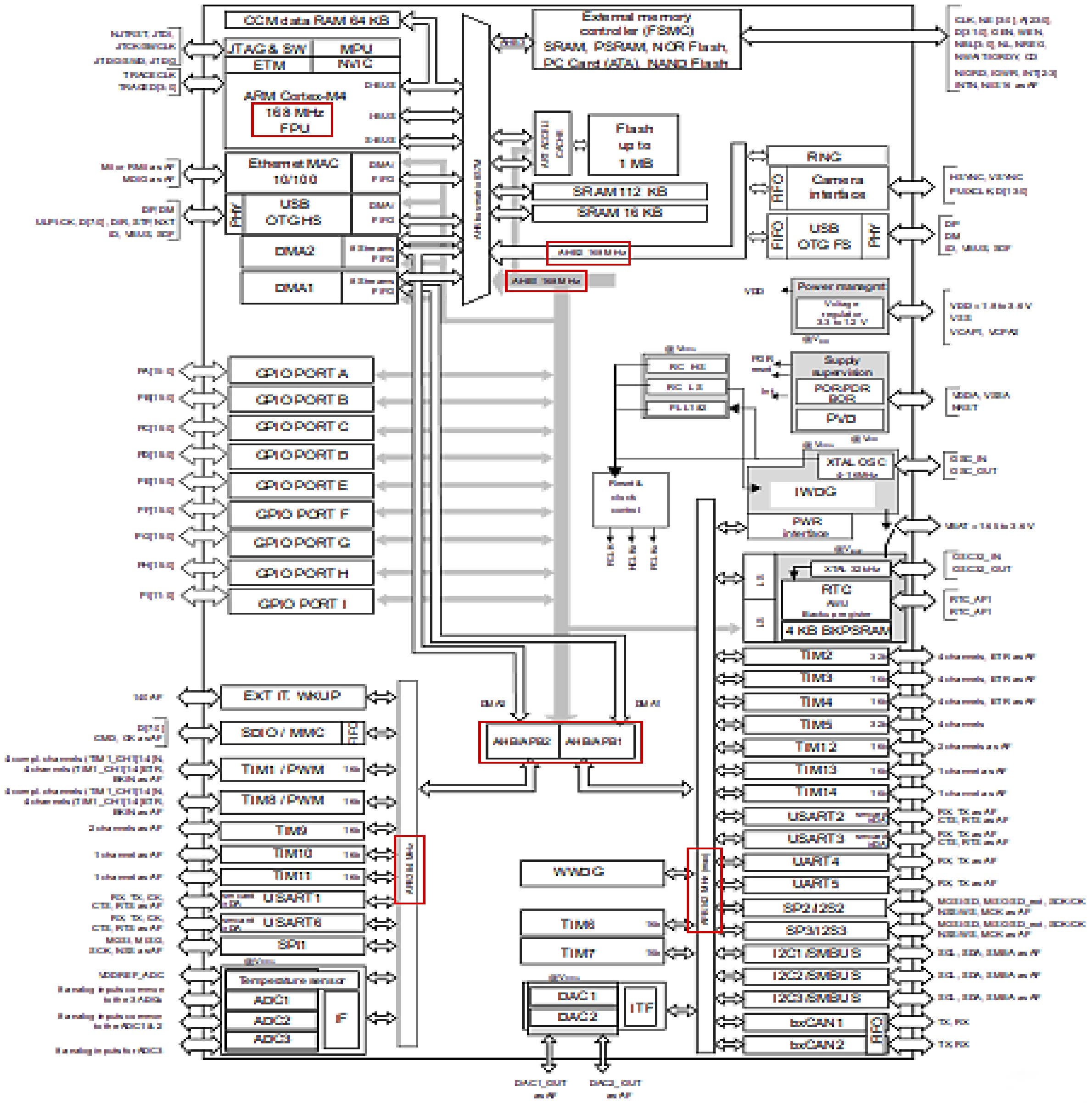
- These micro-controllers have two sources of clock. The internal is based on RC oscillators while the external can be a crystal which is more accurate and faster.
- The discovery board has the external crystal working at 8 MHz.
- To set the clock source as external you have to enable the HSE as Crystal Resonator, under the RCC peripheral.
- To begin working with any peripheral, you need to enable the

clock source of the relevant block, see next slide.



Clock tree

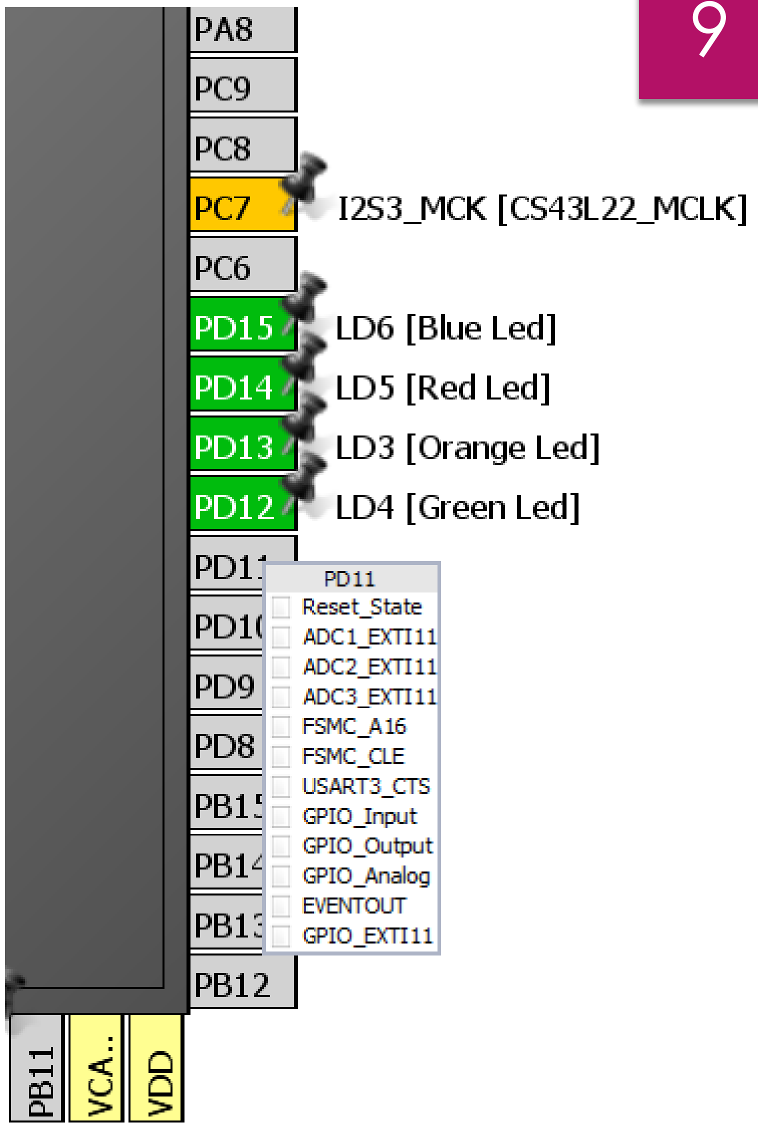

Pinout
Clock Configuration
Configuration
Power Consumption Calculator
RTC Clock Mux
HSE
/ 2
HSE_RTC
Input frequency
32.768
0-1000 KHz
LSE
LSE
LSI
32
To RTC (KHz)
LSI RC
32
32 KHz
32
To IWDG (KHz)
HSI RC
16
16 MHz
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
168
AHB Prescaler
/ 1
HCLK (MHz)
168
168 MHz max
168
Ethernet PTP clock (MHz)
168
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
168
To Cortex System timer (MHz)
168
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 4
PCLK1
42 MHz max
42
APB1 peripheral clocks (MHz)
X 2
84
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 2
PCLK2
84 MHz max
84
APB2 peripheral clocks (MHz)
X 2
168
APB2 timer clocks (MHz)
48
48MHz clocks (MHz)
PLL Source Mux
HSI
HSE
Input frequency
8
4-26 MHz
HSE
/ 8
/ M
X 336
* N
/ 2
/ P
/ 7
/ Q
Main PLL
Enable CSS
X 192
* N
/ 2
/ R
PLLI2S
PLLI2SCLK
I2S source Mux
PLLI2SCLK
Ext.clock
96
I2S clocks (MHz)
Input frequency
12.288



CCM data RAM 64 KB
JTAG & SW
ETM
MPU
NVIC
ARM Cortex-M4
168 MHz
FPU
External memory
controller (FSMC)
SRAM, PSRAM, NOR Flash,
PC Card (ATA), NAND Flash
Flash
up to
1 MB
SRAM 112 KB
SRAM 16 KB
Ethernet MAC
10/100
USB
OTG HS
DMA2
DMA1
RNG
Camera
interface
USB
OTG FS
FIFO
PHY
GPIO PORT A
GPIO PORT B
GPIO PORT C
GPIO PORT D
GPIO PORT E
GPIO PORT F
GPIO PORT G
GPIO PORT H
GPIO PORT I
IWDG
PVD
RTC
4 KB BKPSRAM
EXT IT. WKUP
SDIO / MMC
TIM1 / PWM
TIM8 / PWM
TIM9
TIM10
TIM11
USART1
USART6
SPI1
Temperature sensor
ADC1
ADC2
ADC3
WWDG
TIM6
TIM7
DAC1
DAC2
ITF
TIM2
TIM3
TIM4
TIM5
TIM12
TIM13
TIM14
USART2
USART3
UART4
UART5
SPI2/I2S2
SPI3/I2S3
I2C1/SMBUS
I2C2/SMBUS
I2C3/SMBUS
bxCAN1
bxCAN2

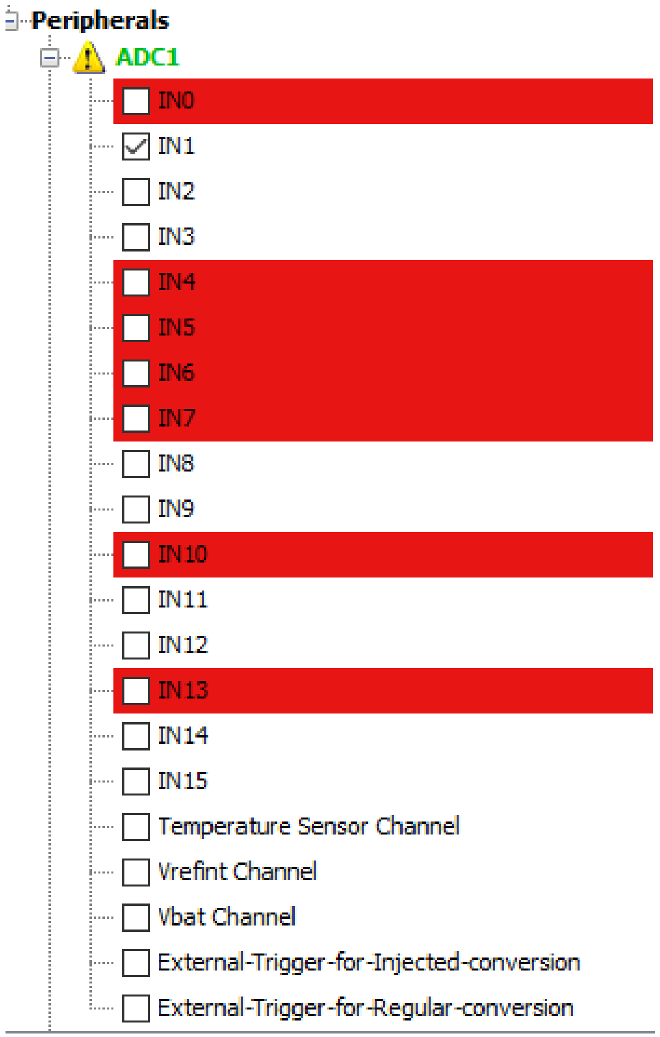
GPIO settings

- Main GPIO settings are Input and Output modes.
- Outputs are Open-Drain and Push-Pull.
- The "Input" is different than "Analog", as the later accepts all ranges of voltages within 0 to 3.3v.



Analog to Digital Convertor

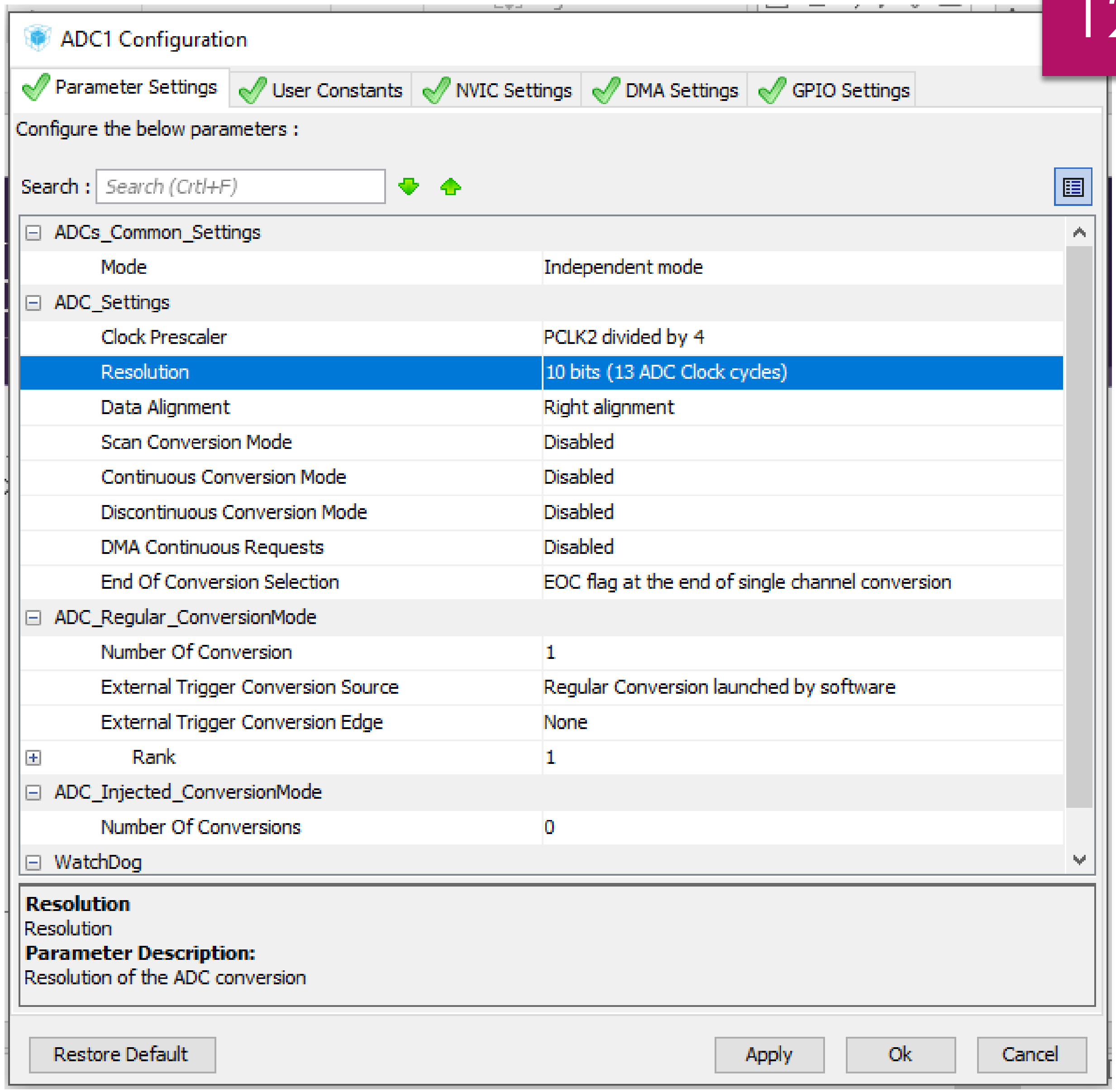
- Each ADC block is able to read and convert 16 external channels.
- ADC1 is on GPIO-A.
- Once you select the IN1, the GPIOA-1 will turn into GPIO_Analog



How the ADC work

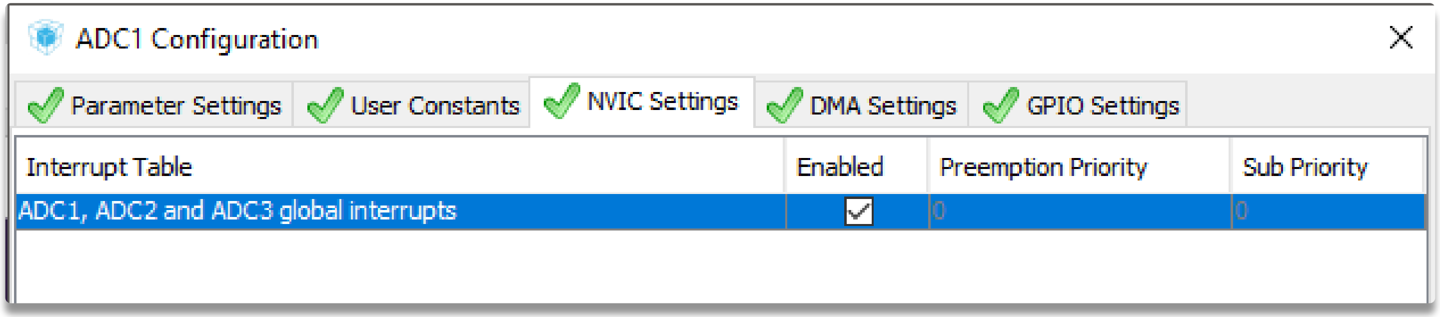
- The ADC blocks are based on SAR architecture. For more information you are encouraged to study more about it.
- To begin:
  - you should first enable its clock source and prescaler,
  - Select the channel number,
  - Conversion resolution (6,8,10,12 bits),
  - Conversion modes (continues, or single),
  - And data access mode (DMA or manual reading)

- When you choose the Ext trigger source as software, that means the user requests a new conversion by calling the HAL_ADC_START_xx().
- Other options are internal or external sources such as Timers or a GPIO pin.



ADC in interrupt mode

- By enabling the interrupt, you will be informed about the end of conversion using interrupt calls.
- Once the conversion is done, the function ADC_IRQHandler in the stm32f4xx_it.c will be called.
- To read the converted value, you should use HAL_ADC_GetValue(). Then you need to convert the digital value into the real analog voltage using a simple equation.
- If you do not use interrupts, then you have to wait until the end of conversion. This technique is called polling mode.
- In this assignment you need to call the conversion by HAL_ADC_Start_IT() function wherever the SysTick interrupt handler will be called.



Interrupts and NVIC (Overview)

- NVIC stands for Nested Vector Interrupt Controller. Why is it called like this?
- Supports up to 240 Programmable Interrupts (could be from SoC internal peripheral sources (TIMx,ADC) or external sources (i.e. MEMS sensors through GPIO)).
- The priorities are simply assigned by

numbers. Lower numbers have the higher priority.

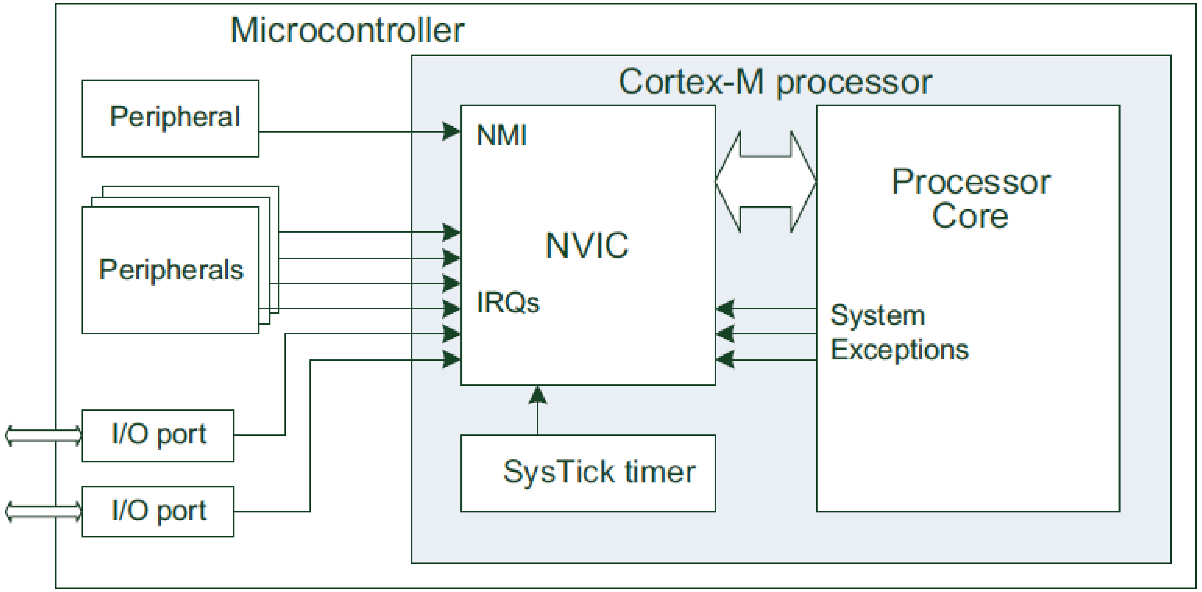


Table 7.3 CMSIS-Core Exception Definitions

Exception Number	Exception Type	CMSIS-Core Enumeration (IRQn)	CMSIS-Core Enumeration Value	Exception Handler Name
1	Reset	-	-	Reset_Handler
2	NMI	NonMaskableInt_IRQn	–14	NMI_Handler
3	Hard Fault	HardFault_IRQn	–13	HardFault_Handler
4	MemManage Fault	MemoryManagement_IRQn	–12	MemManage_Handler
5	Bus Fault	BusFault_IRQn	–11	BusFault_Handler
6	Usage Fault	UsageFault_IRQn	–10	UsageFault_Handler
11	SVC	SVCall_IRQn	–5	SVC_Handler
12	Debug Monitor	DebugMonitor_IRQn	–4	DebugMon_Handler
14	PendSV	PendSV_IRQn	–2	PendSV_Handler
15	SYSTICK	SysTick_IRQn	–1	SysTick_Handler
16	Interrupt #0	(device-specific)	0	(device-specific)
17	Interrupt #1 - #239	(device-specific)	1 to 239	(device-specific)

Interrupts and NVIC (cont.)

- The peripheral asserts an interrupt request to the processor,
- The processor suspends the current task which is running,
- The processor executes an Interrupt Service Routine(ISR) to service the peripheral, and then clear the interrupt request (a flag) by software if needed.
- The processor backs to the previously suspended task and resumes it.
- To set the interrupt of a peripheral block you should set the priority and then enable it:

  HAL_NVIC_SetPriority(XXX_IRQn, Y, X);   (order: group = Y, individual = X)

  HAL_NVIC_EnableIRQ(XXX_IRQn);

Interrupts and NVIC (cont.)

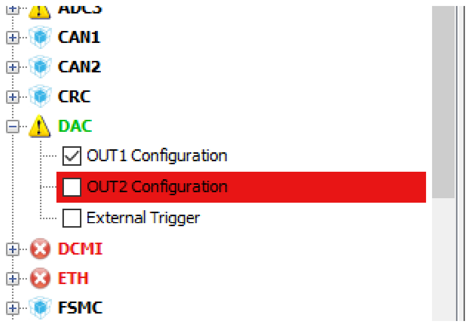
- The body of your ISR/Callback should be minimal. Why?
- Exceptions and Interrupts run in a privileged state, could be dangerous.
- For periodic interrupts, make sure that the duration of the ISR body is less than the interrupt period.

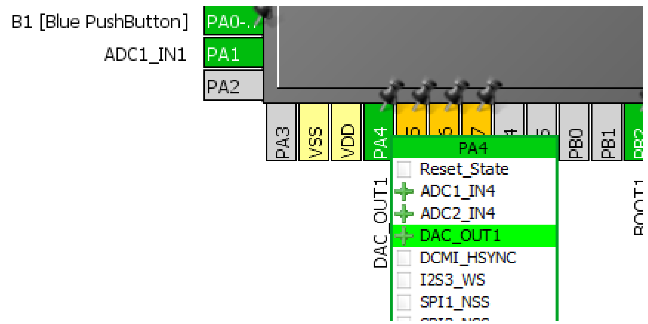
SysTick Timer

- SysTick is a 24-bit hardware timer in the ARM processor core. This is in different than other TIMx modules provided by ST or other companies in the SoCs.
- In the STM32F4 CUBE firmware, SysTick is enabled by default when the HAL_Init() is called. The base frequency is configured to **1 ms** (or 1 KHz). You may need to change this frequency. **How?**
- The function to set SysTick frequency is **HAL_SYSTICK_Config ()**
- This timer counts from 0 to 2^24-1. To set the frequency, you must pass the desired number to the above function.
- This timer has an ISR body in the _it.c file. Its interrupt priority is configurable.

Digital to Analog Converter

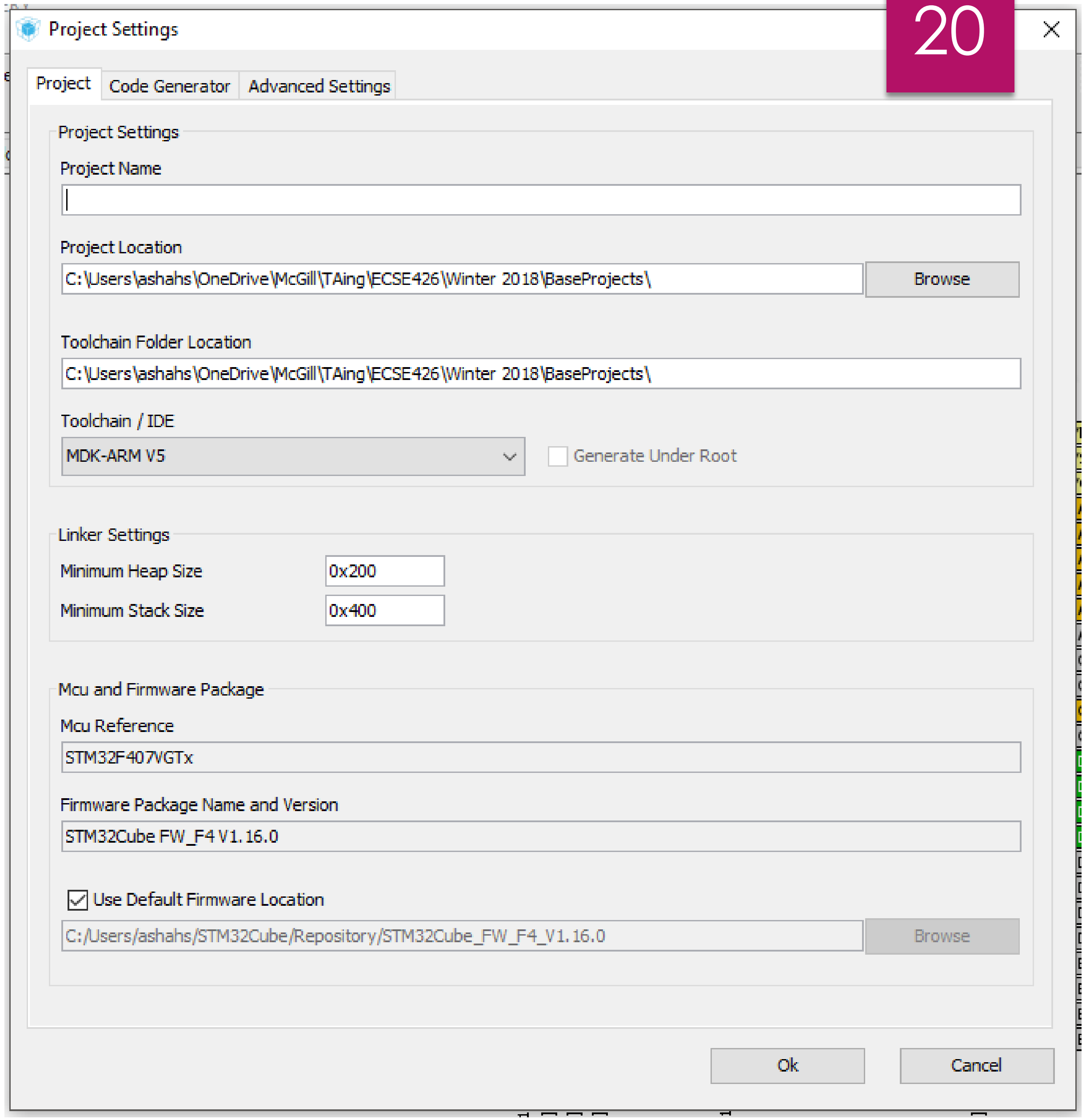
- This micro-controller has only one DAC block with 2 channel outputs.
- It has only 8 and 12 bits resolutions.
- The two output pins of this block are connected to a headphone and speaker amplifier on the board.
- To use one of them, you have to click on PA.4 and choose DAC_OUT1.
- Use **HAL_DAC_Start()** and **HAL_DAC_SetValue()** to start your DAC block operation.





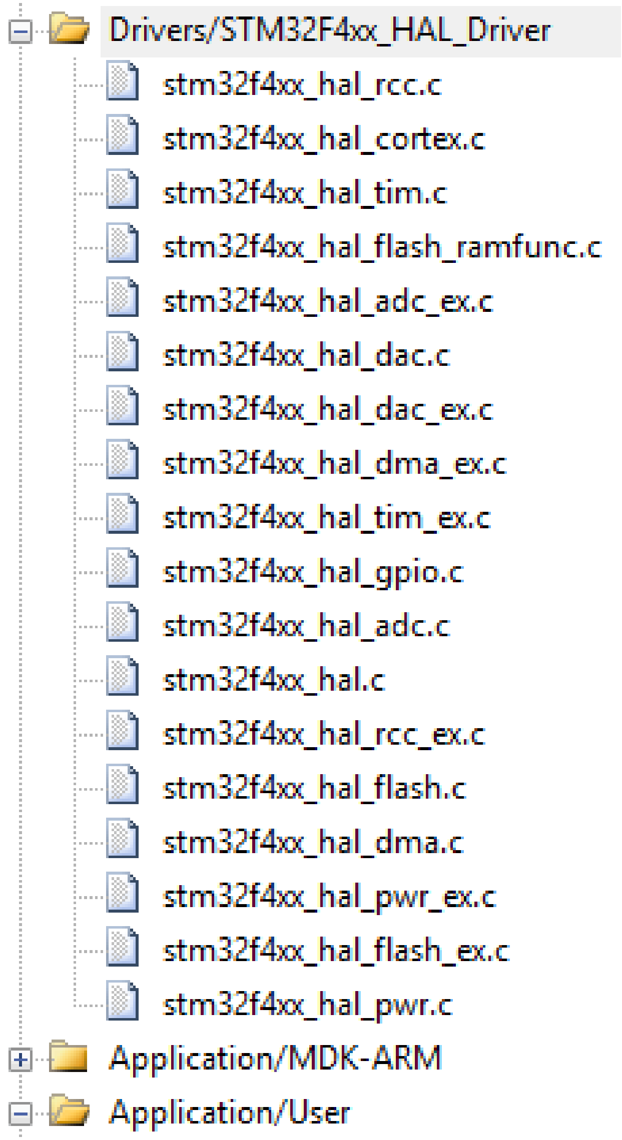
Generating project files

- By selecting MDK-ARM V5 from project setting, you can generate the base project for Keil software.
- Name the project and then OK.
- Click on Generate Code from project menu.
- The software will start generating codes and open the Keil.



STM32Cube Organisation

- Base peripheral drivers only have one C file (driver implementation file), and header file (API, parameters, declarations).
- Other peripherals have a basic driver and an extended driver (with added _ex suffix),
- you will find a listing of your drivers. Only after you compile your project that the “+” signs will show up enabling you to see the header files when you expand them,
- STM32F4xx_hal.c (Essential for HAL initialization, HAL Delays, many drivers depend on this file).



Essential points

- If you want to use the "Printf()" function, you have to creat a **fput_debug.c** file into your project, and copy the content of this file you had in LAB1.
- You also need to change few more things in the "Options for the Target".
- The core clock should be the value you set as the HCLK.
- In this menu, ensure that the ST-LinkV/2 is the debugger.