

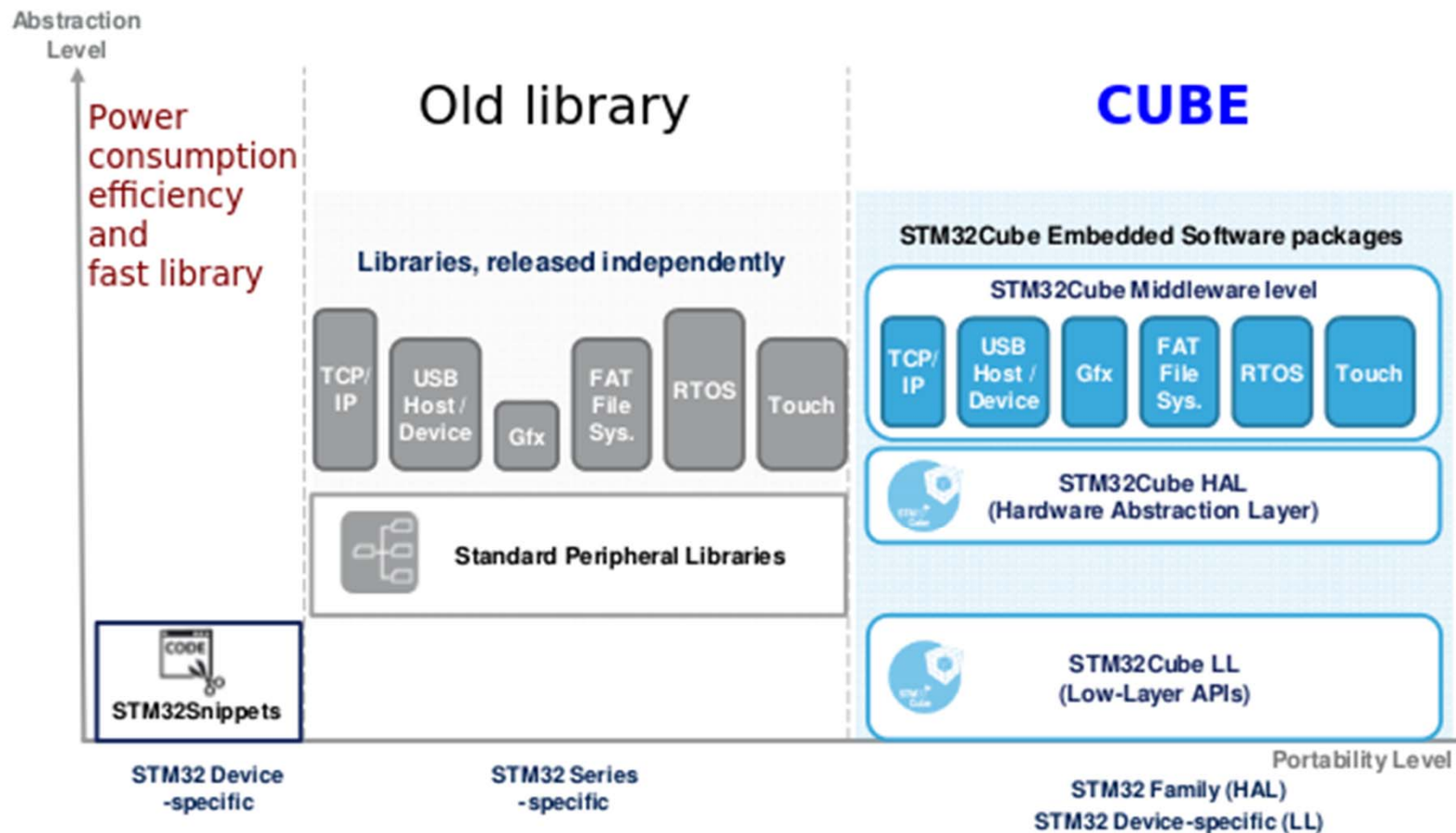
# Hardware Abstraction & Low Level Layers for STM32

## Problem to solve

- What is the best way to program a complex microprocessor-based system and its peripherals?
  - Directly using assembler (complex and device dependent)
  - Using code provided by the manufacturer for a specific device/model
  - Using software libraries with different optimizations levels
    - HAL and LL libraries are examples of this (STM32 CUBE)
  - In our course, we have decided to use the HAL provided by ST Microelectronics because it is the same library for all the 32-bit families



# Three different approaches

## ST Embedded software offer - Positioning

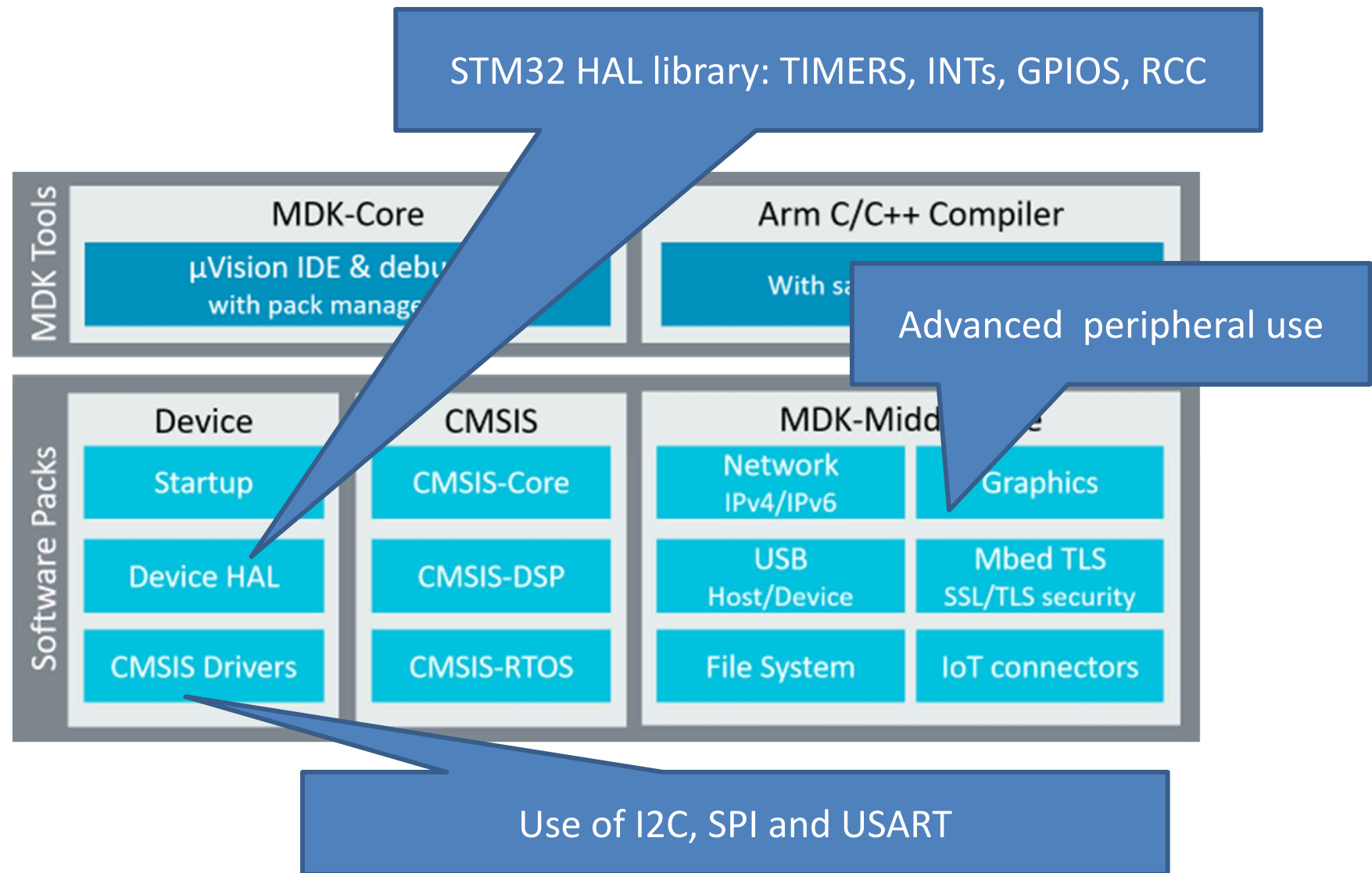


# In Microprocessor Based Systems we will use the HAL API

## ST Embedded software offer – Comparison

Offer		Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
 STM32Snippets			+++			+
 Standard Peripheral Library		++	++	+	++	+++
 STM32 Cube	HAL API	+++	+	++	+++	+++
	LL APIs		+++			++

# Development of applications for ARM 32 devices using KEIL-MDK



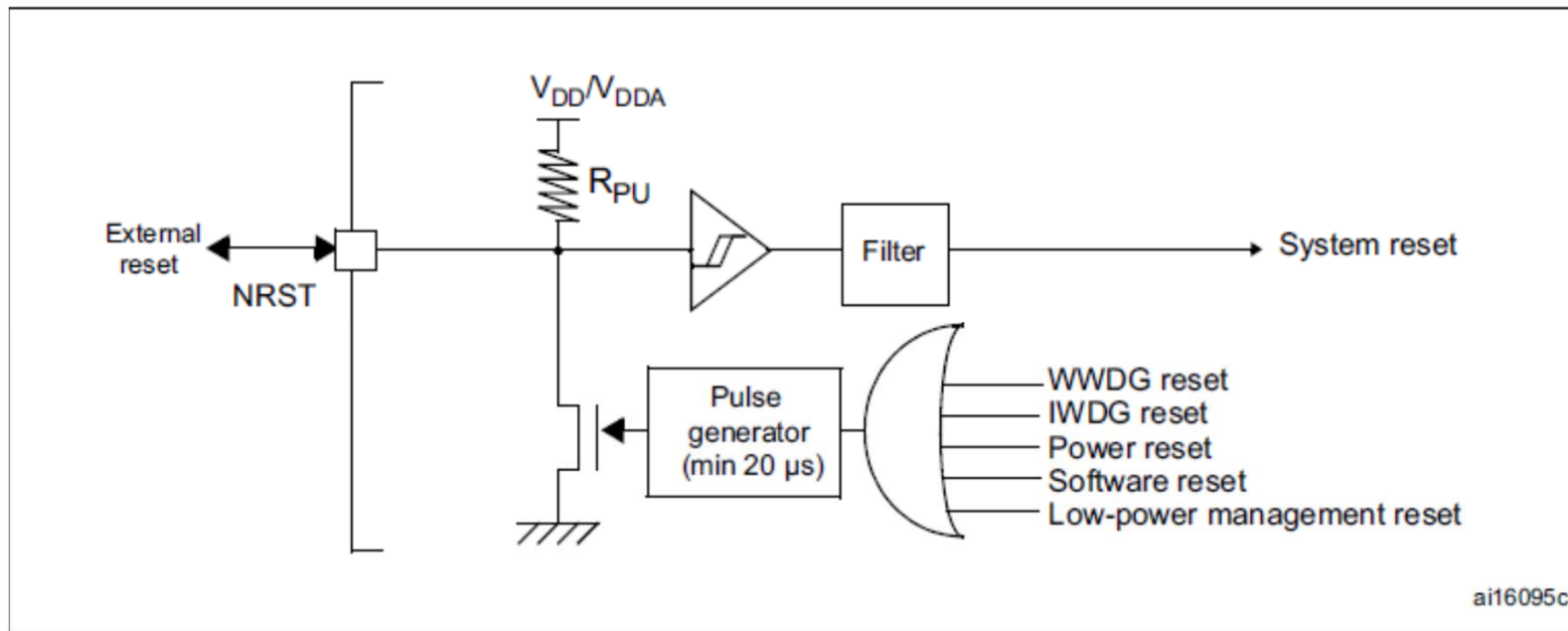
# Software development in SBM HAL

- Keil Microvision for editing, compiling, linking, and debugging
- STM-32 HAL for configuring:
  - clocks and reset circuits (HAL RCC)
  - GPIO
  - Timers
  - Interrupts (NVIC and EXTI)
- CMSIS-Drivers for:
  - I2C and SPI
  - USART

# RCC: Reset and Clock Control

# Reset circuit

- Reset sources acts on NRST pin.
- Reset service routine is fixed at address 0x00000004



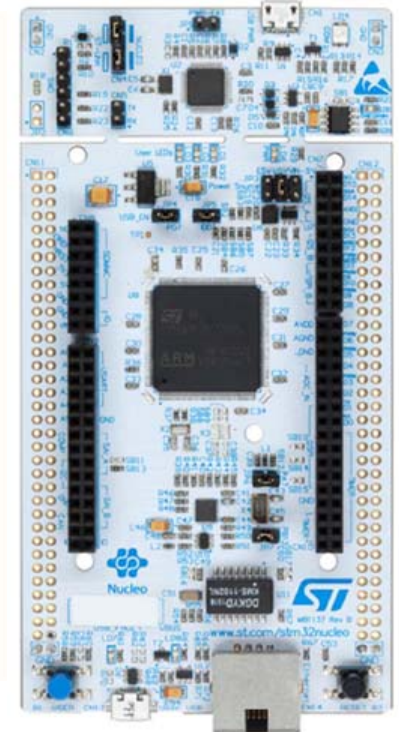
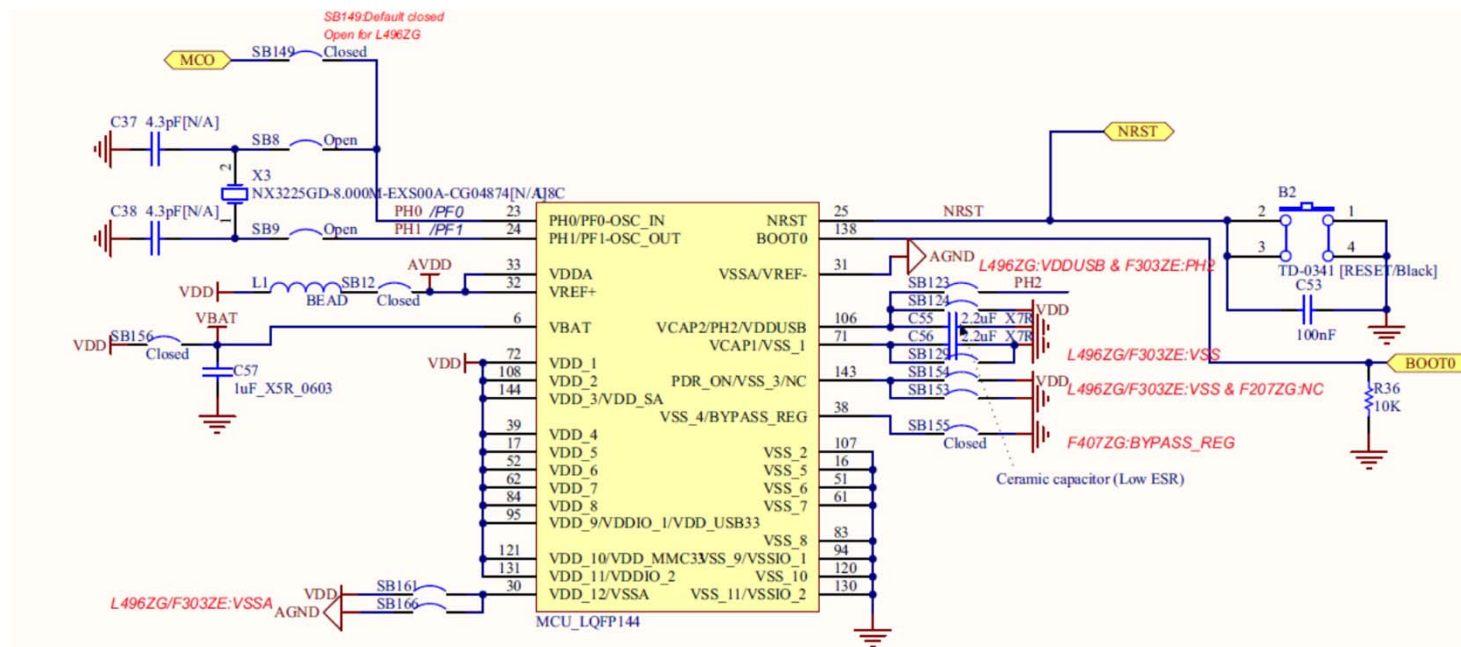


# Reset circuit (3 types of reset sources)

- System Reset
  - Low level on external pin NRST
  - Window watchdog end of count condition (WWDG). Counter that must be refreshed within a specific time window. If not, the watchdog generates a system reset
  - Independent watchdog end of count (IWDG). Triggers a system reset if a counter reaches a given timeout value (it uses the RC oscillator)
  - Software reset
  - Low power management reset
    - Reset when entering standby mode
    - Reset when entering in Stop mode
- Power reset
  - Power-on/down reset (POR/PDR), brownout reset (BOR)
- Backup domain reset

# NUCLEO-144 with STM32F429ZI

- B2 Button: generates reset on NRST signal
- Connection detail in NUCLEO-144 with STM32F429ZI (144 pins)



# Creation of a default project with Keil Microvision

- Run Keil Microvision
- New Project
- Select the ST device
- Select the minimum software elements (next slide)
- Press OK
- Inspect the project created

Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support		NUCLEO-F429ZI	1.0.0	STMicroelectronics NUCLEO-F429ZI Development Board
CMSIS				<a href="#">Cortex Microcontroller Software Interface Components</a>
CORE	<input checked="" type="checkbox"/>		5.5.0	<a href="#">CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M</a>
DSP	<input type="checkbox"/>	Source	1.9.0-dev	<a href="#">CMSIS-DSP Library for Cortex-M, SC000, and SC300</a>
NN Lib	<input type="checkbox"/>		3.0.0	<a href="#">CMSIS-NN Neural Network Library</a>
RTOS (API)			1.0.0	<a href="#">CMSIS-RTOS API for Cortex-M, SC000, and SC300</a>
RTOS2 (API)			2.1.3	<a href="#">CMSIS-RTOS API for Cortex-M, SC000, and SC300</a>
CMSIS Driver				<a href="#">Unified Device Drivers compliant to CMSIS-Driver Specifications</a>
Compiler		ARM Compiler	1.6.0	<a href="#">Compiler Extensions for ARM Compiler 5 and ARM Compiler 6</a>
Device				<a href="#">Startup, System Setup</a>
Startup	<input checked="" type="checkbox"/>		2.6.3	System Startup for STMicroelectronics STM32F4 Series
STM32Cube Framework (API)			1.1.0	<a href="#">STM32Cube Framework</a>
Classic	<input checked="" type="checkbox"/>		1.7.9	<a href="#">Configuration via RTE_Device.h</a>
STM32CubeMX	<input type="checkbox"/>		1.0.0	<a href="#">Configuration via STM32CubeMX</a>
STM32Cube HAL				<a href="#">STM32F4xx Hardware Abstraction Layer (HAL) Drivers</a>
ADC	<input type="checkbox"/>		1.7.9	Analog-to-digital converter (ADC) HAL driver
CAN	<input type="checkbox"/>		1.7.9	Controller area network (CAN) HAL driver
CRC	<input type="checkbox"/>		1.7.9	CRC calculation unit (CRC) HAL driver
Common	<input checked="" type="checkbox"/>		1.7.9	Common HAL driver
Cortex	<input checked="" type="checkbox"/>		1.7.9	Cortex HAL driver
DAC	<input type="checkbox"/>		1.7.9	Digital-to-analog converter (DAC) HAL driver
DCMI	<input type="checkbox"/>		1.7.9	Digital camera interface (DCMI) HAL driver
DMA2D	<input type="checkbox"/>		1.7.9	Chrom-Art Accelerator (DMA2D) HAL driver
DMA	<input type="checkbox"/>		1.7.9	DMA controller (DMA) HAL driver
ETH	<input type="checkbox"/>		1.7.9	Ethernet MAC (ETH) HAL driver
EXTI	<input type="checkbox"/>		1.7.9	External interrupts and event controller (EXTI) HAL driver
Flash	<input type="checkbox"/>		1.7.9	Embedded Flash memory (Flash) HAL driver
GPIO	<input checked="" type="checkbox"/>		1.7.9	General-purpose I/O (GPIO) HAL driver
HCD	<input type="checkbox"/>		1.7.9	USB Host controller (HCD) HAL driver
I2C	<input type="checkbox"/>		1.7.9	Inter-integrated circuit (I2C) HAL driver
I2S	<input type="checkbox"/>		1.7.9	I2S HAL driver
IRDA	<input type="checkbox"/>		1.7.9	IrDA HAL driver
IWDG	<input type="checkbox"/>		1.7.9	Independent watchdog (IWDG) HAL driver
LTDC	<input type="checkbox"/>		1.7.9	LCD-TFT Controller (LTDC) HAL driver
MMC	<input type="checkbox"/>		1.7.9	Multi Media Card (MMC) HAL driver
NAND	<input type="checkbox"/>		1.7.9	NAND Flash controller (NAND) HAL driver
NOR	<input type="checkbox"/>		1.7.9	NOR Flash controller (NOR) HAL driver
PC Card	<input type="checkbox"/>		1.7.9	PC Card controller (PC Card) HAL driver
PCD	<input type="checkbox"/>		1.7.9	USB Peripheral controller (PCD) HAL driver
PWR	<input checked="" type="checkbox"/>		1.7.9	Power controller (PWR) HAL driver
RCC	<input checked="" type="checkbox"/>		1.7.9	Reset and clock control (RCC) HAL driver
RNG	<input type="checkbox"/>		1.7.9	Random number generator (RNG) HAL driver
RTC	<input type="checkbox"/>		1.7.9	Real-time clock (RTC) HAL driver

Validation Output

Description

Resolve Select Packs Details OK Cancel

Elements to select

## Project

Project: rccbasic

Target 1

Source Group 1

CMSIS

Device

stm32f4xx\_hal.c (STM32Cube HAL:Common)

stm32f4xx\_hal\_cortex.c (STM32Cube HAL:Cortex)

stm32f4xx\_hal\_gpio.c (STM32Cube HAL:GPIO)

stm32f4xx\_hal\_pwr.c (STM32Cube HAL:PWR)

stm32f4xx\_hal\_pwr\_ex.c (STM32Cube HAL:PWR)

stm32f4xx\_hal\_rcc.c (STM32Cube HAL:RCC)

stm32f4xx\_hal\_rcc\_ex.c (STM32Cube HAL:RCC)

RTE\_Device.h (STM32Cube Framework:Classic)

startup\_stm32f429xx.s (Startup)

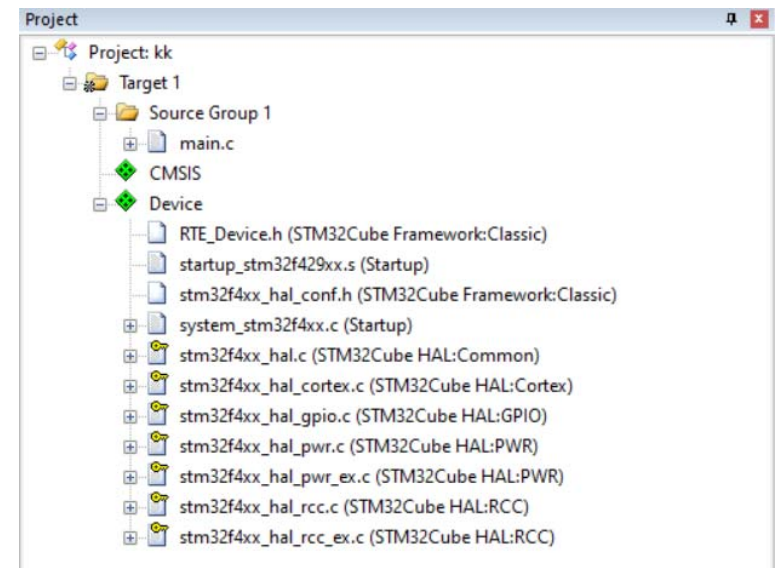
stm32f4xx\_hal\_conf.h (STM32Cube Framework:Classic)

system\_stm32f4xx.c (Startup)

# Reset handler (in ARM assembly)

- See file startup\_stm32f429xx.s
- After a hardware reset the Cortex Microprocessor starts the execution of the Reset Handler. The address of this handler is stored at address 0x00000004
- It calls first to “**SystemInit**” function and then calls “**\_\_main**” (the C main() function)
- SystemInit is defined in “**system\_stm32f4xx.c**” file (startup package)

```
; Reset handler
Reset_Handler PROC
                    EXPORT Reset_Handler [WEAK]
IMPORT SystemInit
IMPORT __main
    LDR R0, =SystemInit
    BLX R0
    LDR R0, =__main
    BX R0
ENDP
```



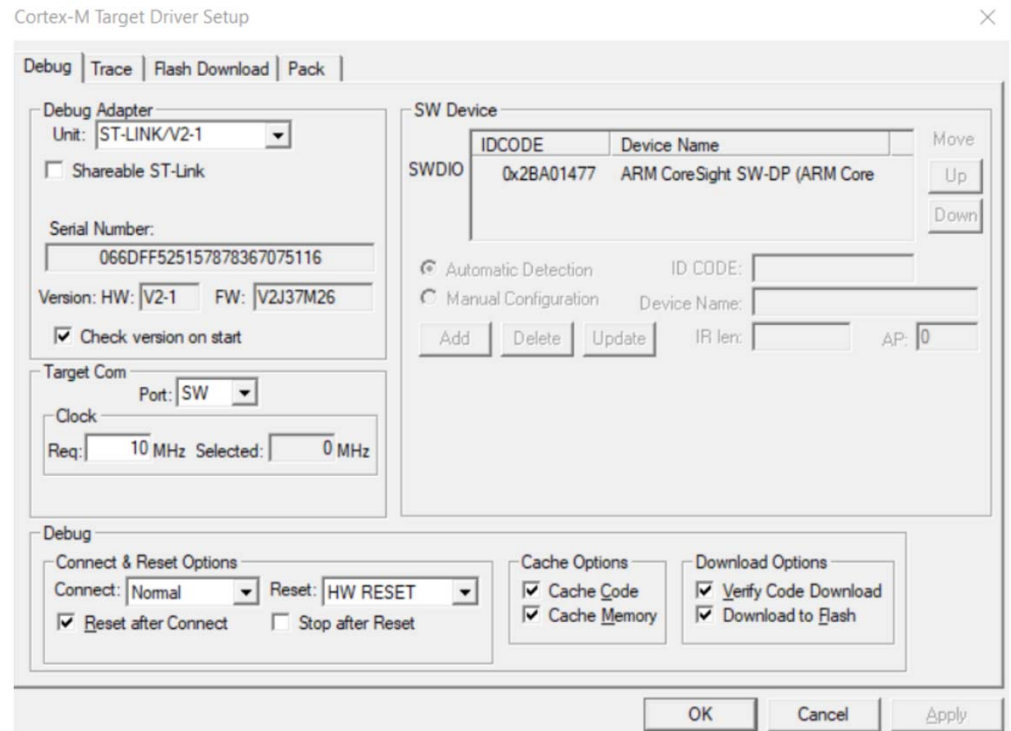
# Reset Handler

- During the execution of the reset handler
  - The processor is using the internal clock
  - The different PINs are in the default state
- The user/developer has to configure the processor and the peripherals. This should be done in the main function
  - the first function to call in the main is “HAL\_Init()” to gain access to the HAL Library



# (Keil) Debug Connect Options

- Normal: stops CPU at the current executed instructions after connecting
- With Pre-reset: applies a hardware reset before connecting the device
- Under Reset: holds the hardware reset active while connecting to the device
- Without stop: connects and disconnects without explicitly stopping the CPU



## (Keil) Debug Reset options

- Reset after connect: enables or disables the operation selected in the Reset drop-down list
- HW Reset: asserts the hardware reset signal
- SYSRESETREQ: performs a software reset (Cortex M and peripherals are reset)
- VECTRESET: Set the VECTRESET bit and only the cortex M is reset

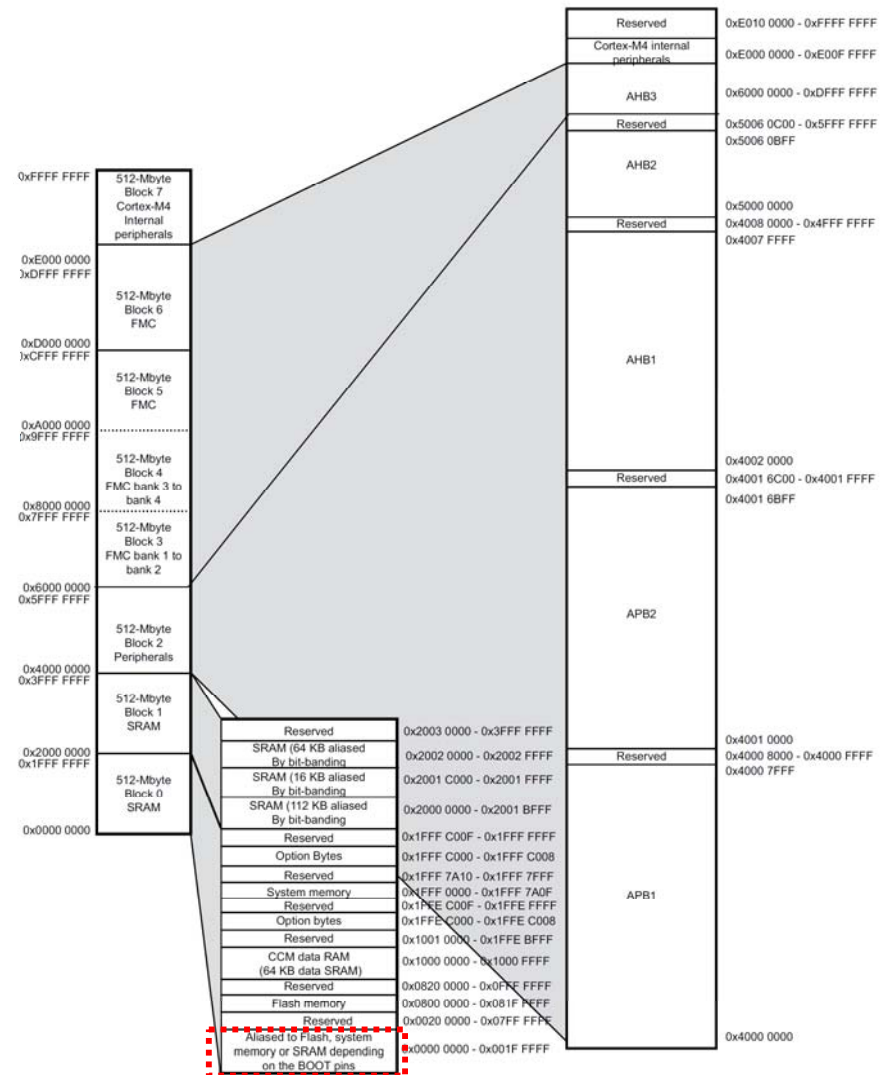


# Memory MAP and boot mode

Table 2. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

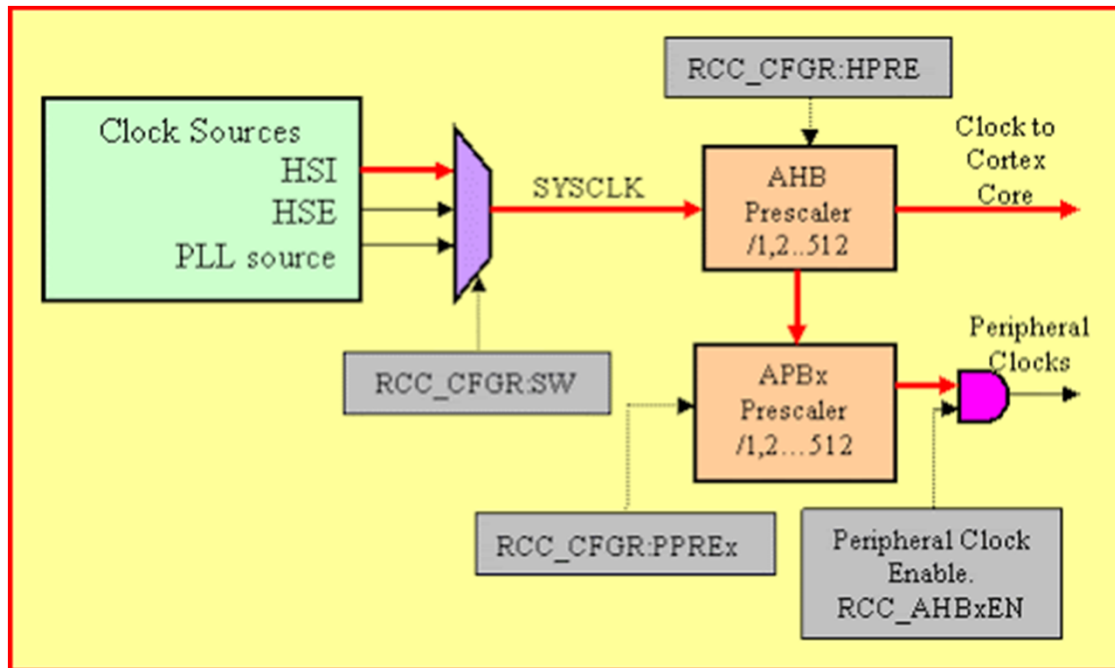
- NUCLEO 144:  
Default BOOT0=0



MS30424V4

# RCC: Clock Control

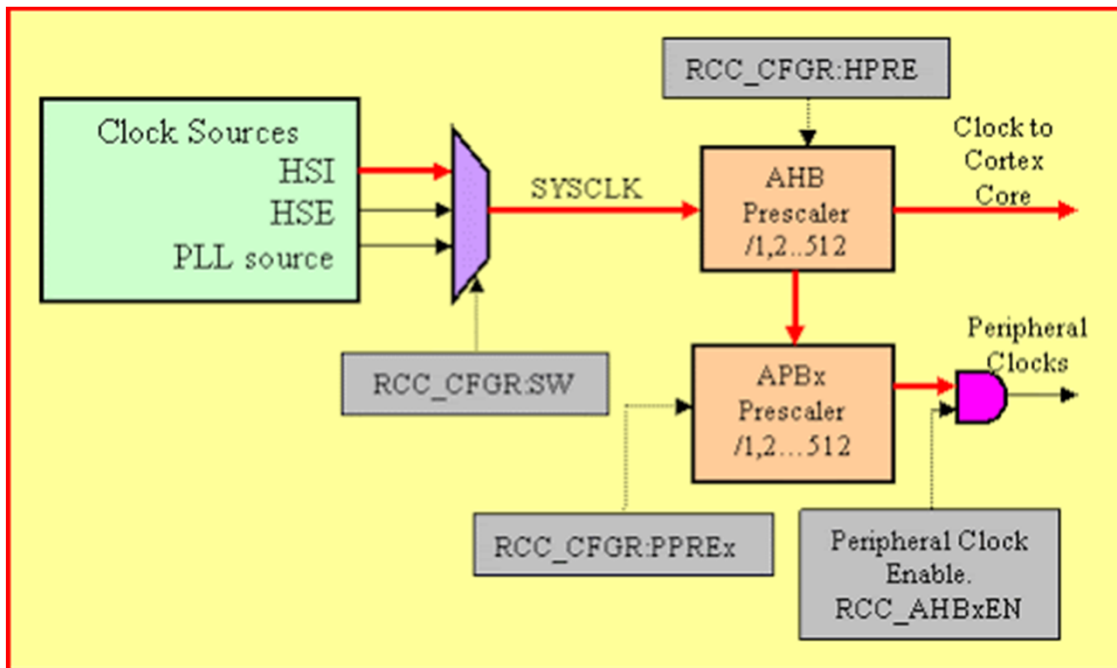
# Basic clock circuitry for the STM32Fxxx



- The STM32F microcontroller system clock can come from one of three sources:
  - The high-speed internal clock (HSI)
  - The high-speed external clock (HSE)
  - The phase locked loop (PLL) clock
- The `RCC_CR` (CLOCK CONTROL) and `RCC_CFGR` (CLOCK CONFIGURATION) registers are used to select and enable the clock source

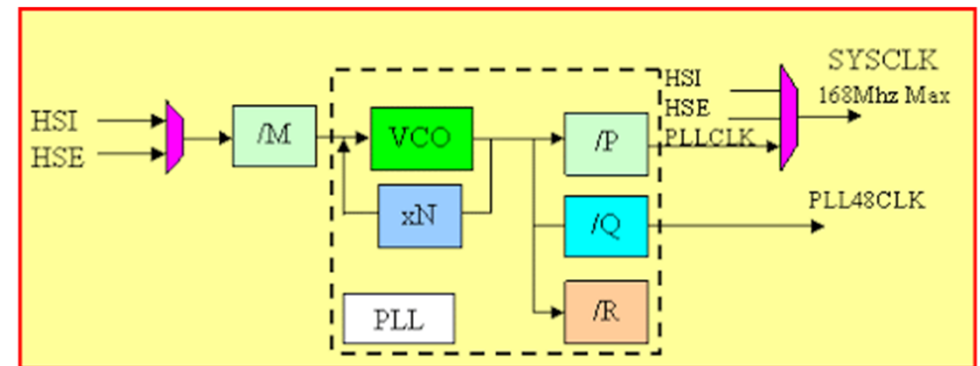
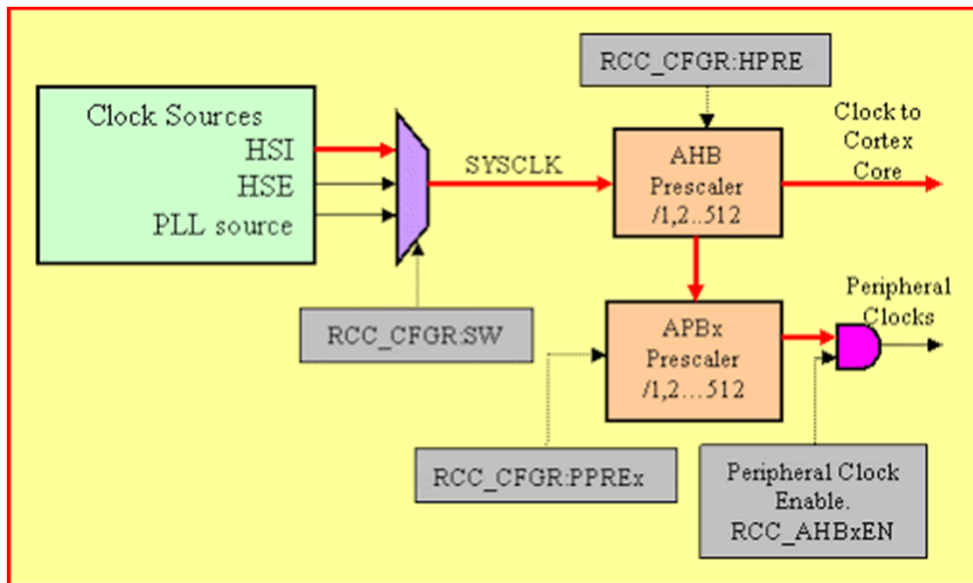
# Basic clock circuitry for the STM32Fxxx

- After resetting the HSI (High-speed internal clock) is enabled
- HSE (High-speed external clock)

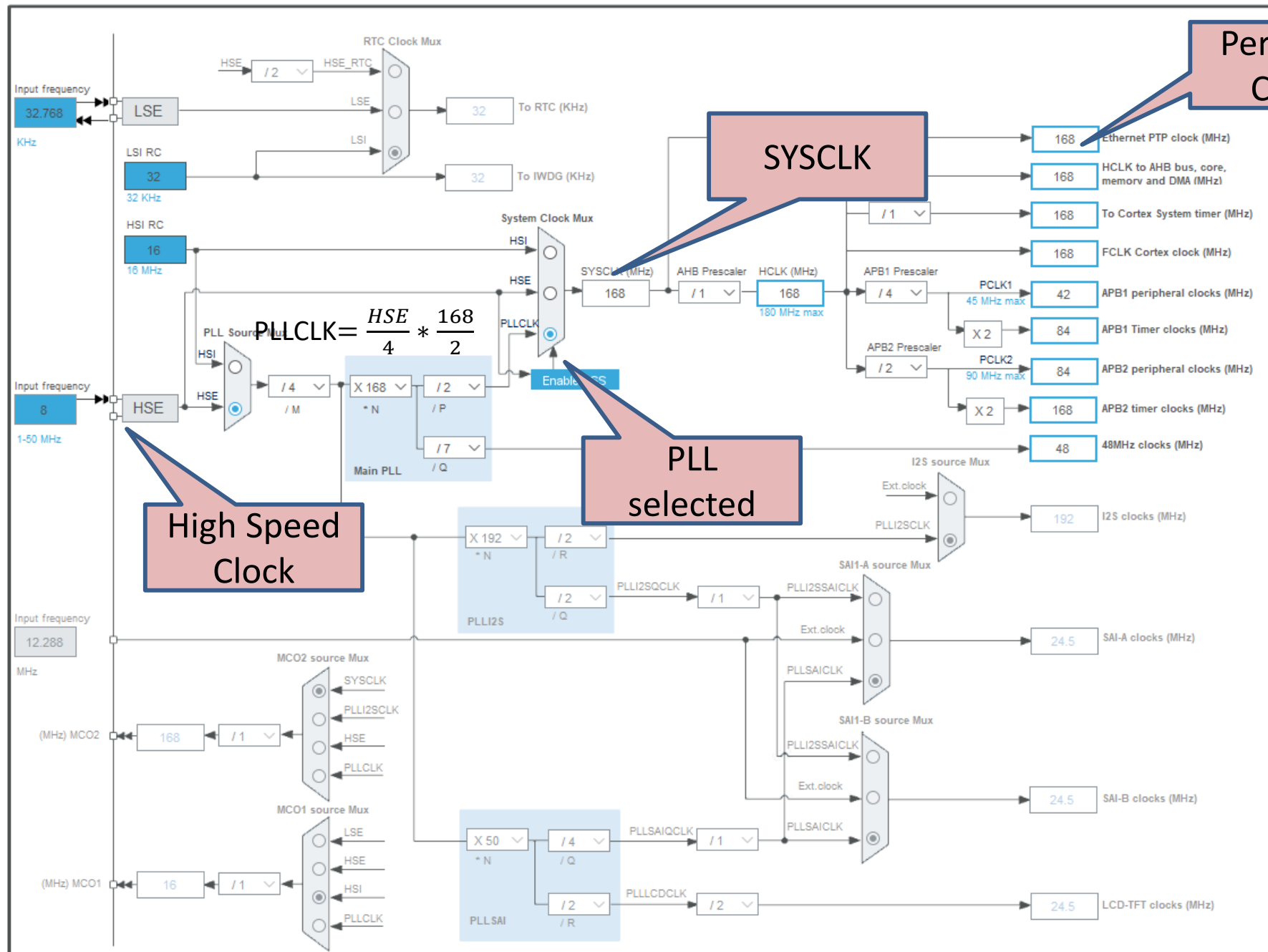


# Basic clock circuitry for the STM32Fxxx

- Bus prescalers and peripheral clocks
- Using the Phase Locked loop

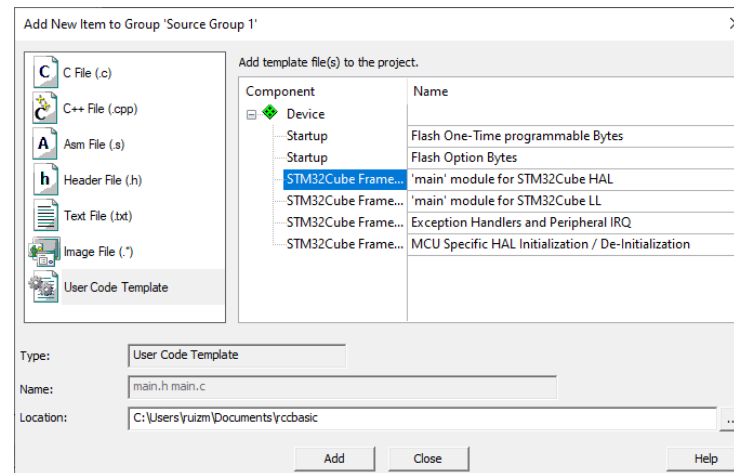


Peripheral  
Clocks



# In the Keil Microvision Project

- In “Source Group 1”->Add New Item (User Code Template)
- Select Device->'main' module for STM32Cube HAL->Add



- Display the content of “main.c”
- Inspect the content of SystemClock\_Config function

# Clock configuration I

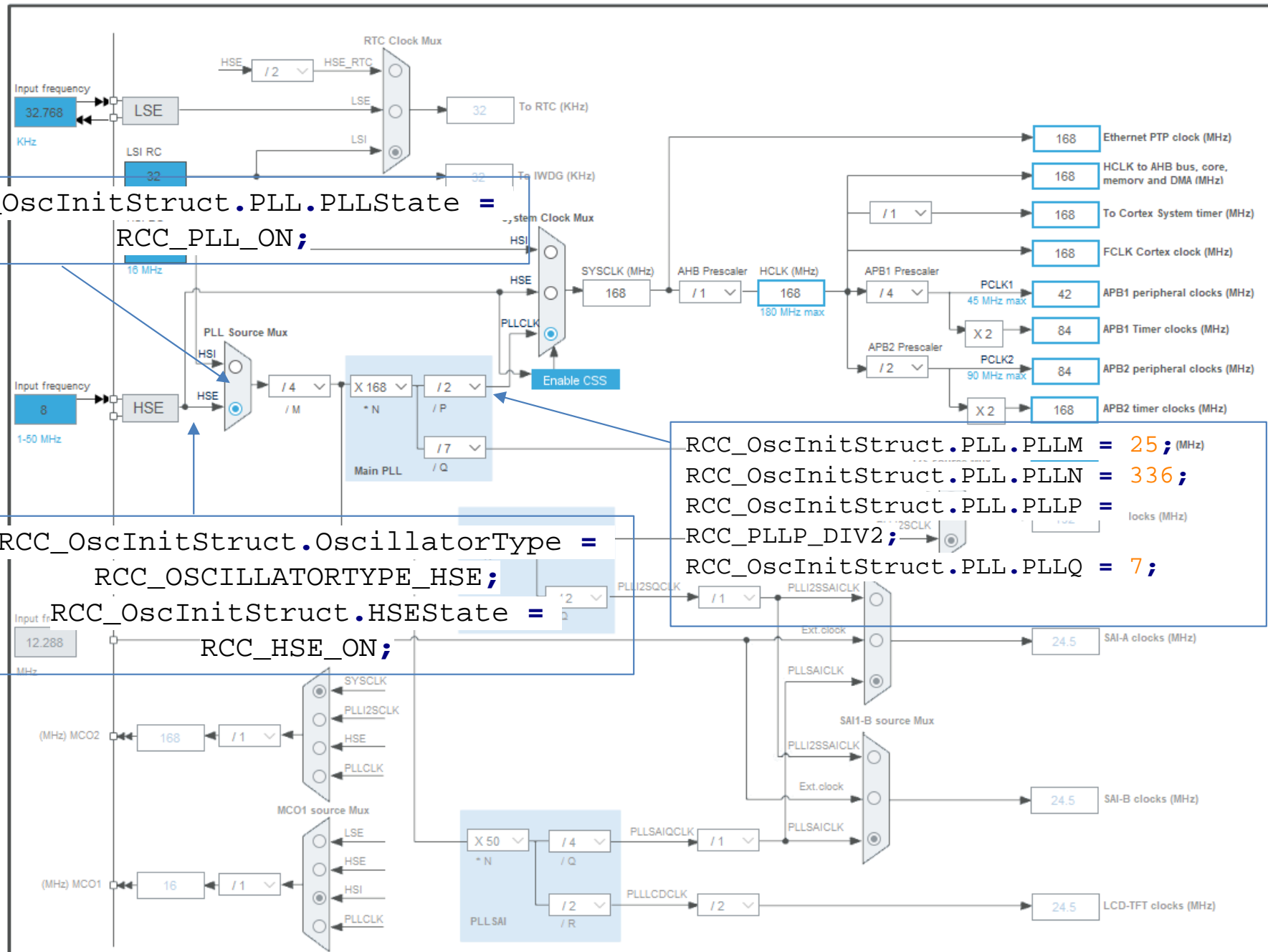
```
static void SystemClock_Config(void)
{
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    RCC_OscInitTypeDef RCC_OscInitStruct;

    /* Enable Power Control clock */
    __HAL_RCC_PWR_CLK_ENABLE();

    .....

    /* Enable HSE Oscillator and activate PLL with HSE as source
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 25;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;
```





```

if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}

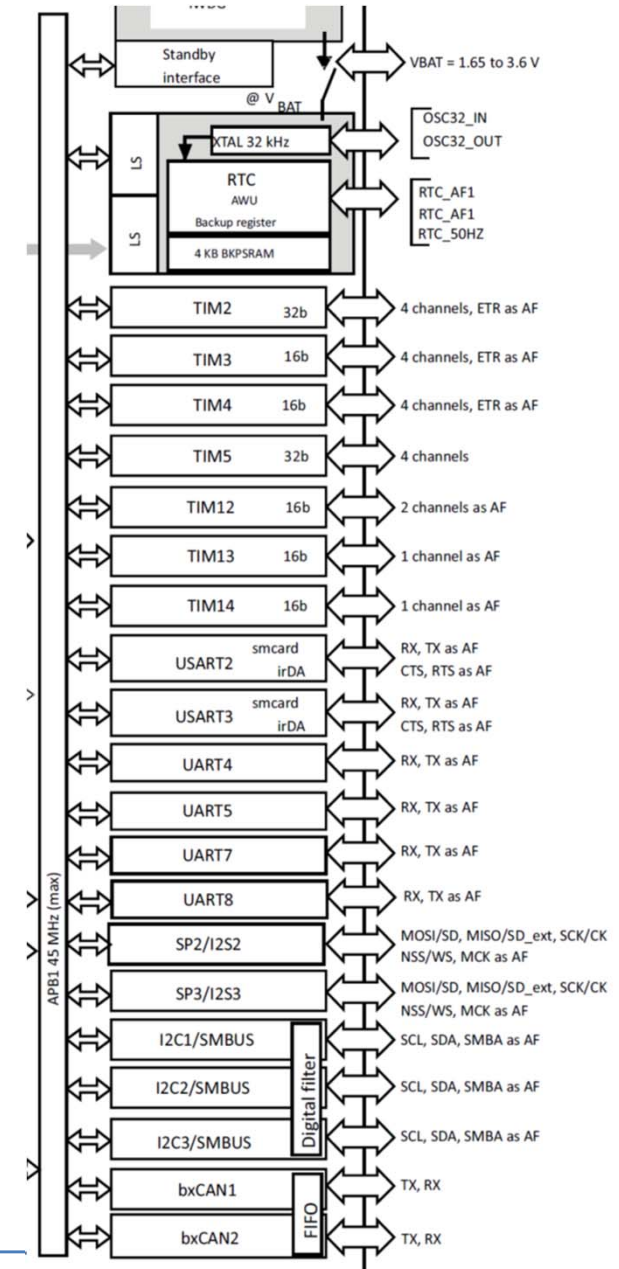
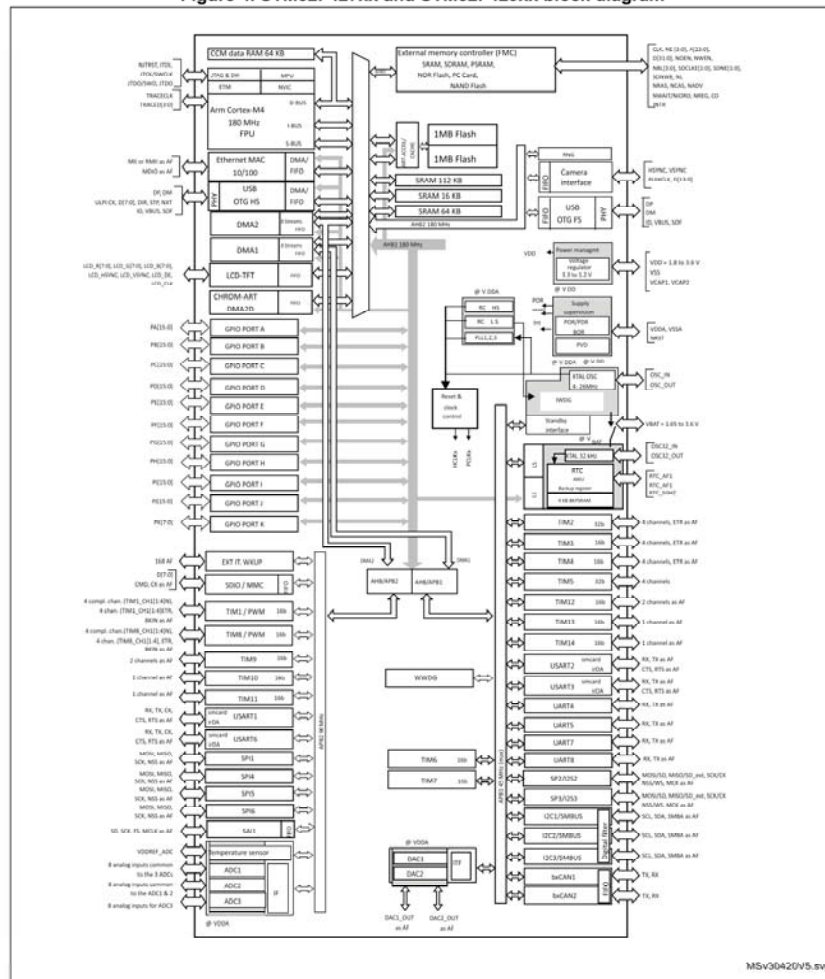
/* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2
   clocks dividers */
RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}

..
}

```

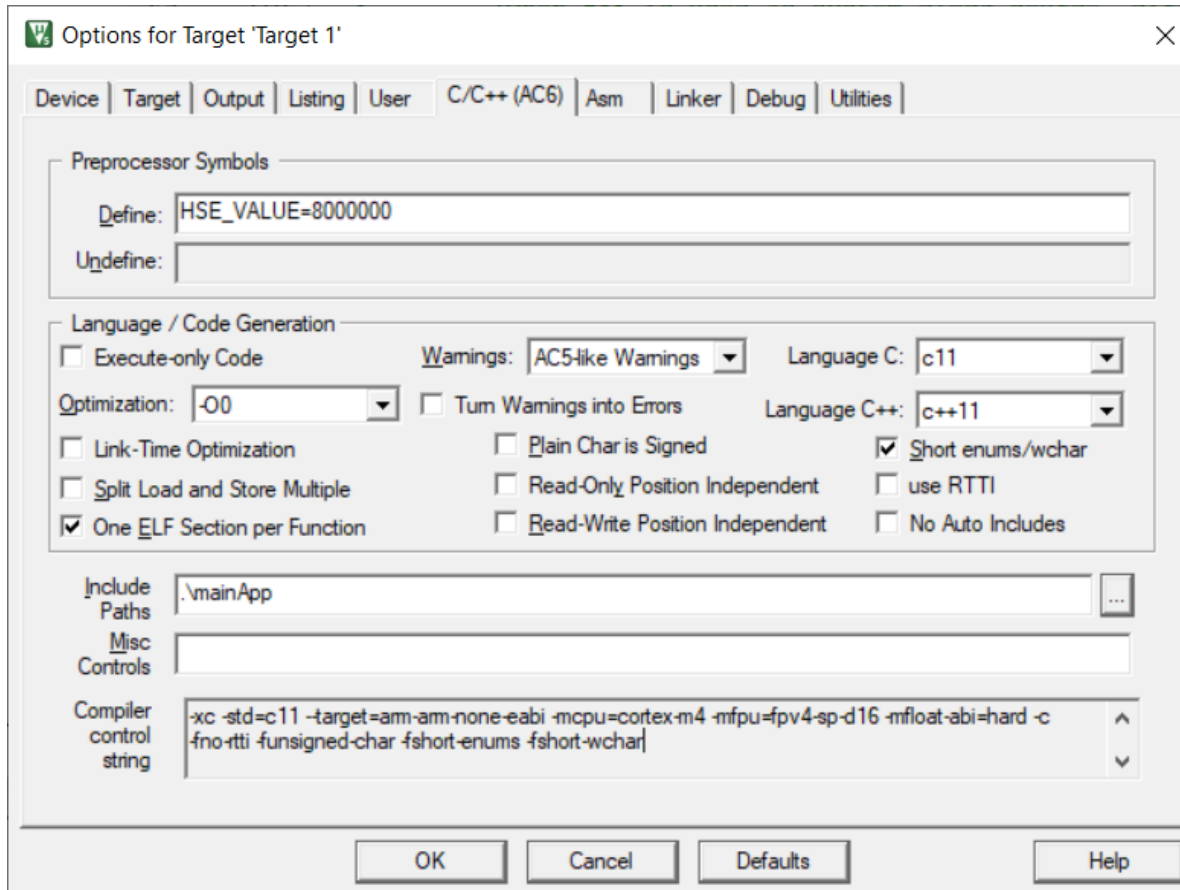
# Clock distribution for peripherals and other HW elements

- 32f429 Datasheet

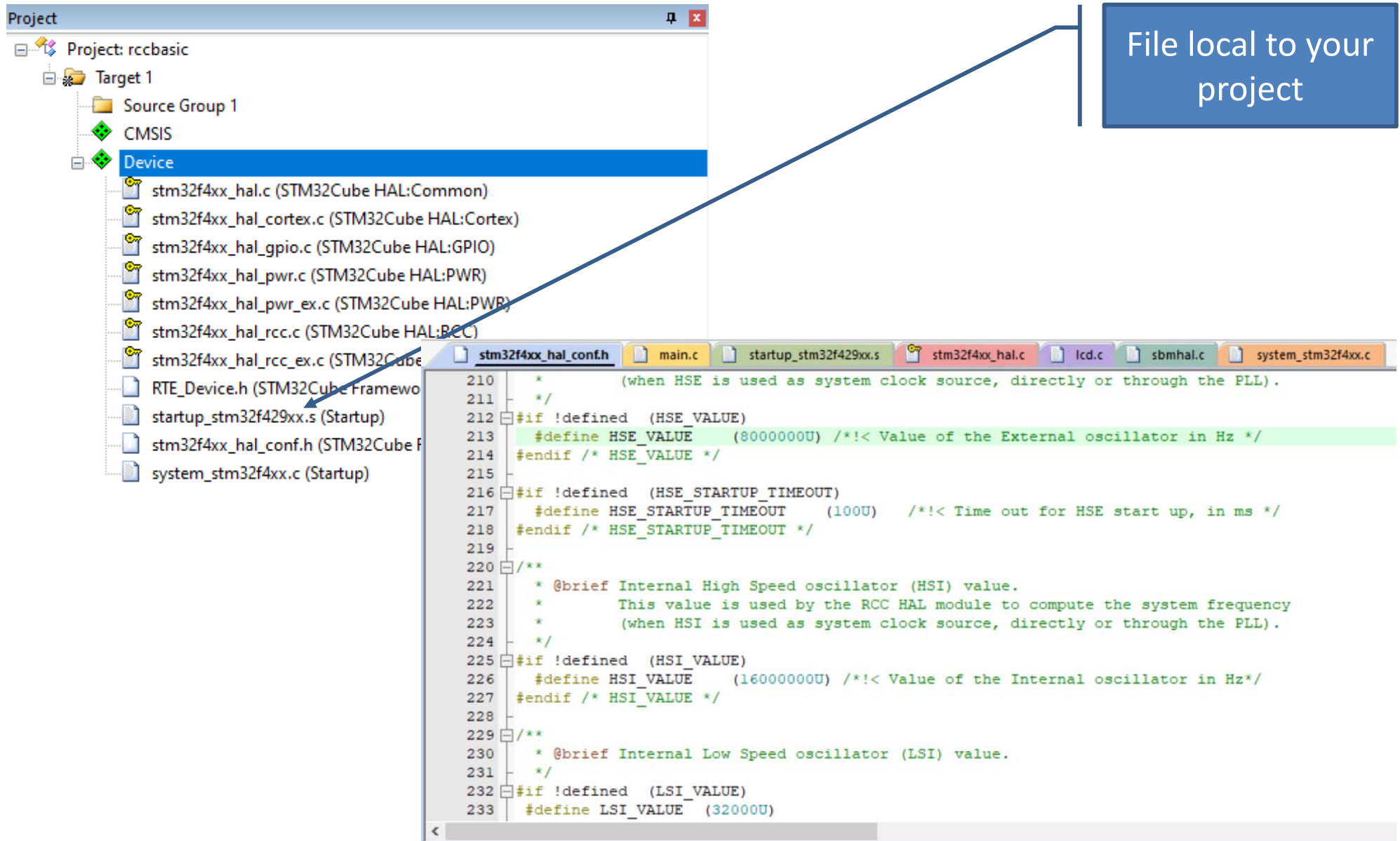


# Defining the HSE\_VALUE in Microvision

- This is equivalent to:
  - `#define HSE_VALUE 8000000`



# HAL parameters “customization”



The screenshot displays an IDE interface. On the left, the 'Project' tree shows a project named 'rccbasic' with a 'Target 1' configuration. Under 'Source Group 1', the 'Device' folder is expanded, listing various HAL files. The file 'stm32f4xx\_hal\_conf.h' is selected and its content is shown in the main editor. The code defines several parameters for the HAL, including HSE and HSI values. A blue box on the right contains the text 'File local to your project', with a blue arrow pointing from it to the 'stm32f4xx\_hal\_conf.h' file in the project tree.

Project: rccbasic

- Target 1
  - Source Group 1
    - CMSIS
    - Device
      - stm32f4xx\_hal.c (STM32Cube HAL:Common)
      - stm32f4xx\_hal\_cortex.c (STM32Cube HAL:Cortex)
      - stm32f4xx\_hal\_gpio.c (STM32Cube HAL:GPIO)
      - stm32f4xx\_hal\_pwr.c (STM32Cube HAL:PWR)
      - stm32f4xx\_hal\_pwr\_ex.c (STM32Cube HAL:PWR)
      - stm32f4xx\_hal\_rcc.c (STM32Cube HAL:RCC)
      - stm32f4xx\_hal\_rcc\_ex.c (STM32Cube HAL:RCC)
      - RTE\_Device.h (STM32Cube Framework)
      - startup\_stm32f429xx.s (Startup)
      - stm32f4xx\_hal\_conf.h (STM32Cube HAL)
      - system\_stm32f4xx.c (Startup)

stm32f4xx\_hal\_conf.h

```
210 * (when HSE is used as system clock source, directly or through the PLL).
211 */
212 #if !defined (HSE_VALUE)
213 #define HSE_VALUE (8000000U) /*!< Value of the External oscillator in Hz */
214 #endif /* HSE_VALUE */
215
216 #if !defined (HSE_STARTUP_TIMEOUT)
217 #define HSE_STARTUP_TIMEOUT (100U) /*!< Time out for HSE start up, in ms */
218 #endif /* HSE_STARTUP_TIMEOUT */
219
220 /**
221 * @brief Internal High Speed oscillator (HSI) value.
222 * This value is used by the RCC HAL module to compute the system frequency
223 * (when HSI is used as system clock source, directly or through the PLL).
224 */
225 #if !defined (HSI_VALUE)
226 #define HSI_VALUE (16000000U) /*!< Value of the Internal oscillator in Hz*/
227 #endif /* HSI_VALUE */
228
229 /**
230 * @brief Internal Low Speed oscillator (LSI) value.
231 */
232 #if !defined (LSI_VALUE)
233 #define LSI_VALUE (32000U)
```

File local to your project