Hardware Abstraction & Low Level Layers for STM32

Eduardo Barrera

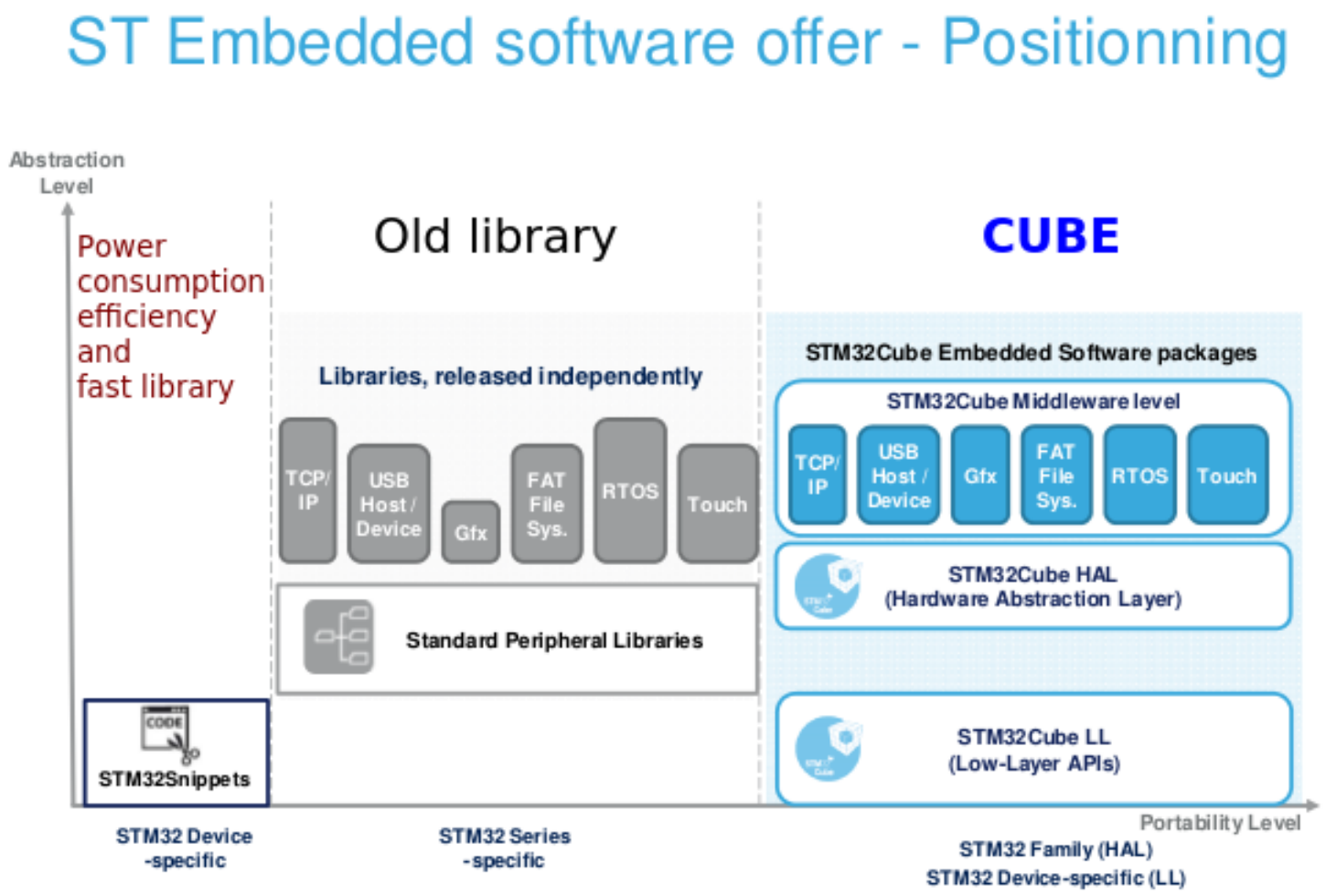
Julián Nieto

Mariano Ruiz

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

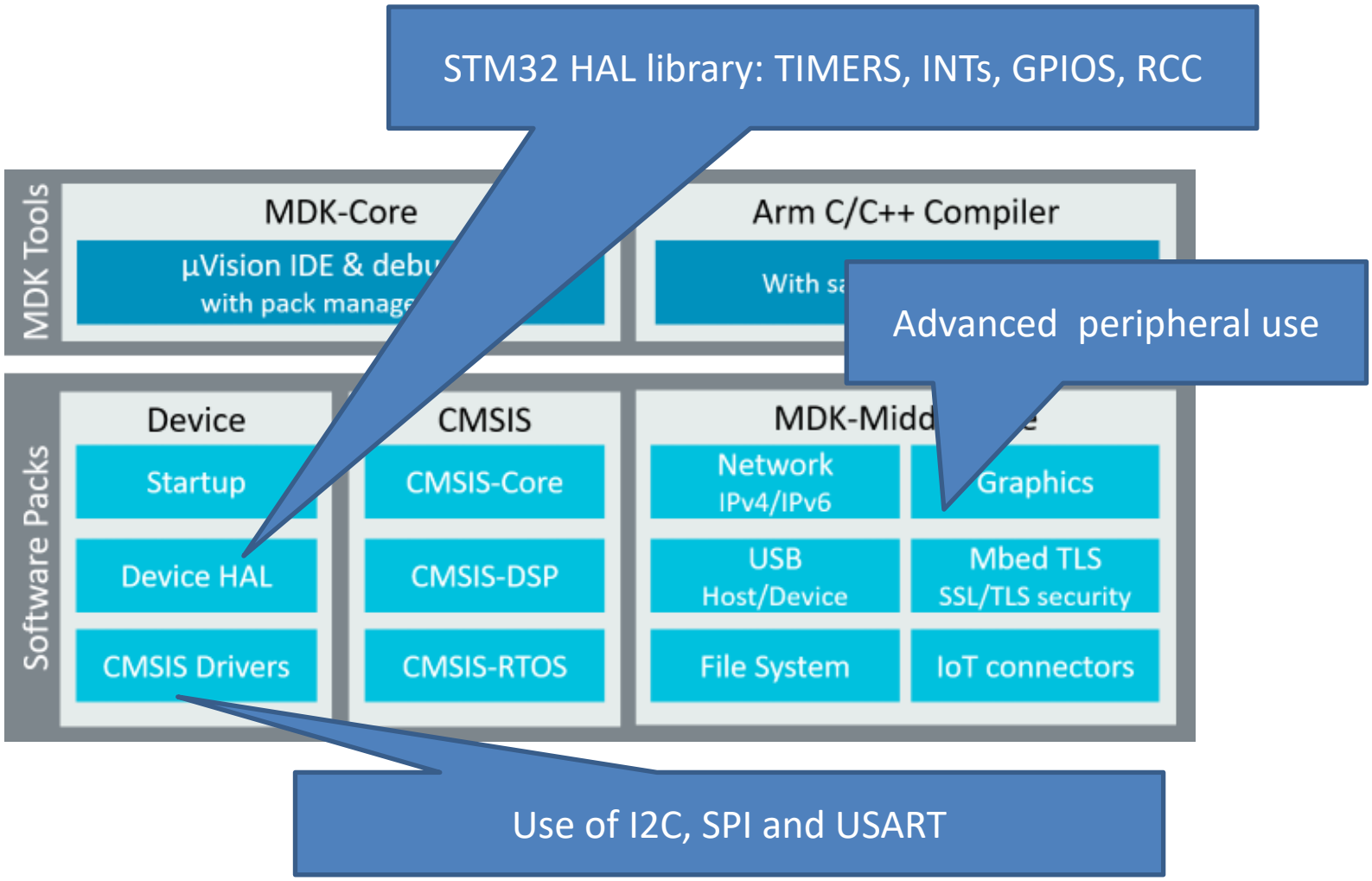


In Microprocessor Based Systems we will use the HAL API

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



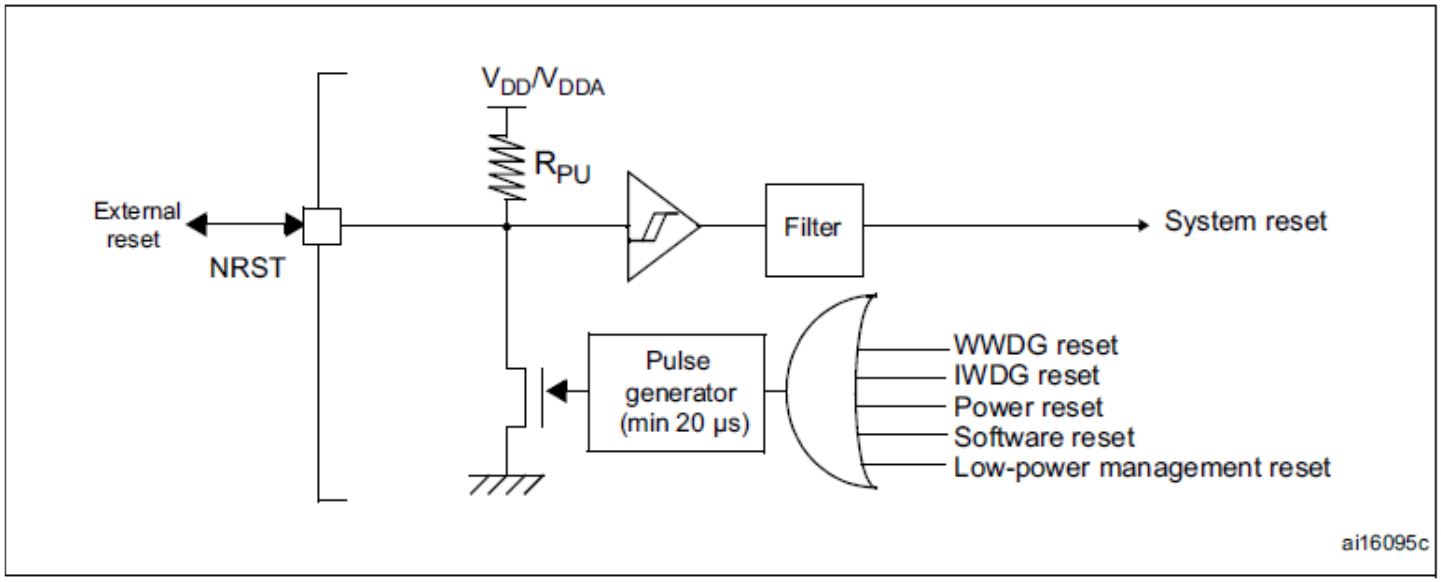
Software development in SBM HAL

- Keil Microvision for editing, compiling, linking, and debugging
- STM-32 HAL for configuring:
  - clocks and reset circuits (HAL RCC)
  - GPIO
  - Timers
  - Interrupts (NVIC and EXT1)
- CMSIS-Drivers for:
  - I2C and SPI
  - USART

RCC: Reset and Clock Control

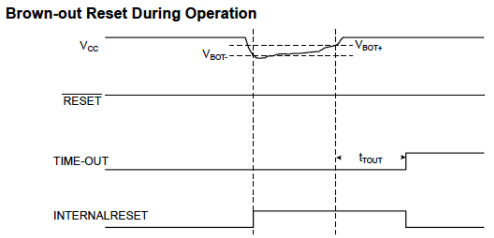
Reset circuit

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



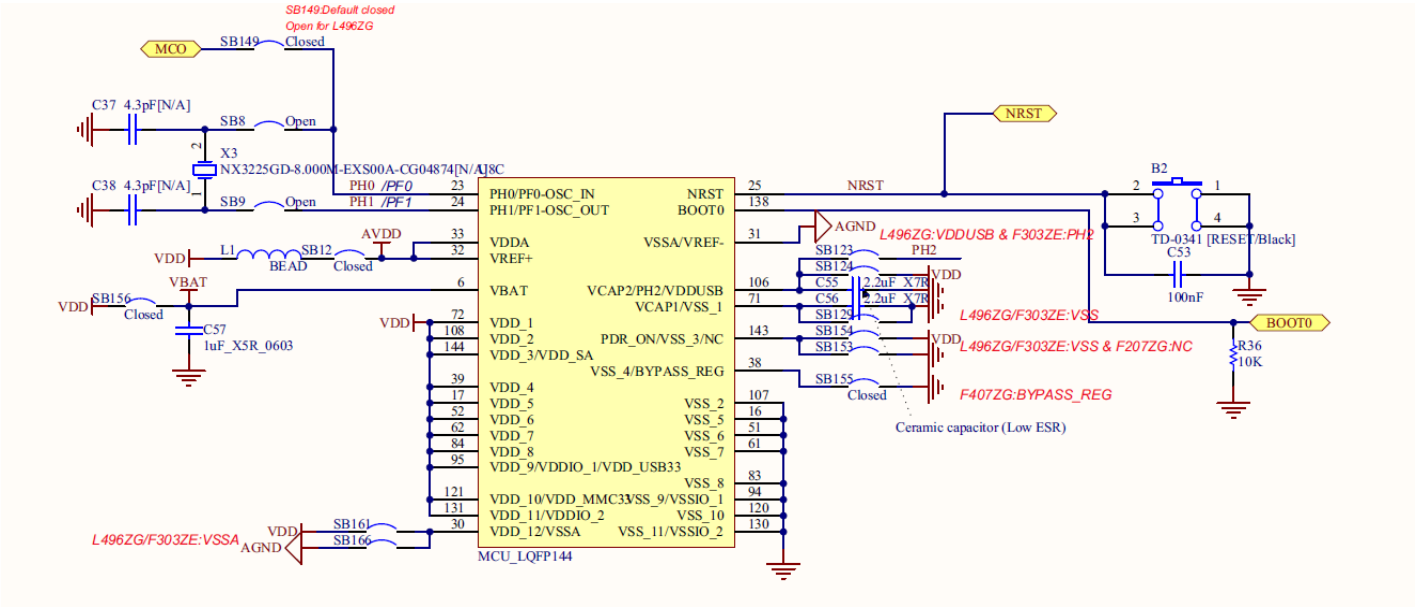
Reset circuit (3 types of reset sources)

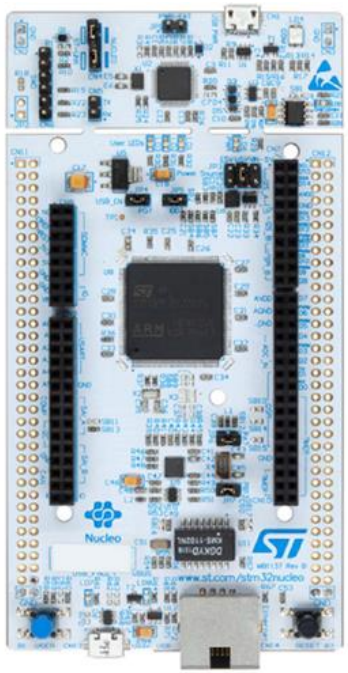
- System Reset
  - Low level on external pin NRST
  - Window watchdog end of count condition (WWDG). Counter that must be refreshed within a specific time window. If not, the watchdog generates a system reset. WWDG is connected to APB1.
  - Independent watchdog end of count (IWDG). Triggers a system reset if a counter reaches a given timeout value (it uses the RC oscillator)
  - Software reset
  - Low power management reset
    - Reset when entering in the Standby mode
    - Reset when entering in Stop mode
- Power reset
  - Power-on/down reset (POR/PDR), “Brownout Reset” (BOR)
- Backup domain reset (Reset of RTC registers)



NUCLEO-144 with STM32F429ZI

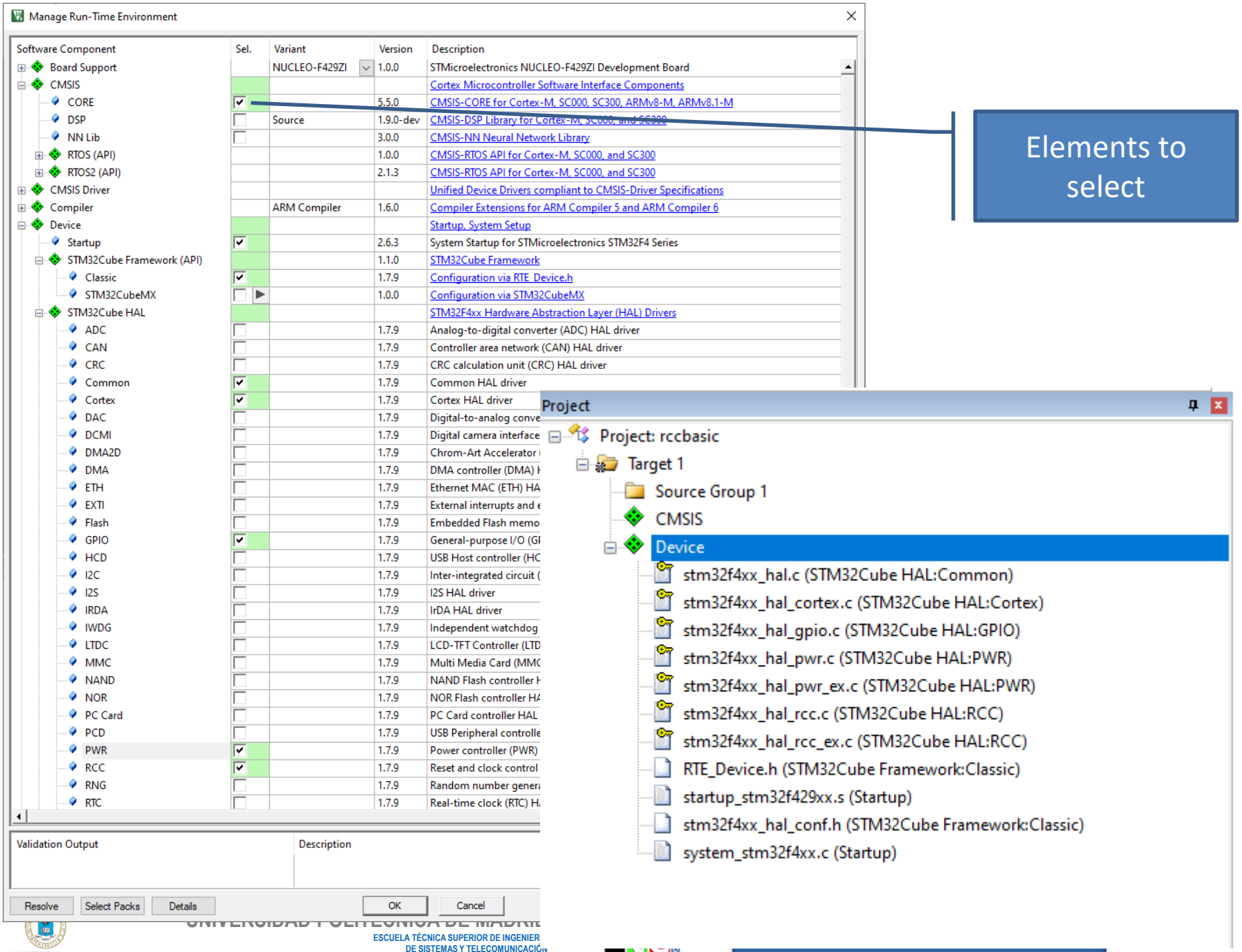
- B2 Button: generates reset on NRST signal
- Connection detail in NUCLEO-144 with STM32F429ZI (144 pins)





Creation of a default project with Keil Microvision

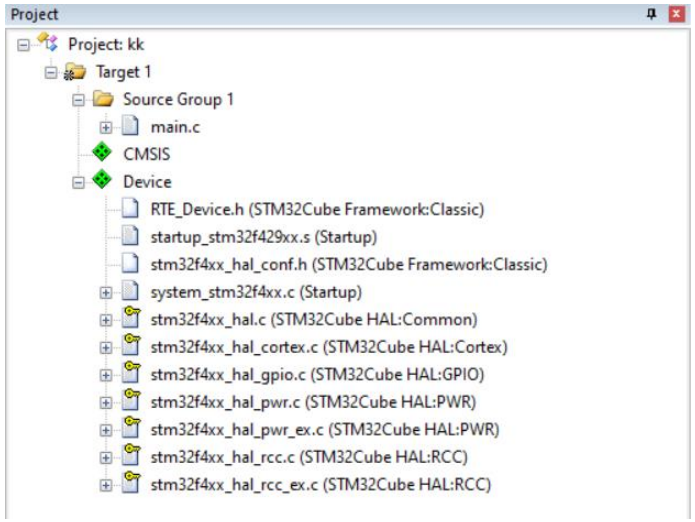
- Run Keil Microvision
- New Project
- Select the ST device
- Select the minimum software elements (next slide)
- Press OK
- Inspect the project created


Manage Run-Time Environment
Software Component
Sel.
Variant
Version
Description
Board Support
NUCLEO-F429ZI
1.0.0
STMicroelectronics NUCLEO-F429ZI Development Board
CMSIS
Cortex Microcontroller Software Interface Components
CORE
5.5.0
CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP
Source
1.9.0-dev
CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib
3.0.0
CMSIS-NN Neural Network Library
RTOS (API)
1.0.0
CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)
2.1.3
CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver
Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler
ARM Compiler
1.6.0
Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device
Startup, System Setup
Startup
2.6.3
System Startup for STMicroelectronics STM32F4 Series
STM32Cube Framework (API)
1.1.0
STM32Cube Framework
Classic
1.7.9
Configuration via RTE_Device.h
STM32CubeMX
1.0.0
Configuration via STM32CubeMX
STM32Cube HAL
STM32F4xx Hardware Abstraction Layer (HAL) Drivers
ADC
1.7.9
Analog-to-digital converter (ADC) HAL driver
CAN
1.7.9
Controller area network (CAN) HAL driver
CRC
1.7.9
CRC calculation unit (CRC) HAL driver
Common
1.7.9
Common HAL driver
Cortex
1.7.9
Cortex HAL driver
DAC
1.7.9
DCMI
1.7.9
DMA2D
1.7.9
DMA
1.7.9
ETH
1.7.9
EXTI
1.7.9
Flash
1.7.9
Embedded Flash memo
GPIO
1.7.9
General-purpose I/O (GI
HCD
1.7.9
USB Host controller (HC
I2C
1.7.9
Inter-integrated circuit (
I2S
1.7.9
I2S HAL driver
IRDA
1.7.9
IrDA HAL driver
IWDG
1.7.9
Independent watchdog
LTDC
1.7.9
LCD-TFT Controller (LTD
MMC
1.7.9
Multi Media Card (MMC
NAND
1.7.9
NAND Flash controller H
NOR
1.7.9
NOR Flash controller HA
PC Card
1.7.9
PC Card controller HAL
PCD
1.7.9
USB Peripheral controlle
PWR
1.7.9
Power controller (PWR)
RCC
1.7.9
Reset and clock control
RNG
1.7.9
Random number genera
RTC
1.7.9
Real-time clock (RTC) HA
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
```

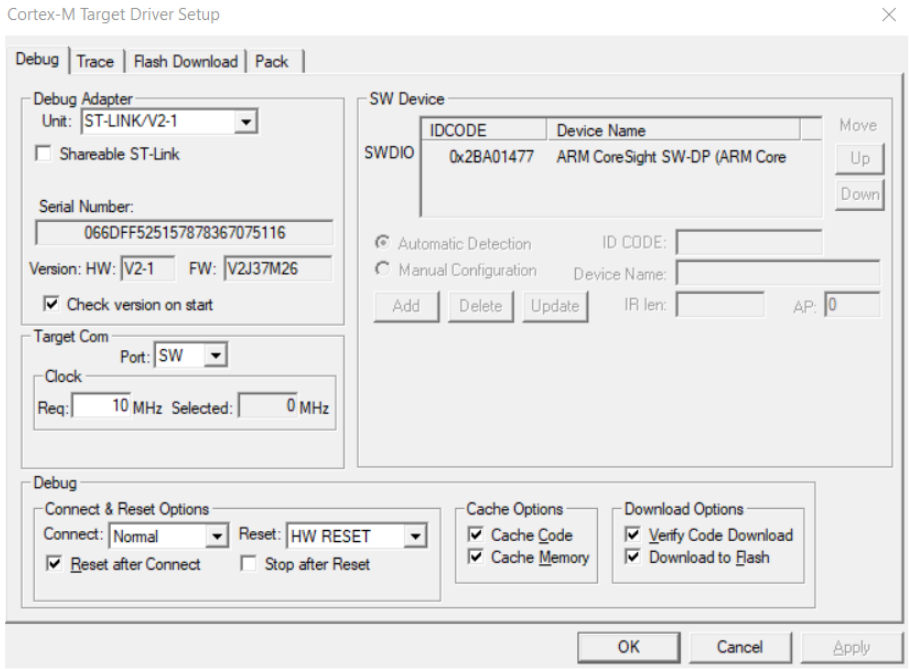


Reset Handler

- During the execution of the reset handler
  - The processor is using de 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

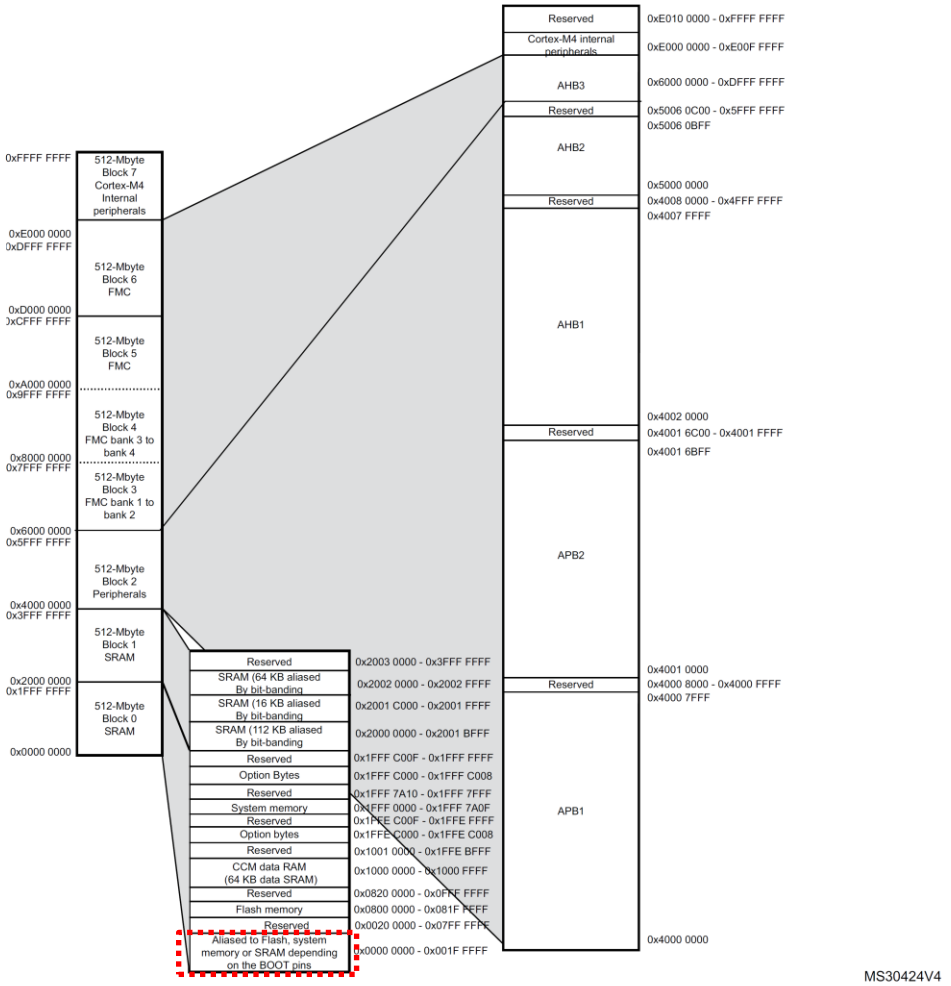
- Reset after connect: enables or disables the operation selected in the Reset drop-down list
- HW Reset: asserts the hardware reset signal
- SYSRESETREQ: performs a software reset (Cortex M and peripherals are reset)
- VECTRESET: Set the VECTRESET bit and only the cortex M is reset

Memory MAP and boot mode

Table 2. Boot modes

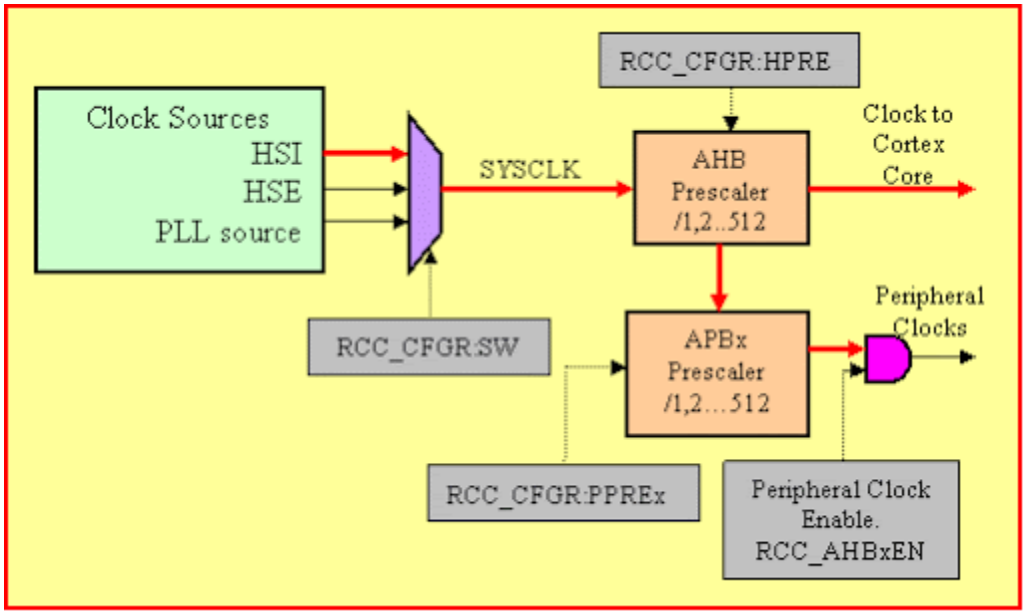
Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

- NUCLEO 144:
  Default BOOT0=0



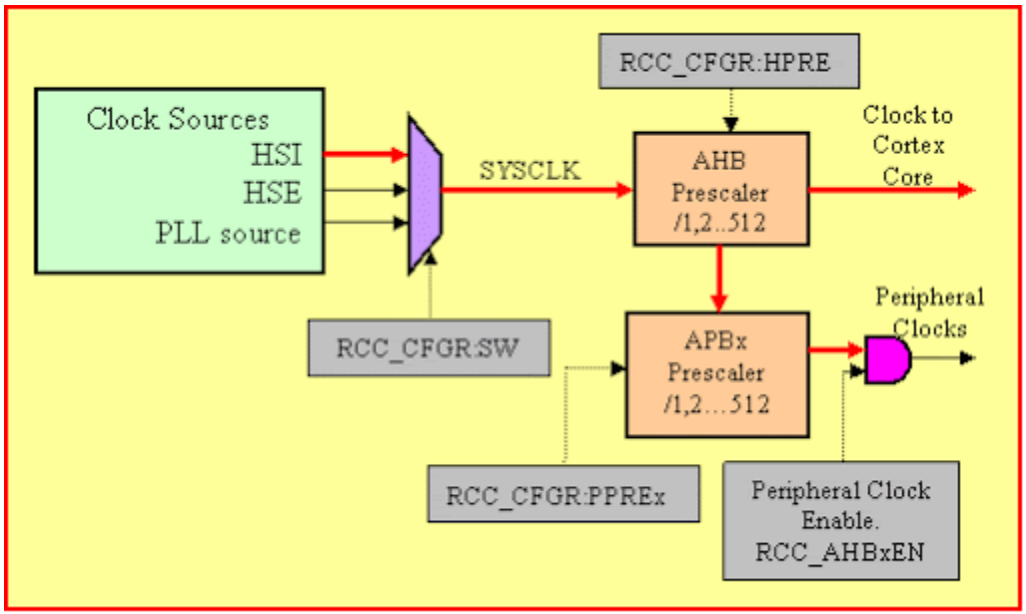
RCC: Clock Control

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

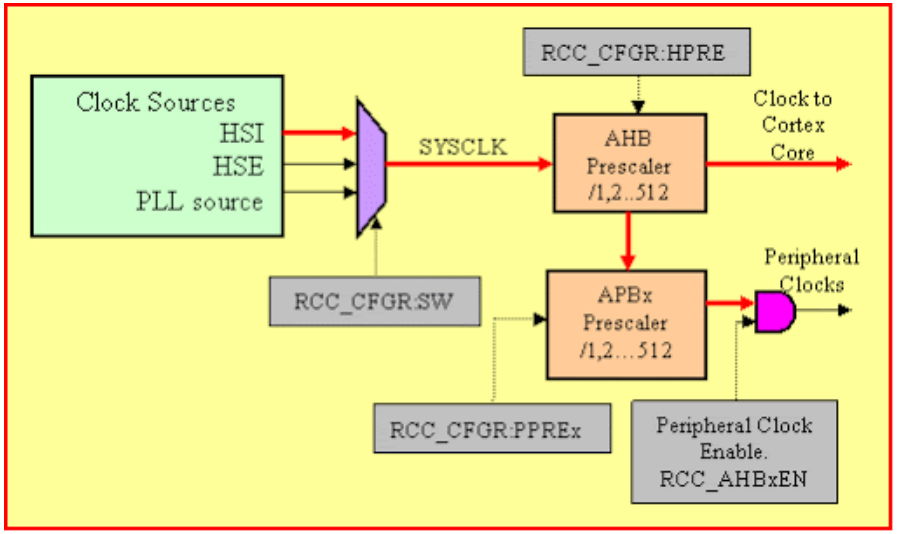
Basic clock circuitry for the STM32Fxxx



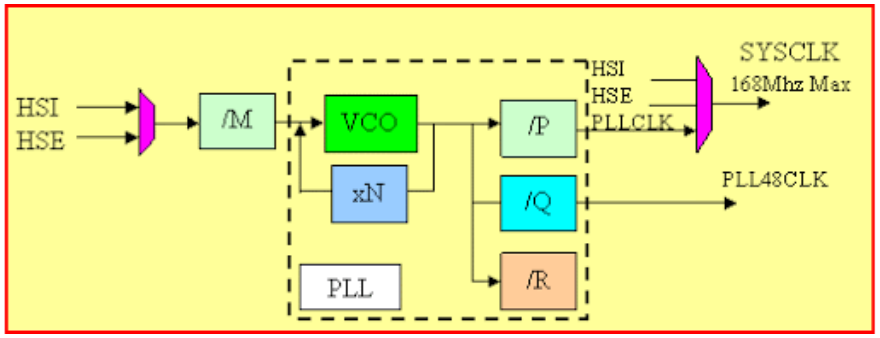
- After resetting the HSI (High-speed internal clock) is enabled
- HSE (High-speed external clock)

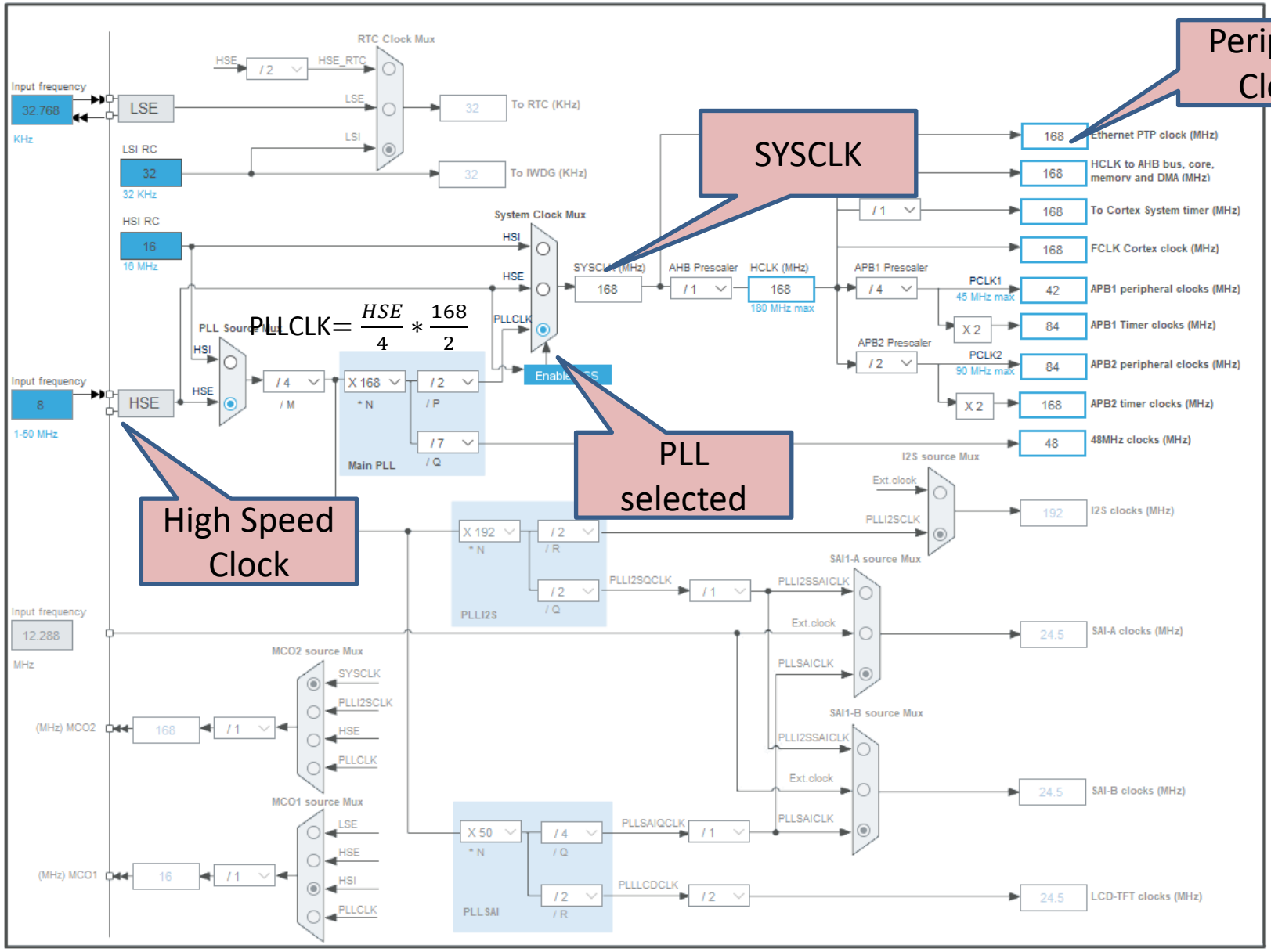
Basic clock circuitry for the STM32Fxxx

- Bus prescalers and peripheral clocks



- Using the Phase Locked loop

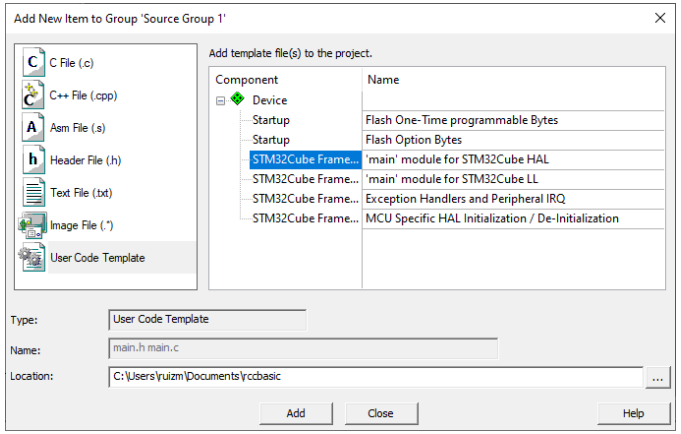




$$PLLCLK = \frac{HSE}{4} * \frac{168}{2}$$

In the Keil Microvision Project

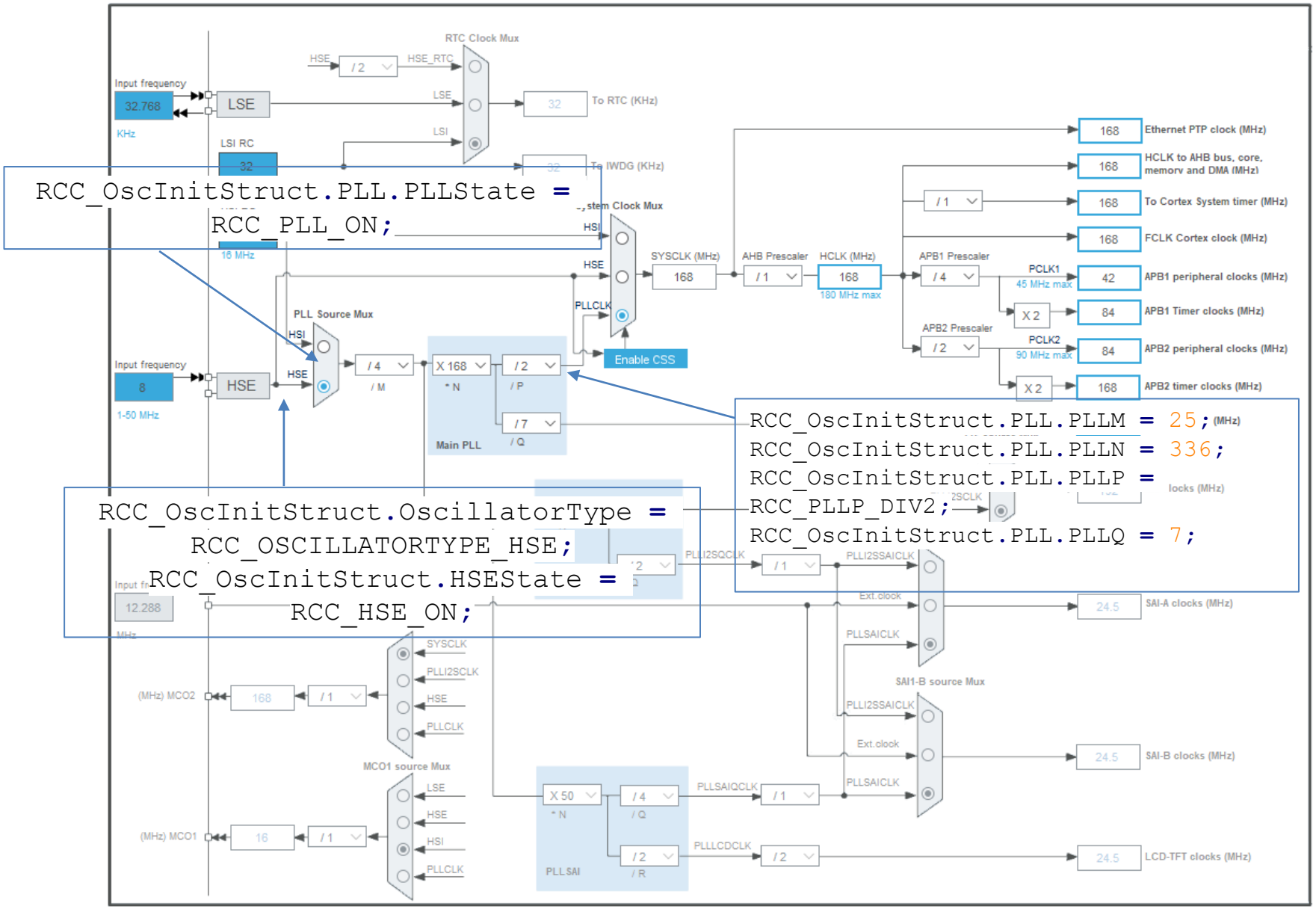
- In “Source Group 1”->Add New Item (User Code Template)
- Select Device->’main’ module for STM32Cube HAL-> Add



- 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;
```

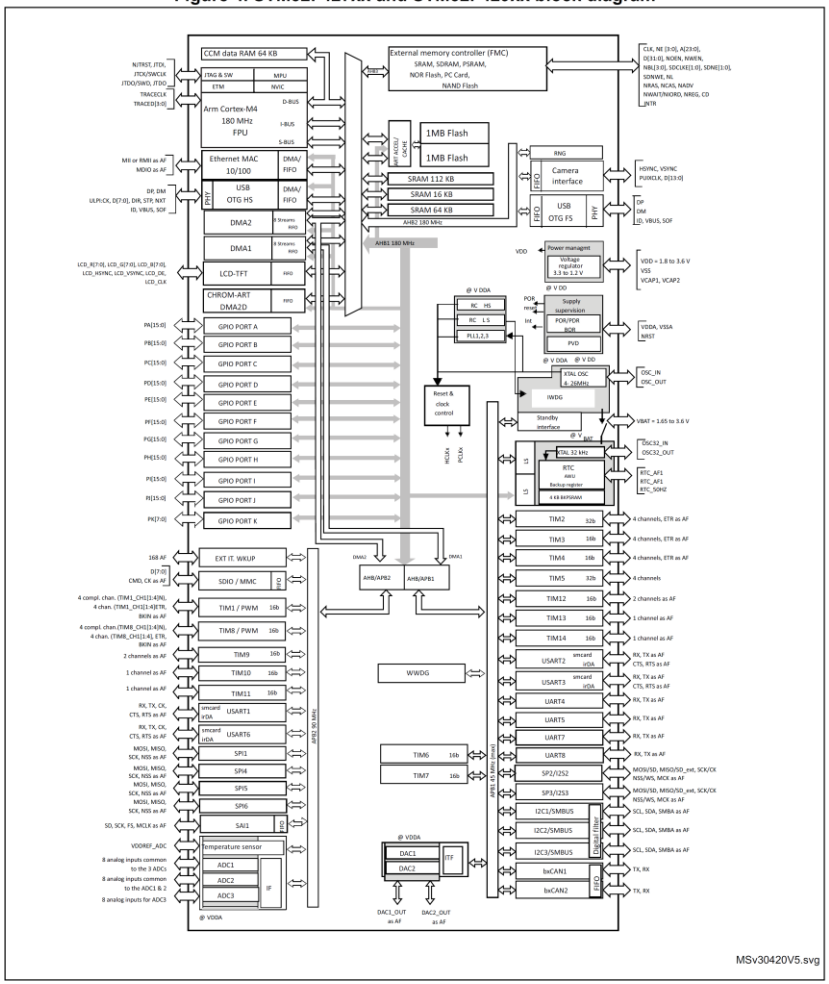

RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.PLL.PLLM = 25; (MHz)
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 7;
RTC Clock Mux
Input frequency
32.768
KHz
LSE
LSI RC
32
To RTC (KHz)
To IWDG (KHz)
HSE
/ 2
HSE_RTC
LSI
System Clock Mux
HSI
16 MHz
HSE
PLLCLK
SYSCLK (MHz)
168
AHB Prescaler
/ 1
HCLK (MHz)
168
180 MHz max
APB1 Prescaler
/ 4
PCLK1
45 MHz max
APB2 Prescaler
/ 2
PCLK2
90 MHz max
Enable CSS
PLL Source Mux
Input frequency
8
1-50 MHz
/ 4
/ M
X 168
* N
/ 2
/ P
/ 7
/ Q
Main PLL
168
Ethernet PTP clock (MHz)
168
HCLK to AHB bus, core, memory and DMA (MHz)
168
To Cortex System timer (MHz)
168
FCLK Cortex clock (MHz)
42
APB1 peripheral clocks (MHz)
X 2
84
APB1 Timer clocks (MHz)
84
APB2 peripheral clocks (MHz)
X 2
168
APB2 timer clocks (MHz)
Input frequency
12.288
MHz
PLLI2SQCLK
/ 1
PLLI2SSAICLK
Ext.clock
PLLSAICLK
24.5
SAI-A clocks (MHz)
SAI1-B source Mux
PLLI2SSAICLK
Ext.clock
PLLSAICLK
24.5
SAI-B clocks (MHz)
SYSCLK
PLLI2SCLK
HSE
PLLCLK
(MHz) MCO2
168
/ 1
MCO1 source Mux
LSE
HSE
HSI
PLLCLK
(MHz) MCO1
16
/ 1
X 50
* N
/ 4
/ Q
PLLSAIQCLK
/ 1
PLLSAICLK
/ 2
/ R
PLLLCDCLK
/ 2
PLLSAI
24.5
LCD-TFT clocks (MHz)

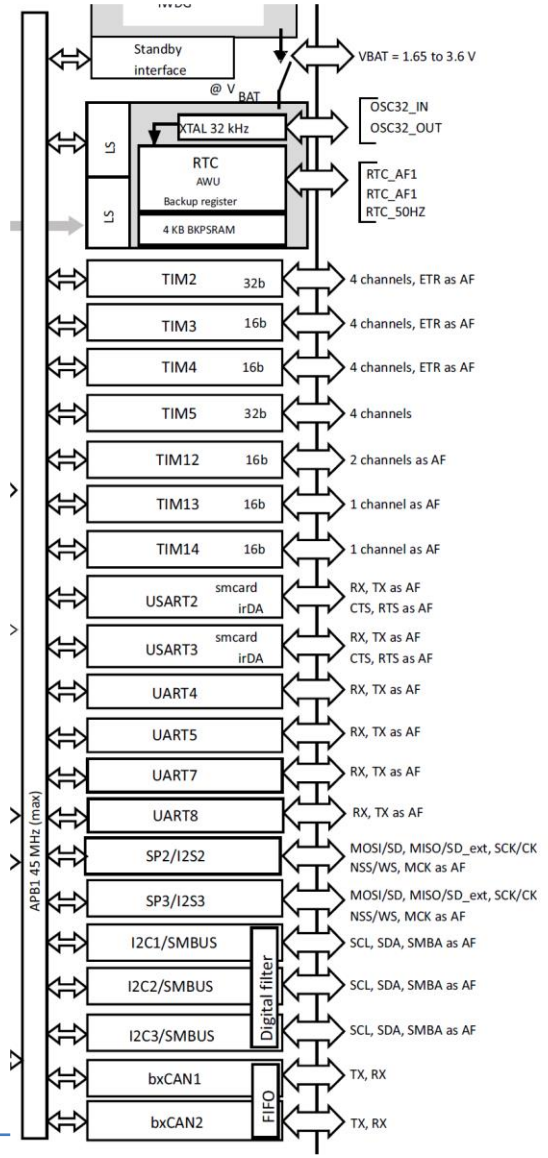
```
 if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    /* Initialization Error */
    Error_Handler();
  }

  /* 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

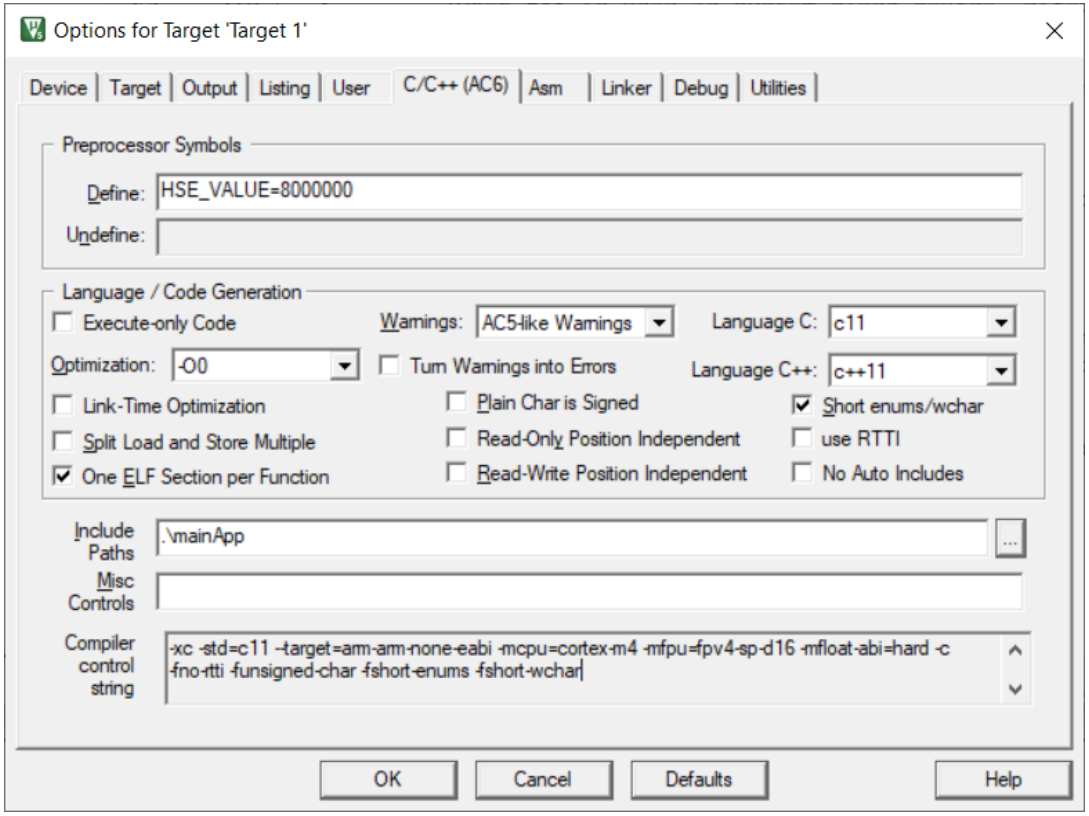
- 32f429 Datasheet

Figure 4. STM32F427xx and STM32F429xx block diagram



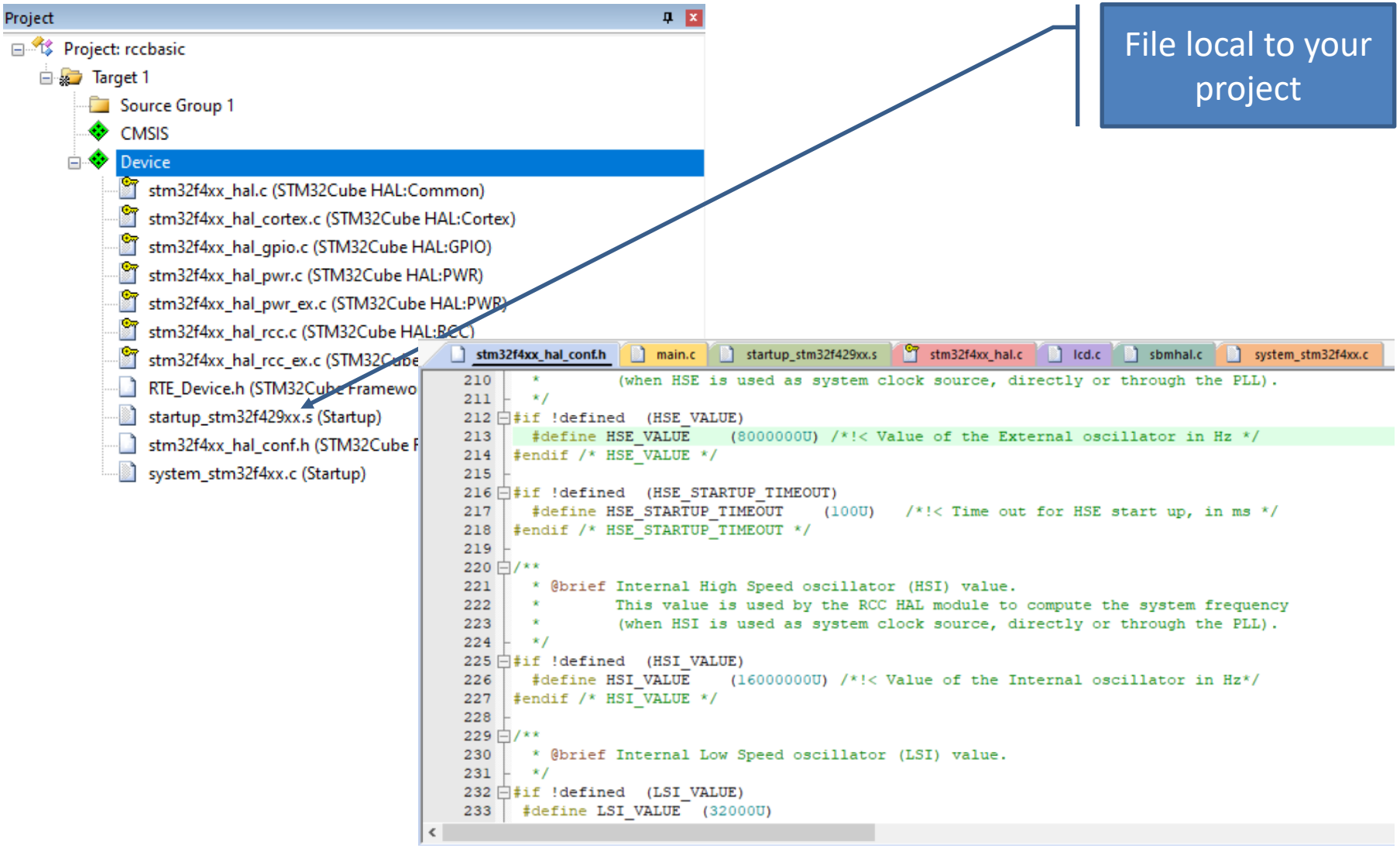


Defining the HSE_VALUE in Microvision



- This is equivalent to:
  - #define HSE_VALUE 8000000

HAL parameters “customization”

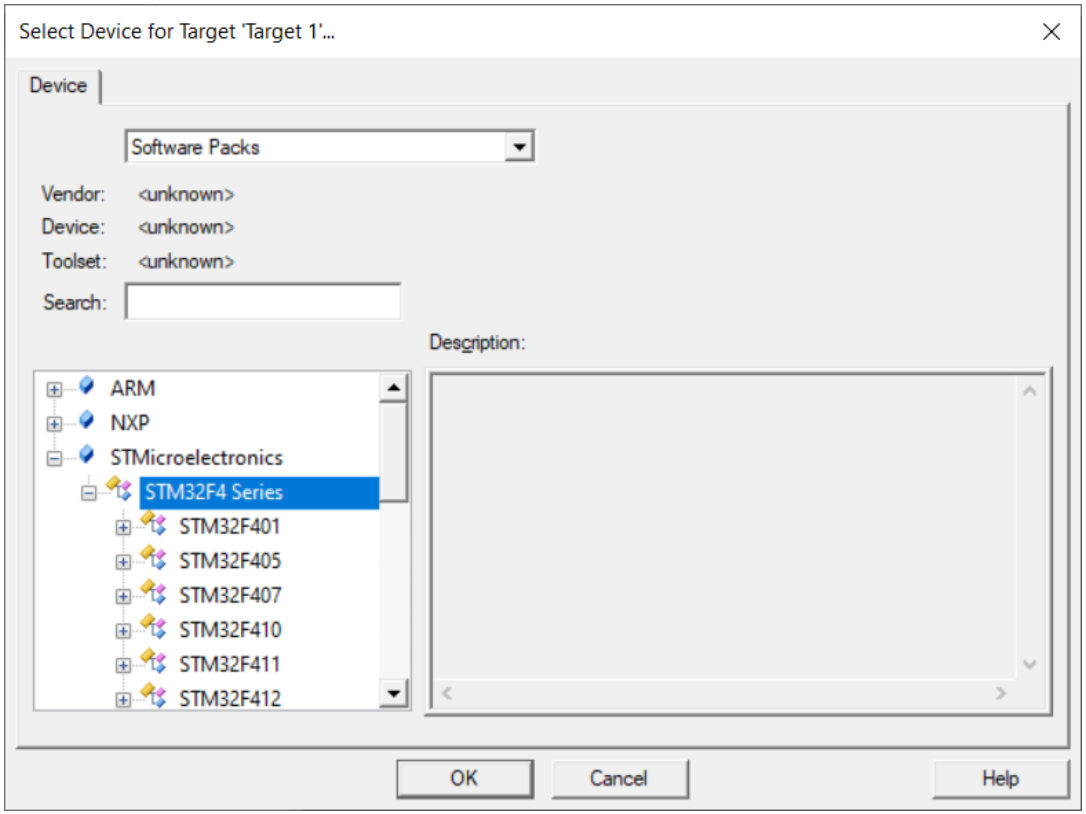


Keil Microvision

First Project from scratch

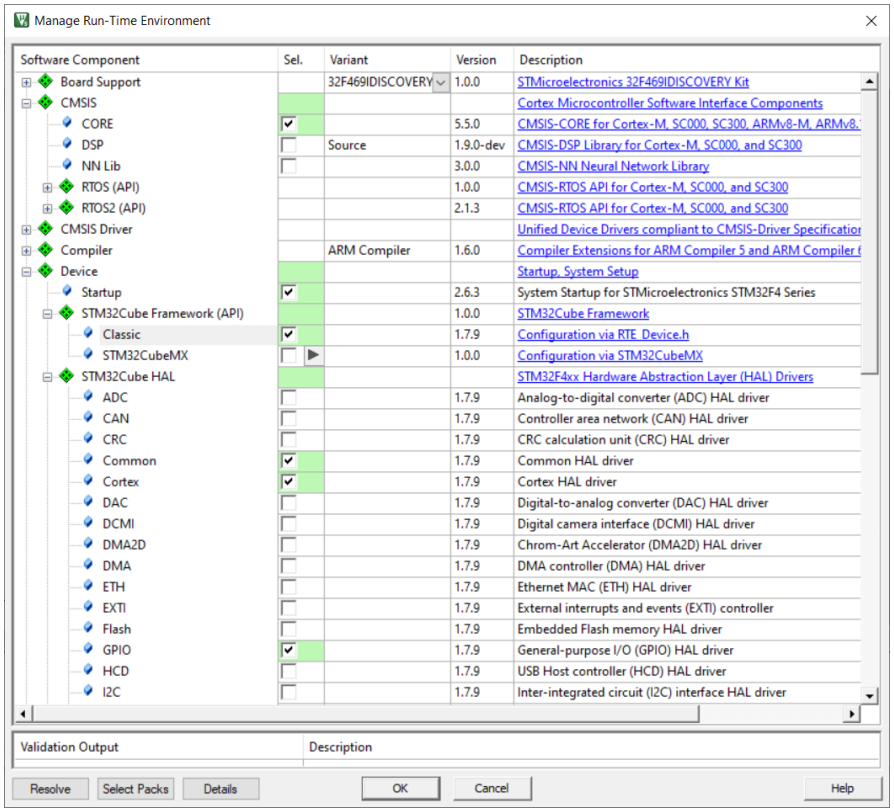
Steps I

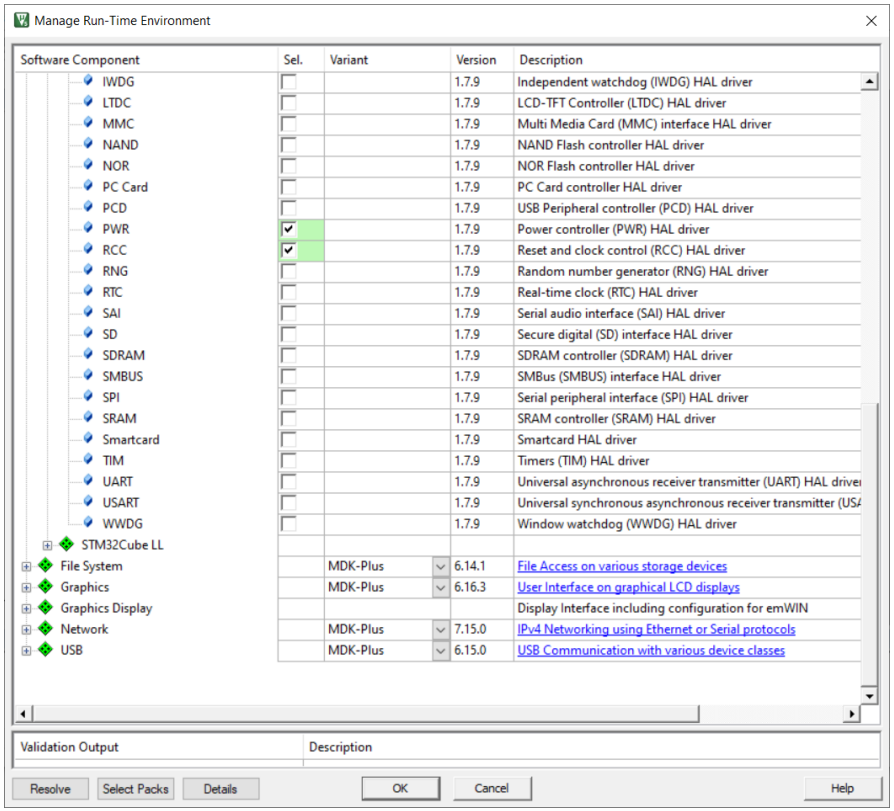
- Project->New Microvision Project
- Device selection (STM32F429ZI)



Steps II

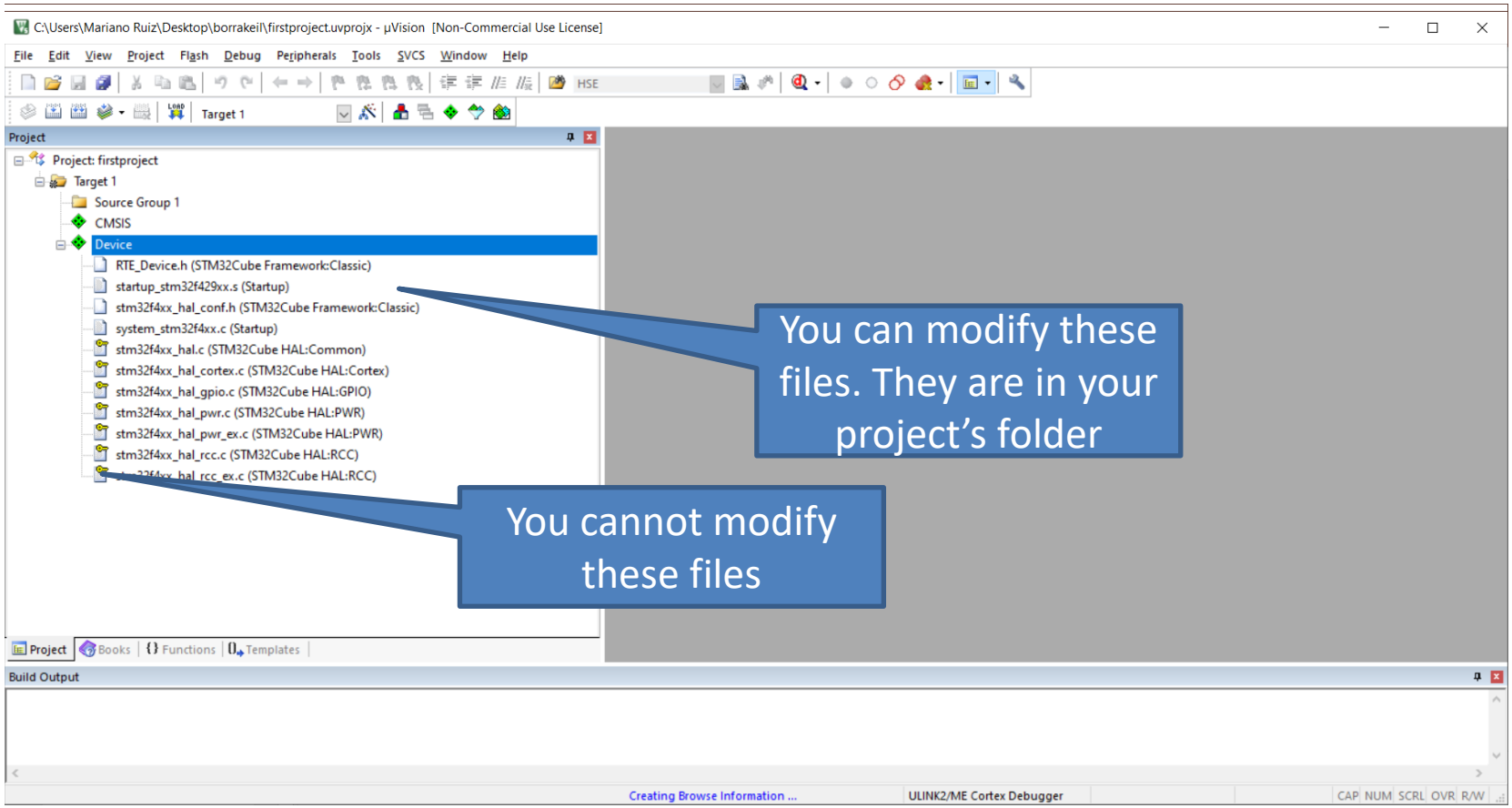
- Configuration of the Run Time Environment





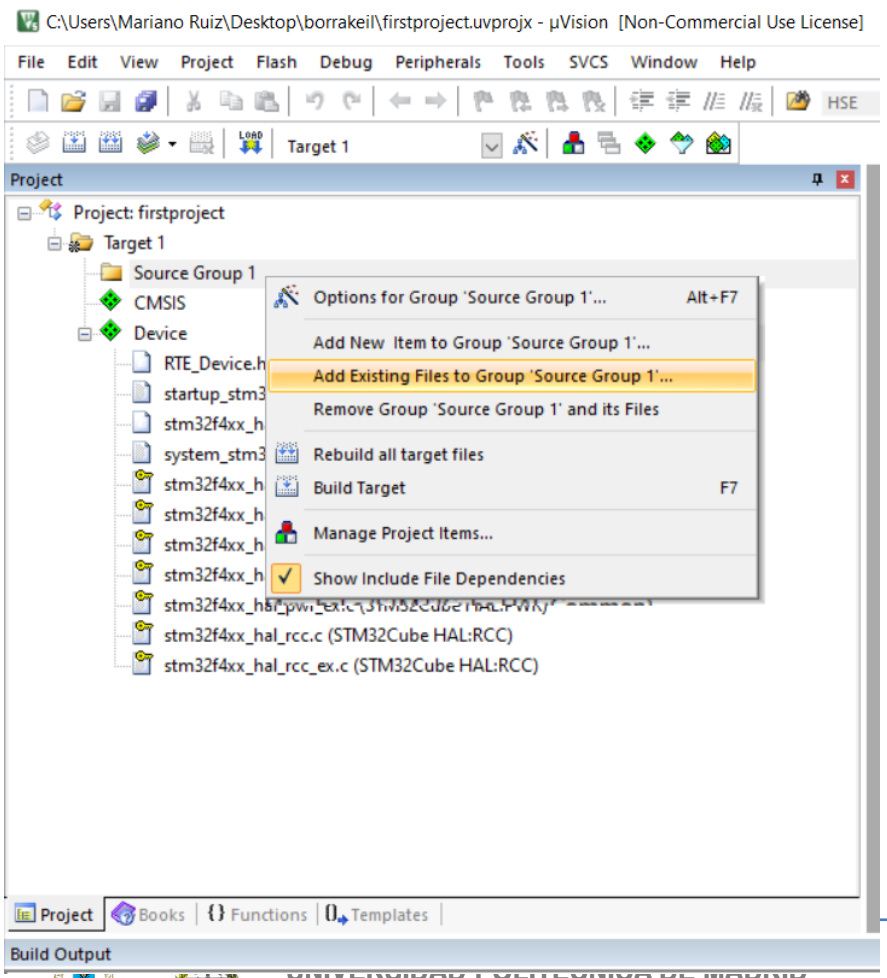
Step III

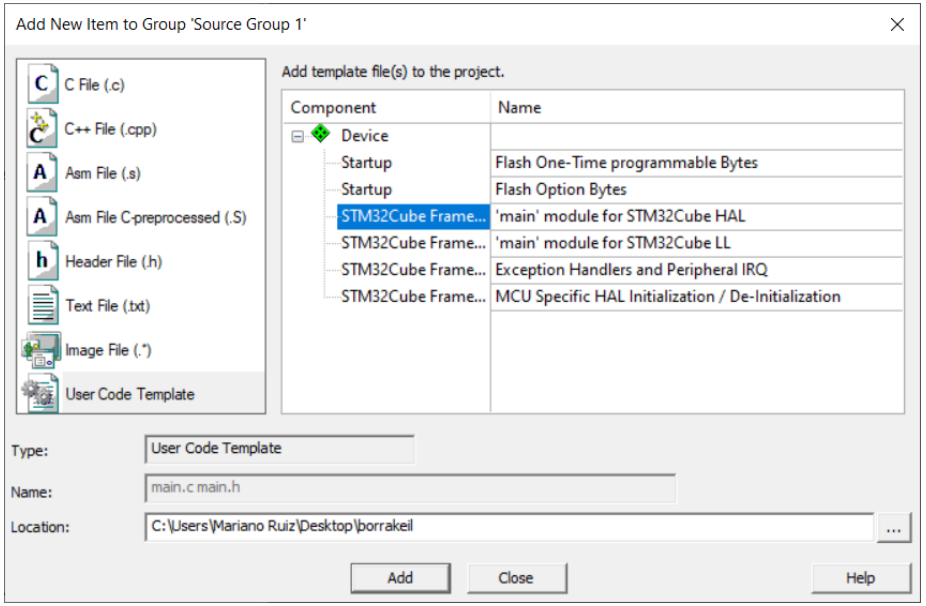
- Project created



Step IV

- Adding a basic “main.c” file



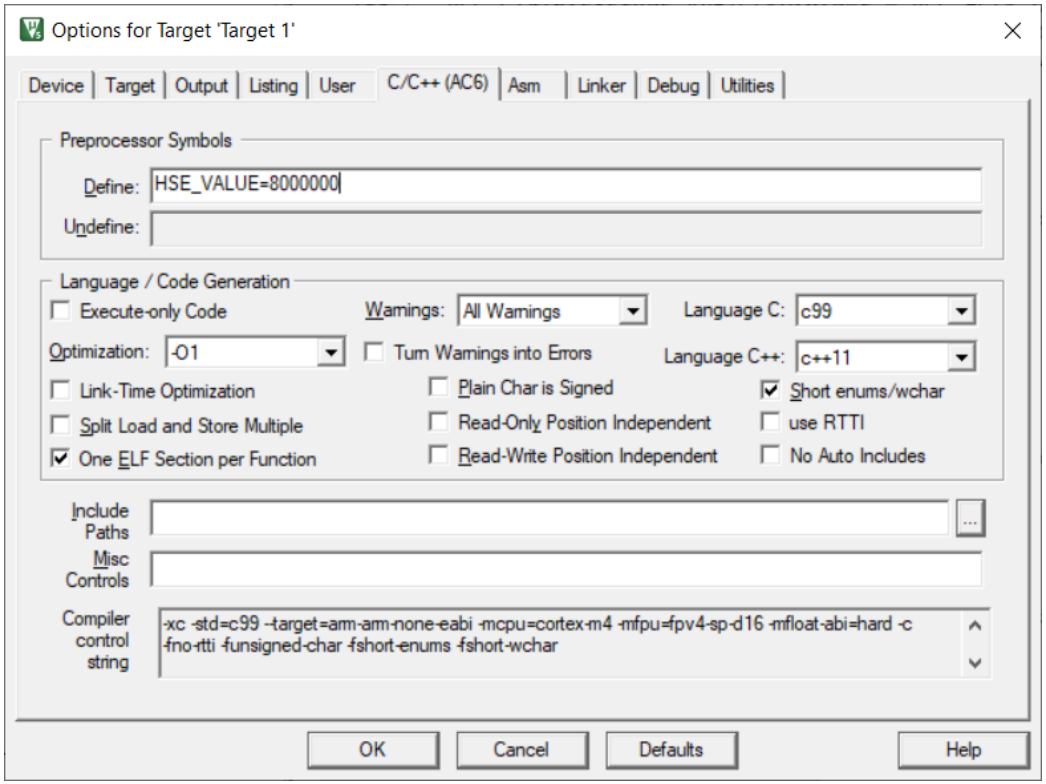


Step V

- Some details of “main.c”
  - RTE_Components.h added
  - main() calls to
    - HAL_Init()
    - SystemClock_Config();
    - SystemCoreClockUpdate();
  - SystemClock_Config() and Error_Handler functions code
  - Main code is an infinite loop

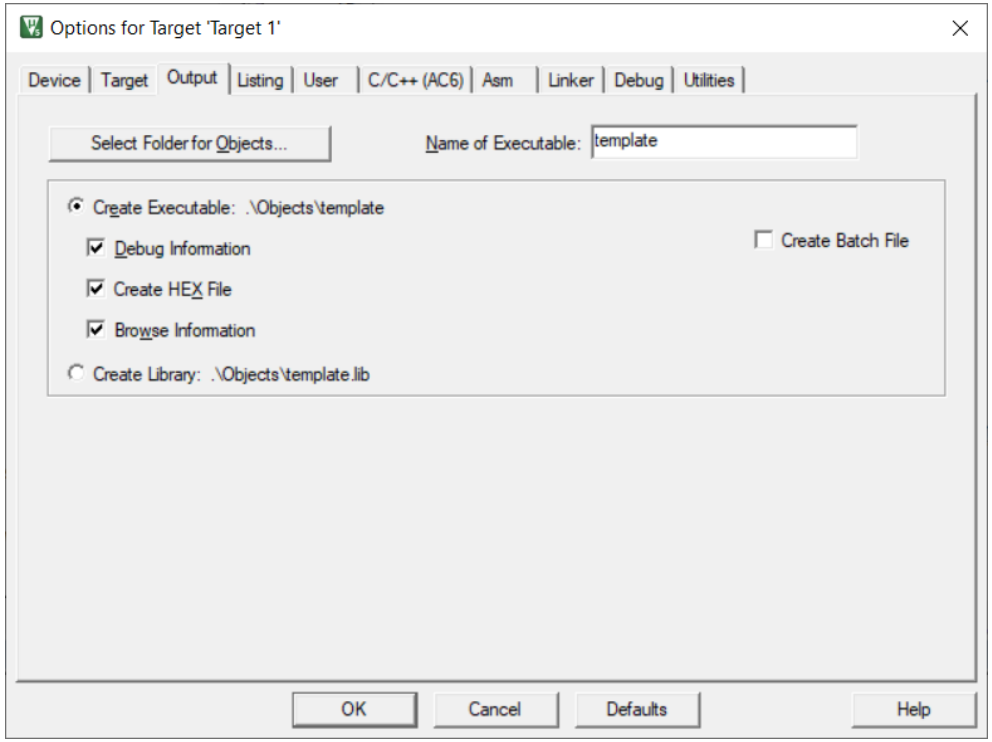
Step VI

- Add definition of HSE_VALUE to the Keil project
- Compile and debug



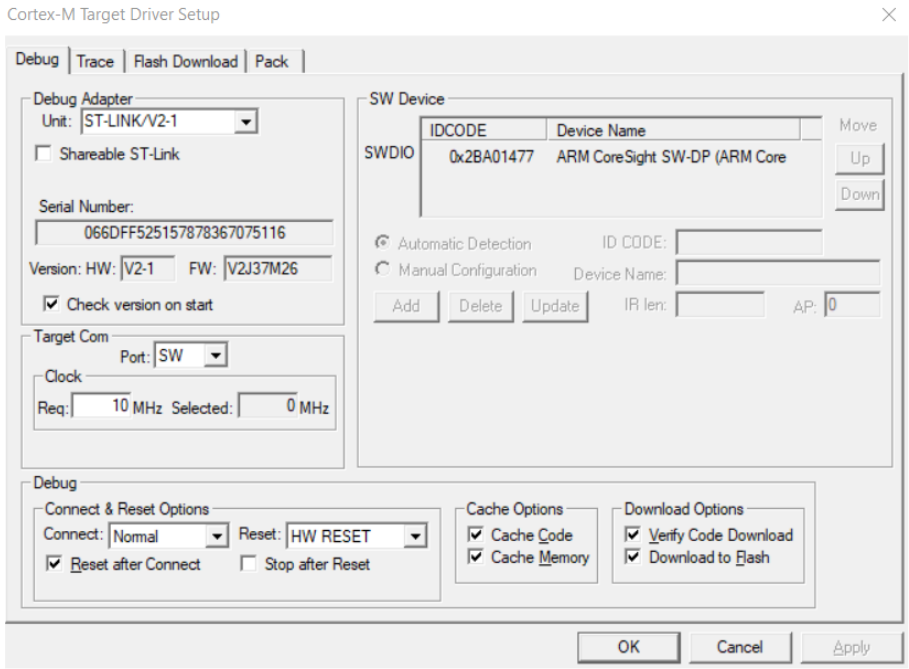
Activating the debug

- Select “Debug Information”
- For release version unselect it!



(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



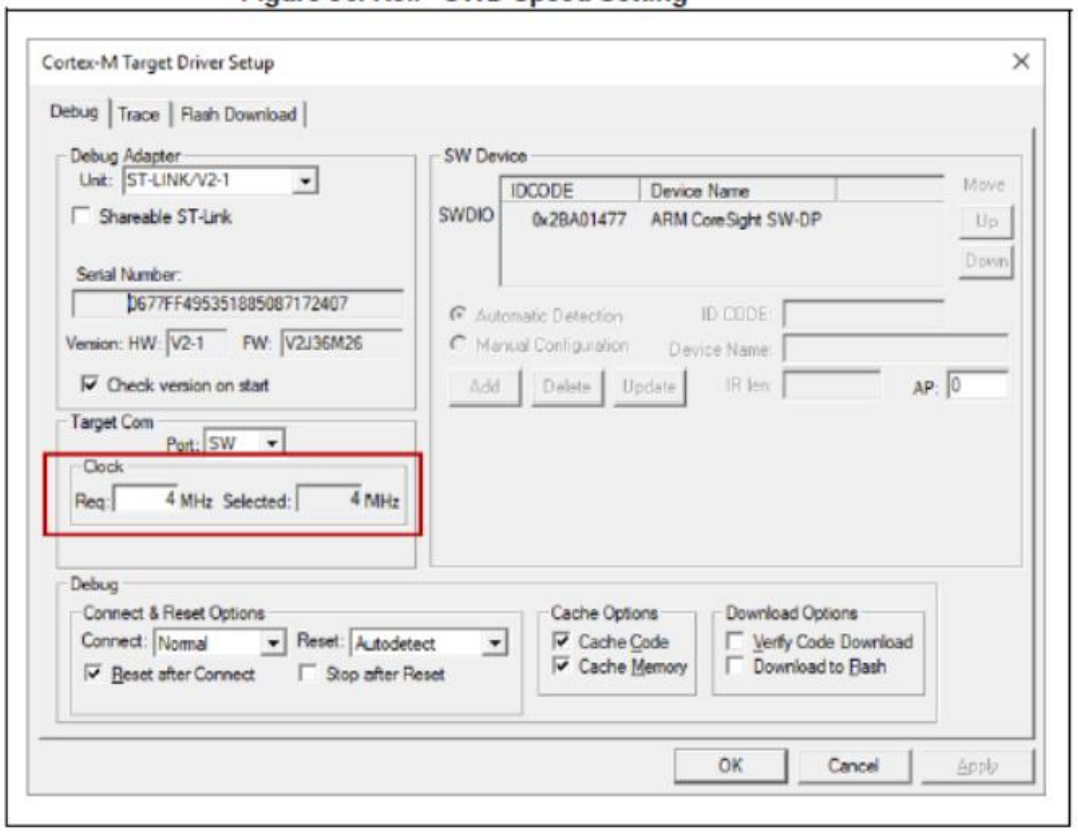
(Keil) Debug Reset options

- 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

Serial Wire Debug (Speed configuration)

- Set to 4MHz maximum

Figure 36. Keil® SWD Speed Setting



Debug capabilities

Table 7. STM32 Series vs. debug capabilties

STM32 Series	Cortex type	SWD	JTAG	ETM	SWO	Hardware breakpoints	Core Reset	MCO(1)
L0/F0	M0/0+	Yes	No	No	No	4	No	1
F1/L1/F2	M3	Yes	Yes	Yes(2)	Yes	6	Yes	1
F3/F4/L4	M4	Yes	Yes	Yes(2)	Yes	6	Yes	2(2)
F7/H7	M7	Yes	Yes	Yes(2)	Yes	8	Yes	2(2)

1. Microcontroller Clock Output (refer to *Section 8.2: Microcontroller clock output (MCO) on page 87*)
2. Depends on package size. Check availability in the Pin Allocation Table in the related datasheet.