Clock sources and configure peripheral clock

Understand about the clock source and the frequency. Select the main clock source and its clock tree to enable, disable, and configure the clock on peripherals.

#arm #stm32 #clock

Last update: April 28, 2021

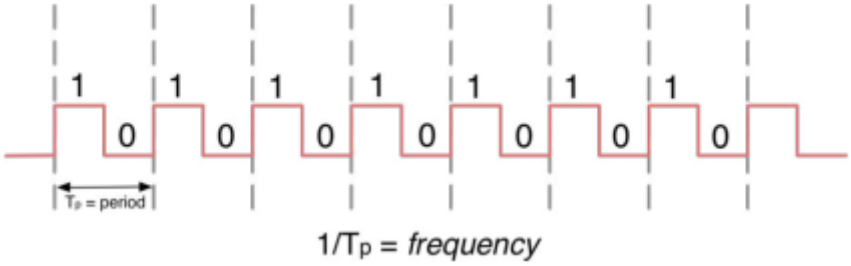
Table of Content

1. Clock source

Almost every digital circuit needs a way to synchronize its internal circuitry or to synchronize itself with other circuits.

A *clock* is a device that usually generates a periodical square wave signal, with a 50% duty cycle. A clock signal oscillates between V_L and V_H voltage levels, which for STM32 microcontrollers are a fraction of the V_{DD} supply voltage.

The most fundamental parameter of a clock is the frequency, which indicates how many times it switches from V_L to V_H in a second. The frequency is expressed in *Hertz*.



A typical clock signal

The majority of STM32 MCUs can be clocked by two distinct clock sources alternatively:

- an *internal RC oscillator* (**High Speed Internal** (HSI)) or
- an *external dedicated crystal oscillator* (**High Speed External** (HSE)).

There are several reasons to prefer an external crystal to the internal RC oscillator:

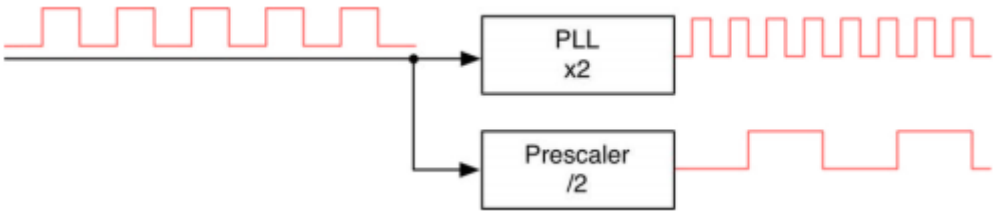
- An external crystal offers a higher precision compared to the internal RC network, which is rated of a 1% accuracy, especially when PCB operative temperatures are far from the ambient temperature of 25°C
- Some peripherals, especially high speed ones, can be clocked only by a dedicated external crystal running at a given frequency

Together with the high-speed oscillator, another clock source can be used to bias the low-speed oscillator, which in turn can be clocked by:

- an external crystal (**Low Speed External** (LSE)) or
- the internal dedicated RC oscillator (**Low Speed Internal** (LSI)).

The low-speed oscillator is used to drive the *Real Time Clock* (RTC) and the *Independent Watchdog* (IWDT) peripheral.

Using several **Programmable Phase-Locked Loops** (PLL) and pre-scalers, it is possible to increase/decrease the source frequency at needs, depending on the requested performances, the maximum speed for a given peripheral or bus and the overall global power consumption.



PLL is used to increased/ decrease clock frequency

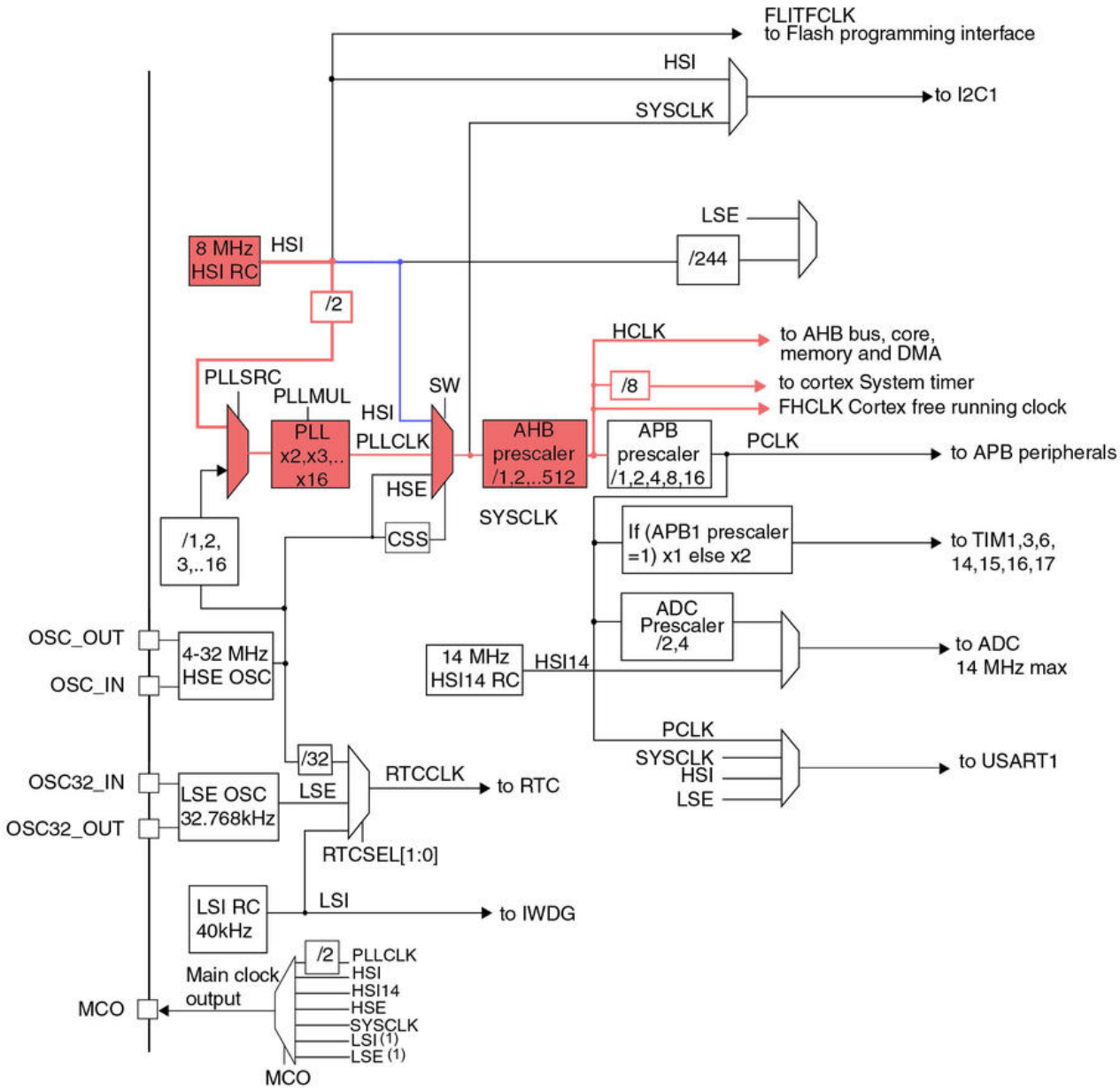
2. Clock tree

The clock tree configuration is performed through a dedicated peripheral named **Reset and Clock Control** (RCC), and it is a process essentially composed by three steps:

1. The high-speed oscillator source is selected (HSI or HSE) and properly configured, if the HSE is used.
2. If need to feed the `SYSCLK` with a frequency higher than the one provided by the high-speed oscillator, then configure the main PLL (which provides the `PLLCLK` signal).
3. The System Clock Switch (SW) is configured to choose the system clock source from HSI, HSE, or `PLLCLK`. Then select the AHB, APB1 and APB2 (if available) pre-scaler settings to reach the wanted frequency of the High-speed clock (`HCLK` - that is the one that feeds the core, DMAs and AHB bus),and the frequencies of Advanced Peripheral Bus 1 (APB1) and APB2 (if available) buses.

SysTick

The System Tick Time `SysTick` generates interrupt requests on a regular basis. This allows an OS to carry out context switching to support multiple tasking. For applications that do not require an OS, the SysTick can be used for time keeping, time measurement, or as an interrupt source for tasks that need to be executed regularly.



Example of a clock tree

3. Get Clock Frequency

Sometimes it is important to know how fast the CPU core is running. The CubeHAL provides a function that can be used to compute the SYSCLK frequency: the `HAL_RCC_GetSysClockFreq()`. However, this function must be handled with special care.

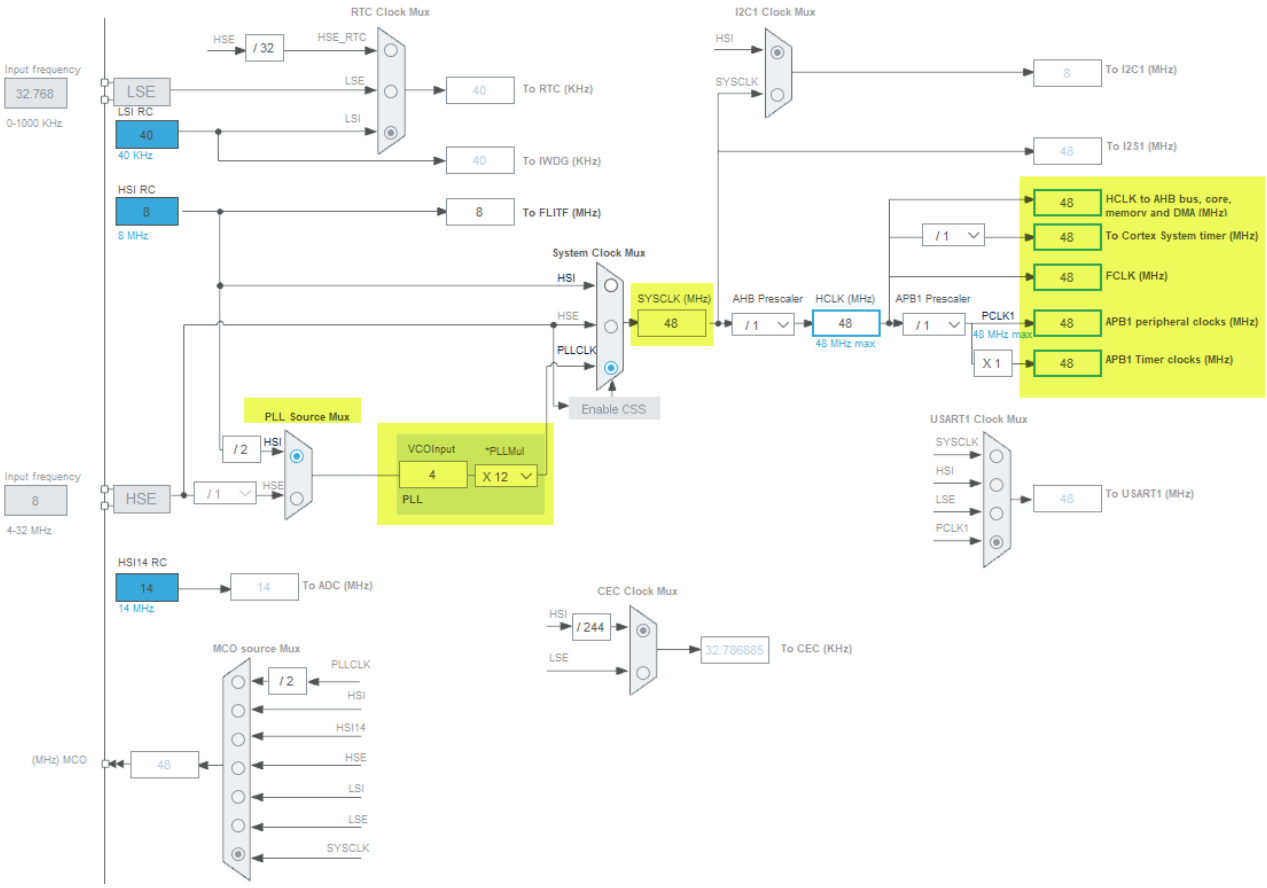
The `HAL_RCC_GetSysClockFreq()` does not return the real `SYSCLK` frequency (it could never do this in a reliable way without having a known and precise external reference), but it bases the result on the following algorithm:

- if `SYSCLK` source is the *HSI* oscillator, then returns the value based on the `HSI_VALUE` macro;
- if `SYSCLK` source is the *HSE* oscillator, then returns the value based on the `HSE_VALUE` macro;
- if `SYSCLK` source is the *PLLCLK*, then returns a value based on `HSI_VALUE` or `HSE_VALUE` multiplied by the `PLL` factor, according the specific STM32 MCU implementation.

`HSI_VALUE` and `HSE_VALUE` macros are defined inside the `stm32xxx_hal_conf.h` file, and they are hardcoded values. The `HSI_VALUE` is defined by ST during chip design, with 1% of accuracy. `HSE_VALUE` macro is defined by programmer, so it could be wrong.

4. Configure clock

The clock path starts from the internal 8MHz oscillator. If the microcontroller relies on an external HSE/LSE crystal, then clock source has to be enabled in the RCC peripheral before it can be used as the main clock source.

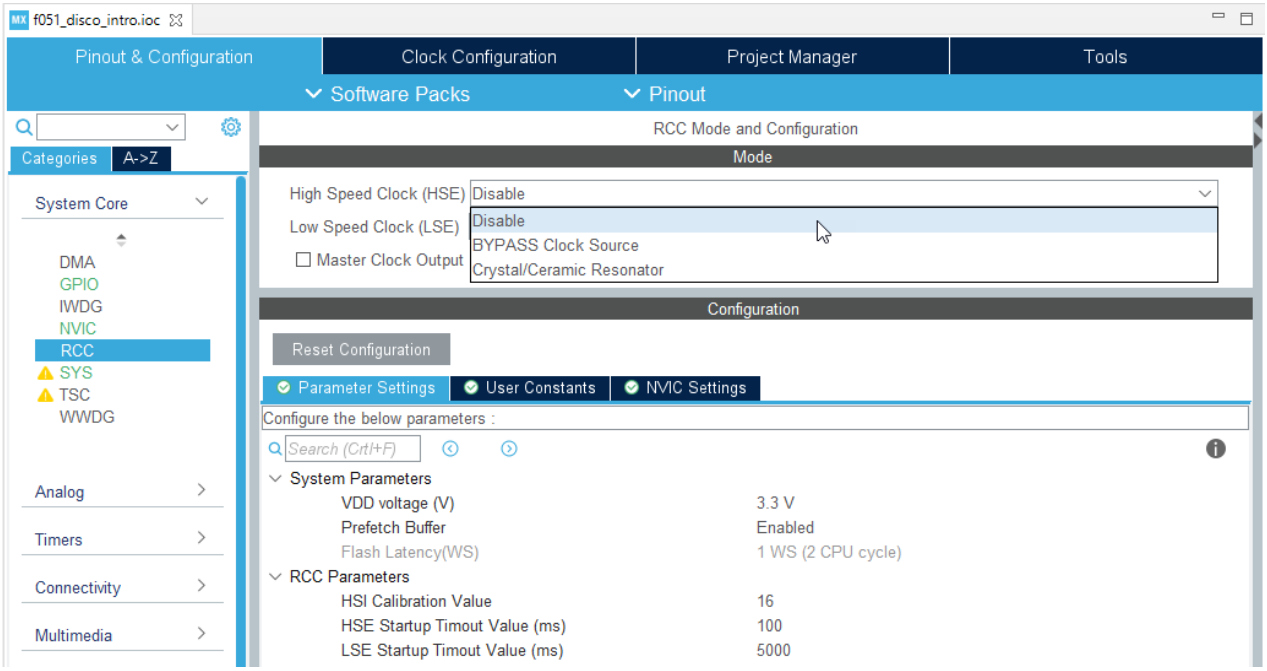


Clock tree of STM32F0

4.1. External clock source

For both `HSE` and `LSE` oscillators, CubeMX offers three configuration options:

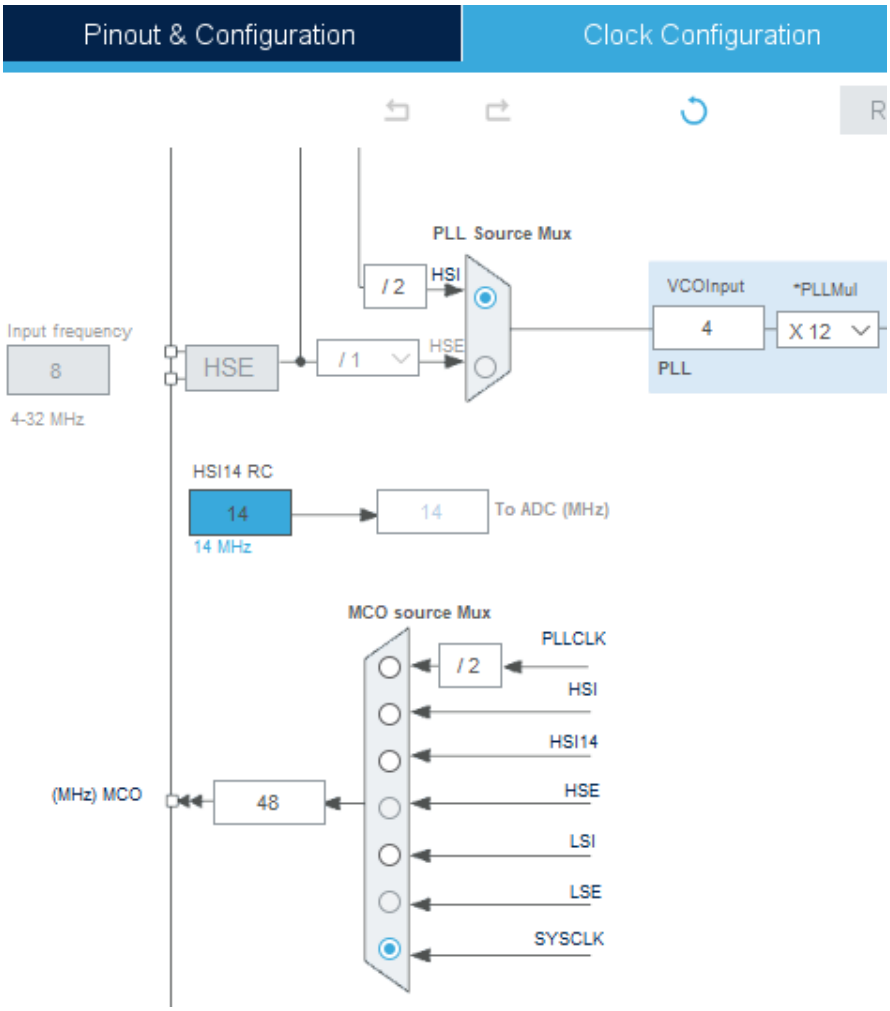
- **Disable**: the external oscillator is not available/used, and the corresponding internal oscillator is used.
- **Crystal/Ceramic Resonator**: an external crystal/ceramic resonator is used and the corresponding main frequency is derived from it. This implies that `RCC_OSC_IN` and `RCC_OSC_OUT` pins are used to interface the HSE, and the corresponding signal I/Os are unavailable for other usages (if use an external low-speed crystal, then the corresponding `RCC_OSC32_IN` and `RCC_OSC32_OUT` I/Os are used too).
- **BYPASS Clock Source**: an external clock source is used. The clock source is generated by another active device. This means that the `RCC_OSC_OUT` is leaved unused, and it is possible to use it as regular GPIO. In almost all development board from ST (included the Nucleo ones) the Master Clock Output (MCO) pin of the ST-LINK interface is used as external clock source for the target STM32 MCU. Enabling this option allows to use the ST-LINK MCO as HSE.



Enable external clock source

4.2. Master Clock Output

The RCC peripheral also allows to enable the *Master Clock Output* (MCO), which is a pin that can be connected to a clock source. It can be used to clock another external device, allowing to save on the external crystal for this other IC. Once the MCO is enabled, it is possible to choose its clock source using the Clock Configuration view.



Select clock source to be output

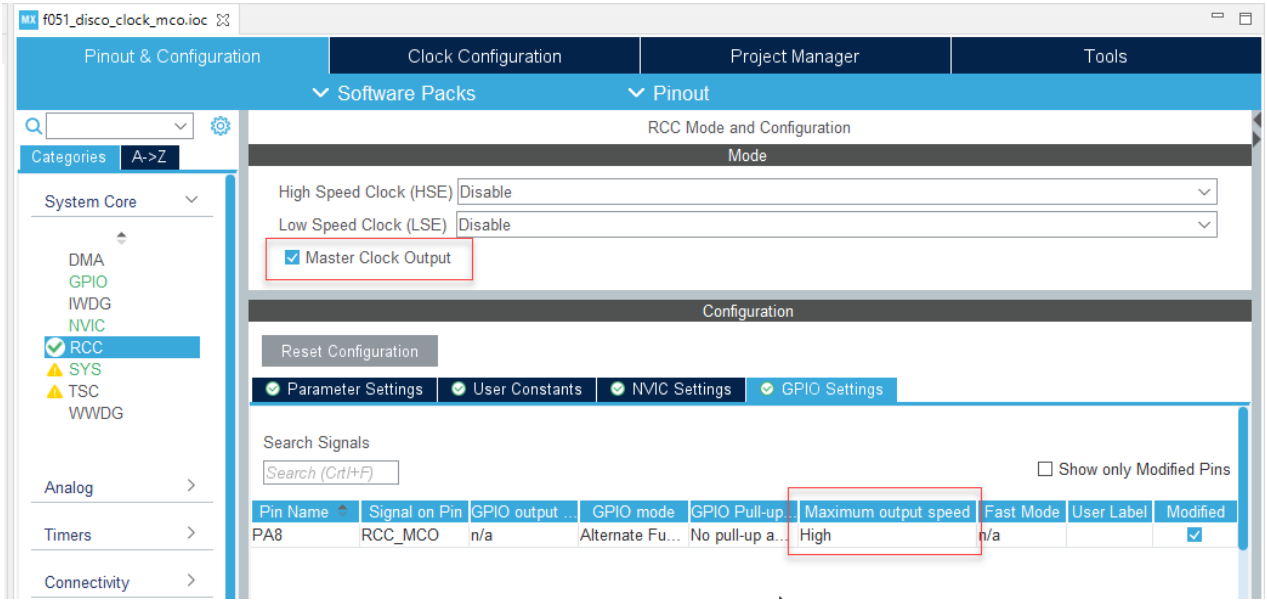
5. HAL Software

Reset and Clock Control (RCC) peripheral is responsible of the configuration for the whole clock tree of an STM32 MCU. The most relevant C struct to configure the clock tree are `RCC_OscInitTypeDef` and `RCC_ClkInitTypeDef`.

CubeMX is designed to generate the right code initialization for the clock tree of our MCU. All the necessary code is packed inside the `SystemClock_Config()` routine in the file *main.c*.

To configure the Master Clock Output, use the function `HAL_RCC_MCOConfig()` to select the clock source and its dividing factor.

Note that when configuring the MCO pin as output GPIO, its speed (that is, the slew rate) affects the quality of the output clock.



Enable MCO ouput

Above config should give a generated code as below:

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters in
the RCC_OscInitTypeDef structure.*/
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK |RCC_CLOCKTYPE_SYSCLK |
RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK) {
    Error_Handler();
  }

  HAL_RCC_MCOConfig(RCC_MCO, RCC_MCO1SOURCE_SYSCLK, RCC_MCODIV_1);
}
```

6. Clock Security System

The Clock Security System (CSS) is a feature of the RCC peripheral used to detect malfunctions of the external HSE. The CSS is an important feature in some critical applications, and the detection of a failure is noticed through the NMI exception, a Cortex-M exception that cannot be disabled.

When the failure of HSE is detected, the MCU automatically switch to the HSI clock, which is selected as source for the SYSCLK clock. So, if a higher core frequency is needed, it needs to perform proper initializations inside the NMI exception handler.

`HAL_RCC_EnableCSS()` function will enable the CSS feature, and define a handling callback:

```
void HAL_RCC_CSSCallback(void) {
  //Catch the HSE failure and take proper actions
}
```

which is called from the NMI interrupt handler:

```
void NMI_Handler(void) {
  HAL_RCC_NMI_IRQHandler();
}
```