# Clocks

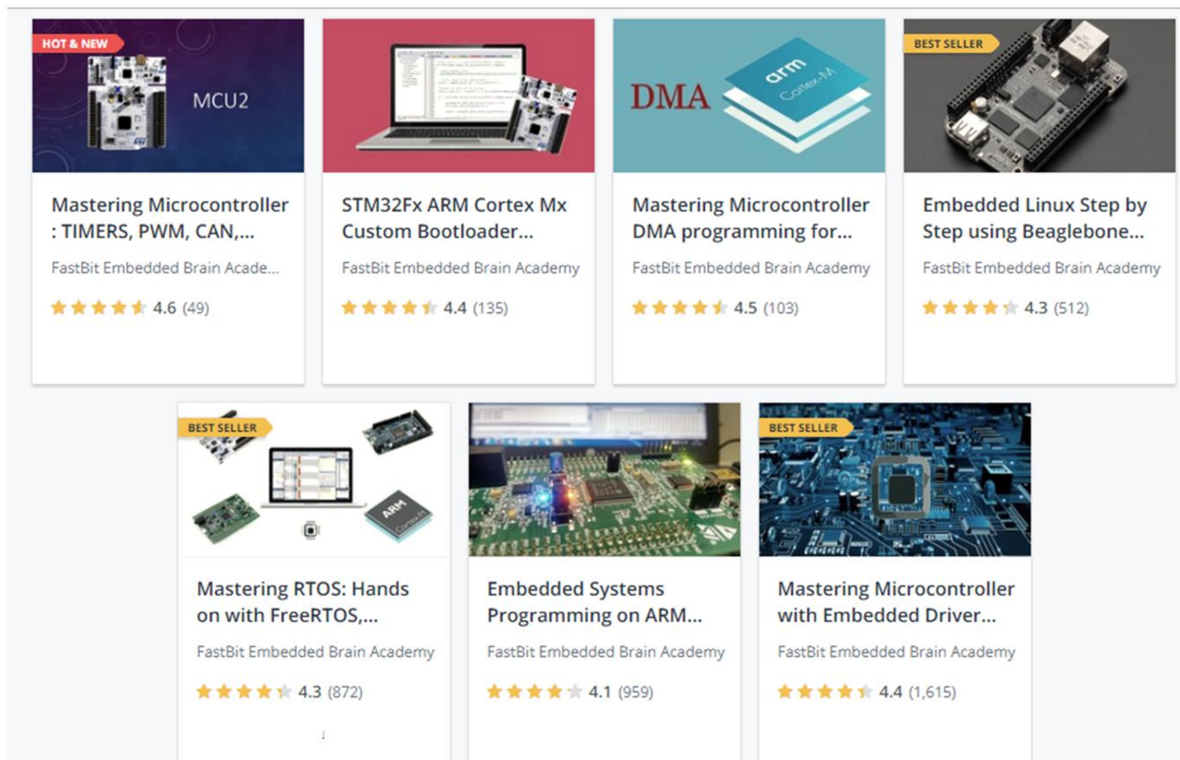
Understanding HSI,HSE,PLL,HCLK,PCLKx and configuration

# For full video course on Microcontroller and RTOS programming please visit : www.fastbitlab.com

**All courses are hosted on Udemy.com**

| Total students | Courses | Reviews |
| --- | --- | --- |
| 16,226 | 7 | 4,245 |

Udemy



Gift this course

# Mastering Microcontroller DMA programming for Beginners

Direct Memory Access Demystified with STM32 Peripherals (ADC, SRAM,UART,M2M,M2P,P2M) and Embedded C code Exercises

4.5 (103 ratings) 1,240 students enrolled

Created by FastBit Embedded Brain Academy Last updated 10/2018

English English [Auto-generated]


**Click here to enroll FREE !!!!**

http://bit.ly/2P47IeX

## Includes

- 9.5 hours on-demand video
- 8 articles
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

Gift this course

# Mastering RTOS: Hands on with FreeRTOS, Arduino and STM32Fx

Learn Running/Porting FreeRTOS Real Time Operating System on Arduino, STM32F4x and ARM cortex M based Mircocontrollers

BESTSELLER 4.2 (866 ratings) 5,480 students enrolled

Created by FastBit Embedded Brain Academy Last updated 10/2018

English English [Auto-generated], Portuguese [Auto-generated], 1 more


**Click here to watch free preview and enroll**

http://bit.ly/2PAQRQh

### Includes

- 16 hours on-demand video
- 28 articles
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

### Interactive Features

- 17 downloadable resources

Gift this course

# Embedded Linux Step by Step using Beaglebone Black

Learn ARM Linux systems, Embedded Linux building blocks , Beaglebone interfacing Projects and much more

BESTSELLER ★★★★★ 4.3 (509 ratings) 3,084 students enrolled

Created by FastBit Embedded Brain Academy Last updated 10/2018

English English [Auto-generated]


**Click here to watch free preview and enroll**

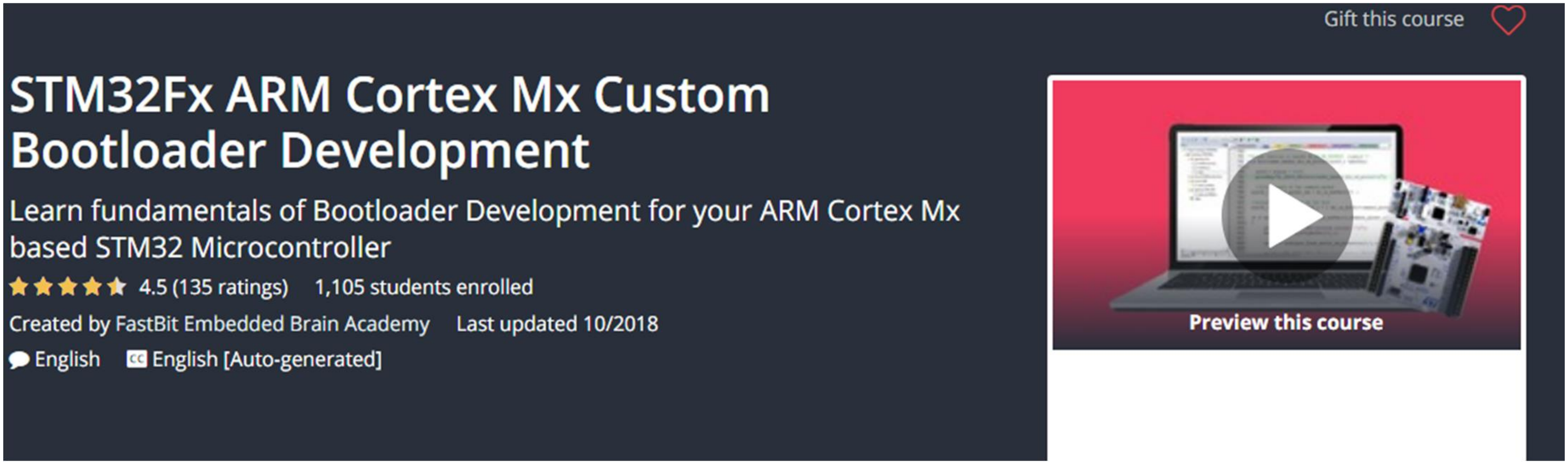
http://bit.ly/2AEcneA

**Includes**

- 15.5 hours on-demand video
- 24 articles
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

**Interactive Features**

- 9 downloadable resources




**Click here to watch free preview and enroll**

http://bit.ly/2zhqDrX

**Includes**

- 7.5 hours on-demand video
- 6 articles
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

**Interactive Features**

- 1 downloadable resource

# Embedded Systems Programming on ARM Cortex-M3/M4 Processor

With hands on Coding using C Programming and assembly on ARM Cortex M Processor based Microcontroller

4.1 (958 ratings) 6,257 students enrolled

Created by FastBit Embedded Brain Academy Last updated 10/2018

English English [Auto-generated], Spanish [Auto-generated]


**Click here to watch free preview and enroll**

http://bit.ly/2OjgFf7

### Includes

- 11.5 hours on-demand video
- 12 articles
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

### Interactive Features

- 2 downloadable resources

# System Clock(SYSCLK)

- Three different clock sources can be used to drive the system clock (SYSCLK):
  - HSI oscillator clock
  - HSE oscillator clock
  - Two main PLL (PLL) clocks
- The devices have the two following secondary clock sources
  - 32 kHz low-speed internal RC (LSI RC) which drives the independent watchdog and, optionally, the RTC used for Auto-wakeup from the Stop/Standby mode.
  - 32.768 kHz low-speed external crystal (LSE crystal) which optionally drives the RTCclock (RTCCLK)

# NUCLEO-F446RE board

- HSI → 16MHz (Internal to MCU)
- HSE→ 8MHz (External to MCU)
- PLL can generate clock up to 180MHz (Internal to MCU)
- LSI→32kHz (Internal to MCU)
- LSE→32.768kHz (External to MCU)

# Default Clock State

After reset of the MCU,

HSI is ON, HSE is OFF , PLL is OFF, LSE is OFF,

LSI is OFF

So, SYSCLK is sourced by HSI .

I.e : SYSCLK = 16MHz

**Each clock source can be switched on or off independently when it is not used, to optimize power consumption.**

# HSI

✓The HSI is used (enabled by hardware) as system clock source after startup from Reset, wake-up from STOP and STANDBY mode, or in case of failure of the HSE used directly or indirectly as system clock

✓The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components).

✓It also has a faster startup time than the HSE crystal oscillator.

✓However, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator

✓The HSI signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails.

# STM32 Cube Clock Handling APIs

STM32 HAL RCC Driver

Source

stm32f4xx_hal_rcc.c
stm32f4xx_hal_rcc_ex.c

*ex = Extension*

header

stm32f4xx_hal_rcc.h
stm32f4xx_hal_rcc_ex.h

# Methods to configure the STSCLK Source

- First Enable the required clock and wait until the clock is ready . If your application needs PLL, then configure the PLL and enable it.
- Initializes the CPU, AHB and APB busses clock prescalers according to your application requirements . Do not cross maximum limits.
- Configure the flash latency properly by referring to MCU RM
- Select newly enabled Clock as SYSCLK

# STM32 Cube Clock Handling APIs

**1) HAL_RCC_OscConfig(RCC_OscInitTypeDef  *RCC_OscInitStruct)**

**2) HAL_RCC_ClockConfig(RCC_ClkInitTypeDef  *RCC_ClkInitStruct, uint32_t FLatency)**

# RCC_OscInitTypeDef

```
typedef struct
{
  uint32_t OscillatorType;

  uint32_t HSEState;

  uint32_t LSEState;

  uint32_t HSIState;

  uint32_t HSICalibrationValue;

  uint32_t LSIState;

  RCC_PLLInitTypeDef PLL;
}RCC_OscInitTypeDef;
```

***RCC Internal/External Oscillator (HSE, HSI, LSE and LSI) configuration structure definition***

# RCC_ClkInitTypeDef

```
typedef struct
{
  uint32_t ClockType;


  uint32_t SYSCLKSource;


  uint32_t AHBCLKDivider;


  uint32_t APB1CLKDivider;


  uint32_t APB2CLKDivider;


}RCC_ClkInitTypeDef;
```

***RCC System, AHB and APB busses clock configuration structure definition***

# HSI Calibration

**The operating temperature has an impact on the accuracy of the RC oscillators. At 25 °C, the HIS oscillators have an accuracy of ±1% typically, but in the temperature range of -40 to 105 °C, the accuracy decreases.**

**To compensate for the influence of temperature in the application, the output frequency of the HSI oscillator can be further trimmed by the user runtime calibration routine to improve the HSI frequency accuracy. This may prove crucial for communication peripherals.**

# HSI Calibration adjustment

## 6.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

**HSI RC oscillators are factory calibrated by ST to have a 1% accuracy at TA = 25 °C. After reset, the factory calibration value is automatically loaded in the internal calibration bits.**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | PLLSAI RDY | PLLSAI ON | PLLI2S RDY | PLLI2S ON | PLL RDY | PLL ON | Res. | Res. | Res. | Res. | CSS ON | HSE BYP | HSE RDY | HSE ON |
| | | r | rw | r | rw | r | rw | | | | | rw | rw | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HSICAL[7:0] | | | | | | | | HSITRIM[4:0] | | | | | Res. | HSI RDY | HSI ON |
| r | r | r | r | r | r | r | r | rw | rw | rw | rw | rw | | r | rw |

# HSI Calibration adjustment

the calibration value is loaded in HSICAL[7:0] bits after reset. Five trimming bits HSITRIM[4:0] are used for fine-tuning. The default trimming value is 16

The frequency of the internal RC oscillators can be fine-tuned to achieve better accuracy with wider temperature and supply voltage ranges. The trimming bits are used for this purpose.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | PLLSAI RDY | PLLSAI ON | PLLI2S RDY | PLLI2S ON | PLL RDY | PLL ON | Res. | Res. | Res. | Res. | CSS ON | HSE BYP | HSE RDY | HSE ON |
| | | r | rw | r | rw | r | rw | | | | | rw | rw | r | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSICAL[7:0] | | | | | | | | HSITRIM[4:0] 16 | | | | | Res. | HSI RDY | HSI ON |
| r | r | r | r | r | r | r | r | rw | rw | rw | rw | rw | | r | rw |

# HSI Calibration adjustment

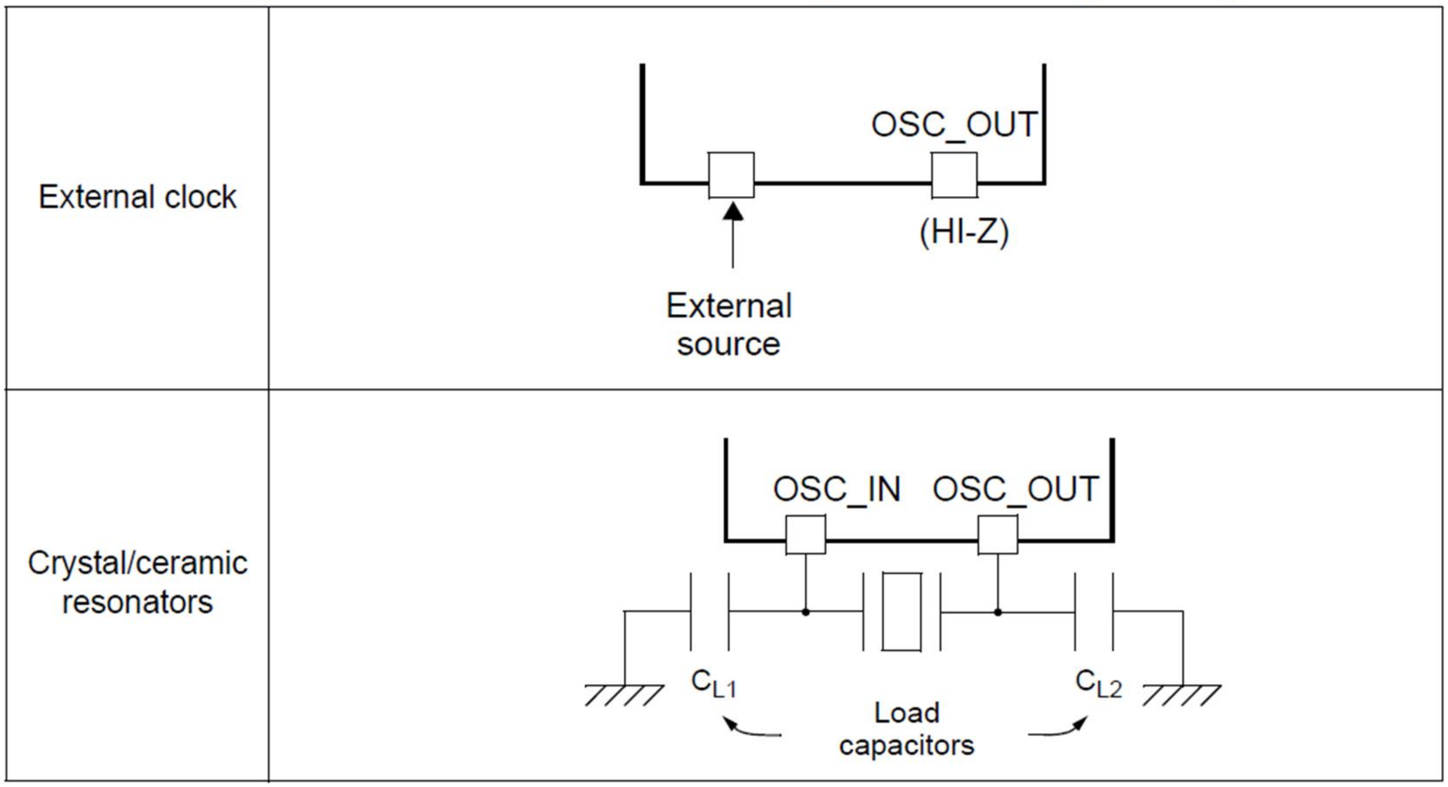
- Effect of HSITRIM[4:0]
  - The default trimming value is 16
  - An increase in this trimming value causes an increase in HSI frequency
  - Decrease in this trimming value causes an decrease in HSI frequency
  - The HSI oscillator is fine-tuned in steps of 0.5% (around 80 kHz)

# HSI Calibration adjustment

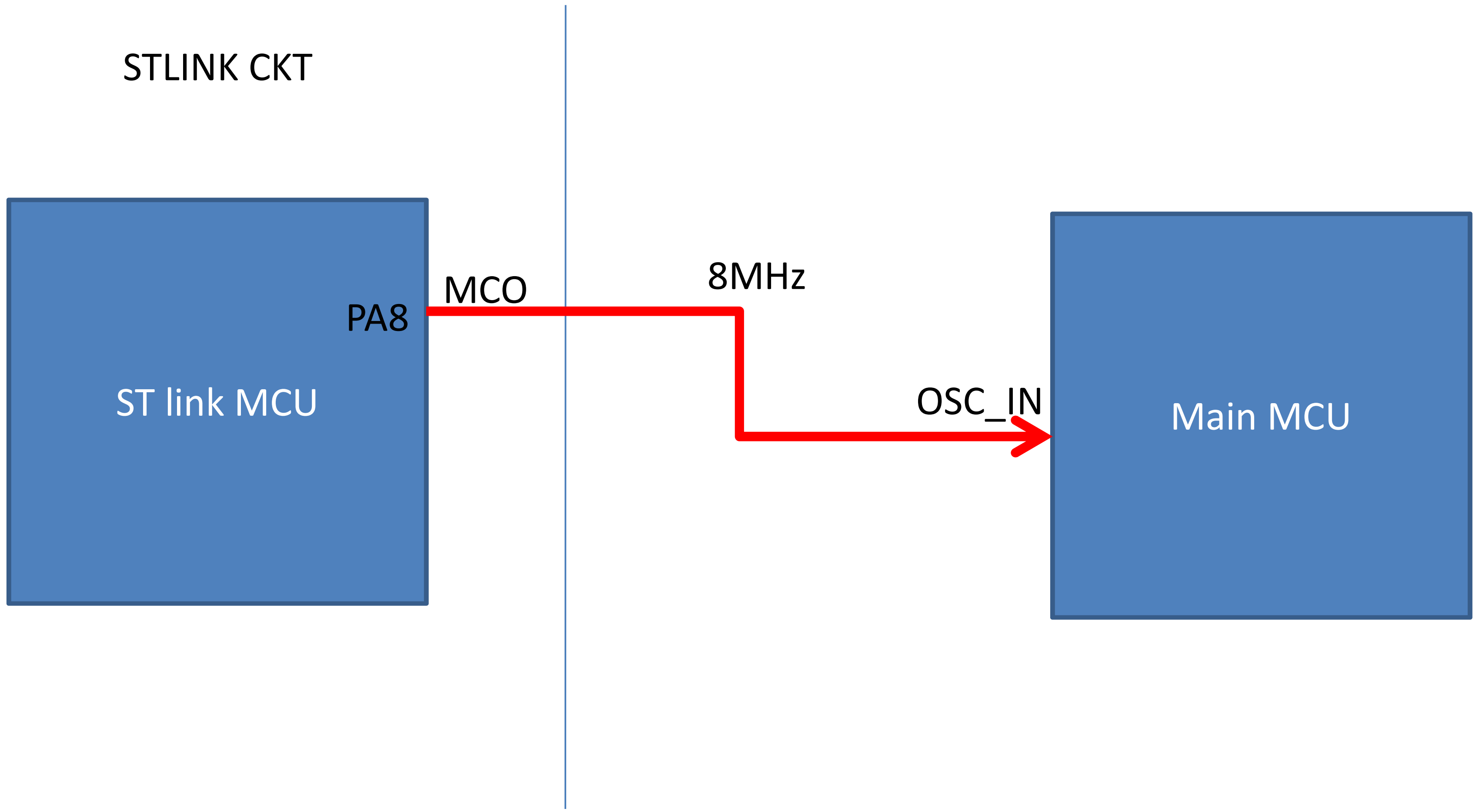
- Summary
  - Writing a trimming value in the range of 17 to 31 increases the HSI frequency.
  - Writing a trimming value in the range of 0 to 15 decreases the HSI frequency
  - Writing a trimming value equal to 16 causes the HSI frequency to keep its default value. (+- 1%)

# HSE BYPASS

**Figure 15. HSE/ LSE clock sources (hardware configuration)**



# HSE BYPASS-NUCLEO-F446RE



# Exercise

Using HSE Configure the SYSCLK as 8MHz.

AHB clock as 4MHz (HCLK)

APB1 clock as 2MHz (PCLK1)

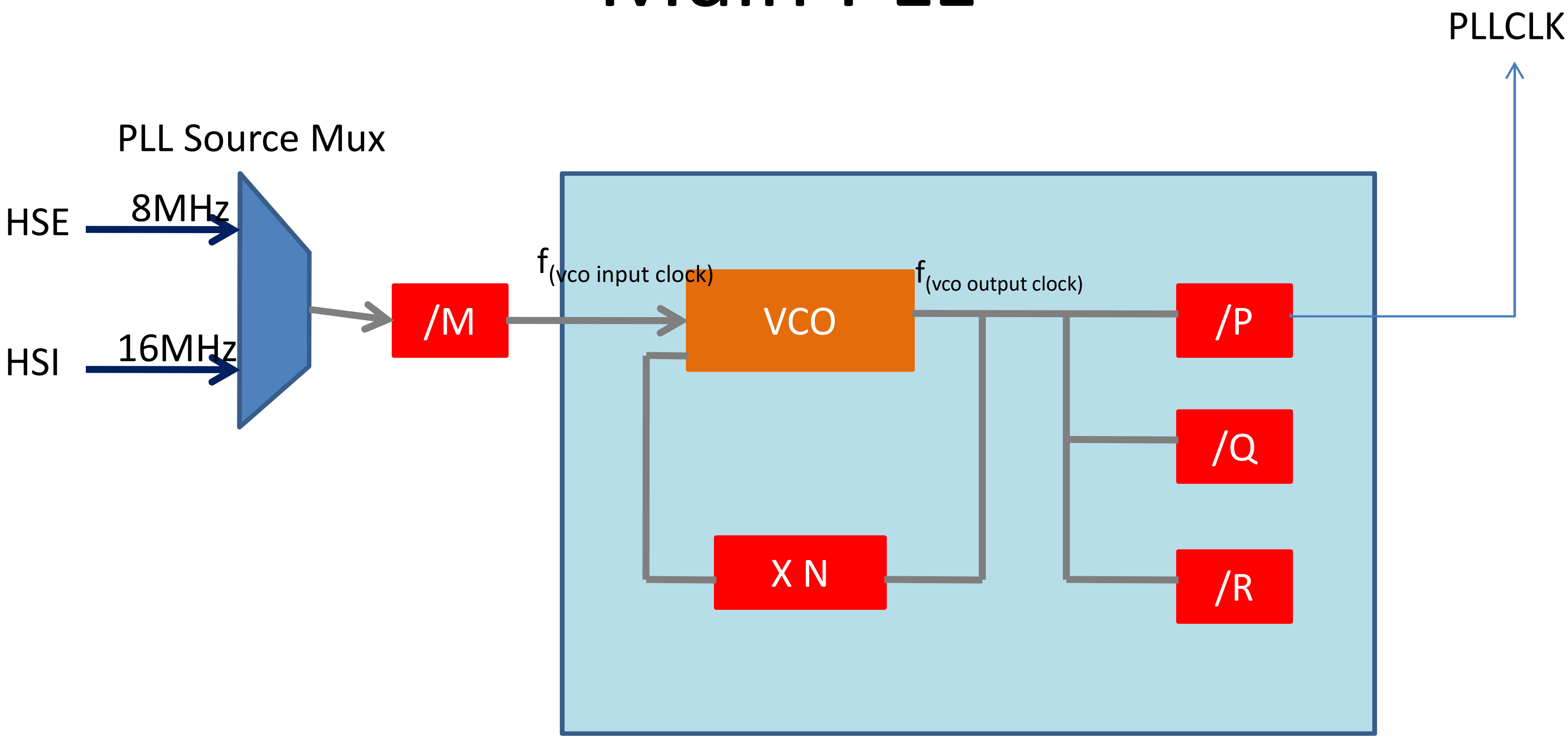
APB2 clock as 2MHz (PCLK2)

# PLL(Phase Locked Loop)

- The PLL engine of the MCU is used to generate different high frequency output clocks by taking input clock sources such as HSE or HSI.

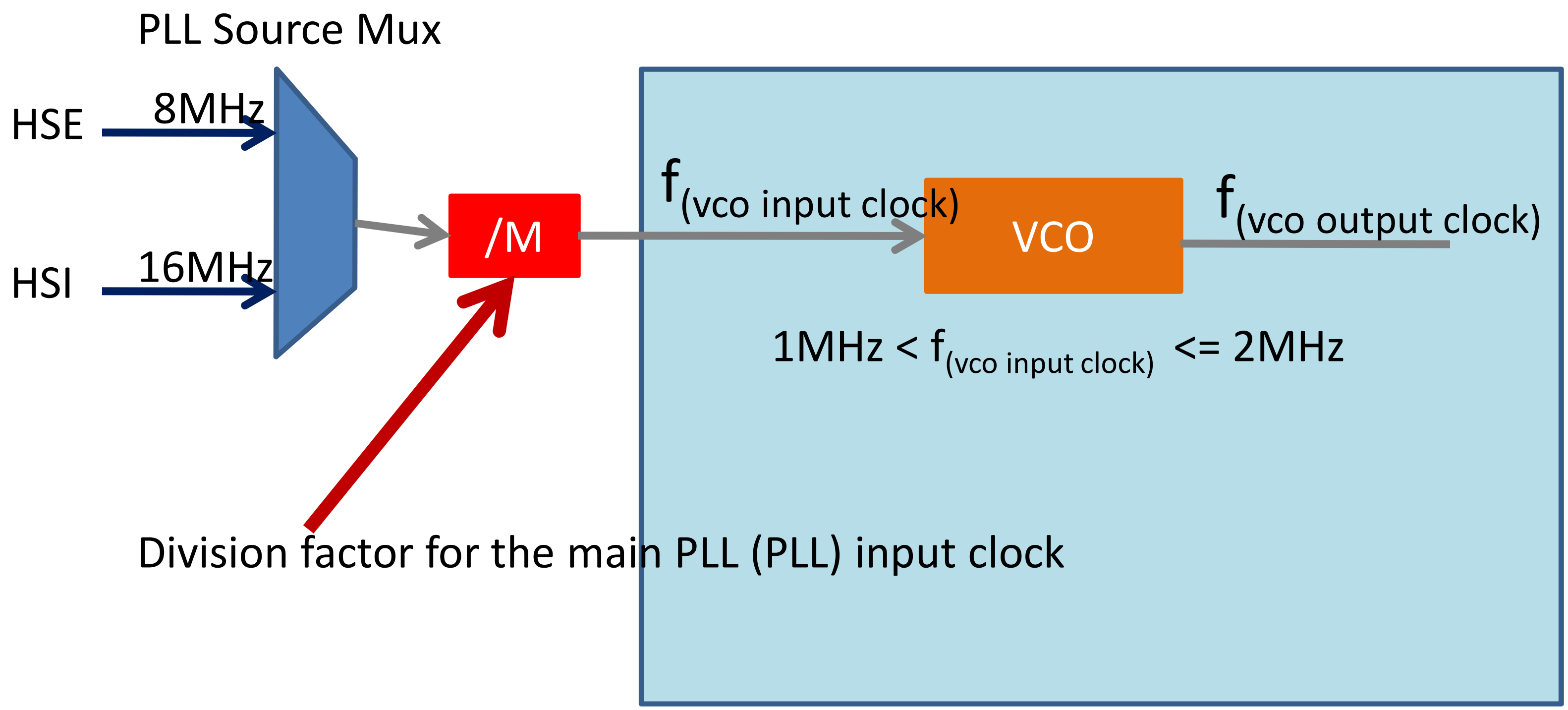
- By using PLL you can drive SYSCLK up to 180MHz in STM32F446RE MCU

# Main PLL

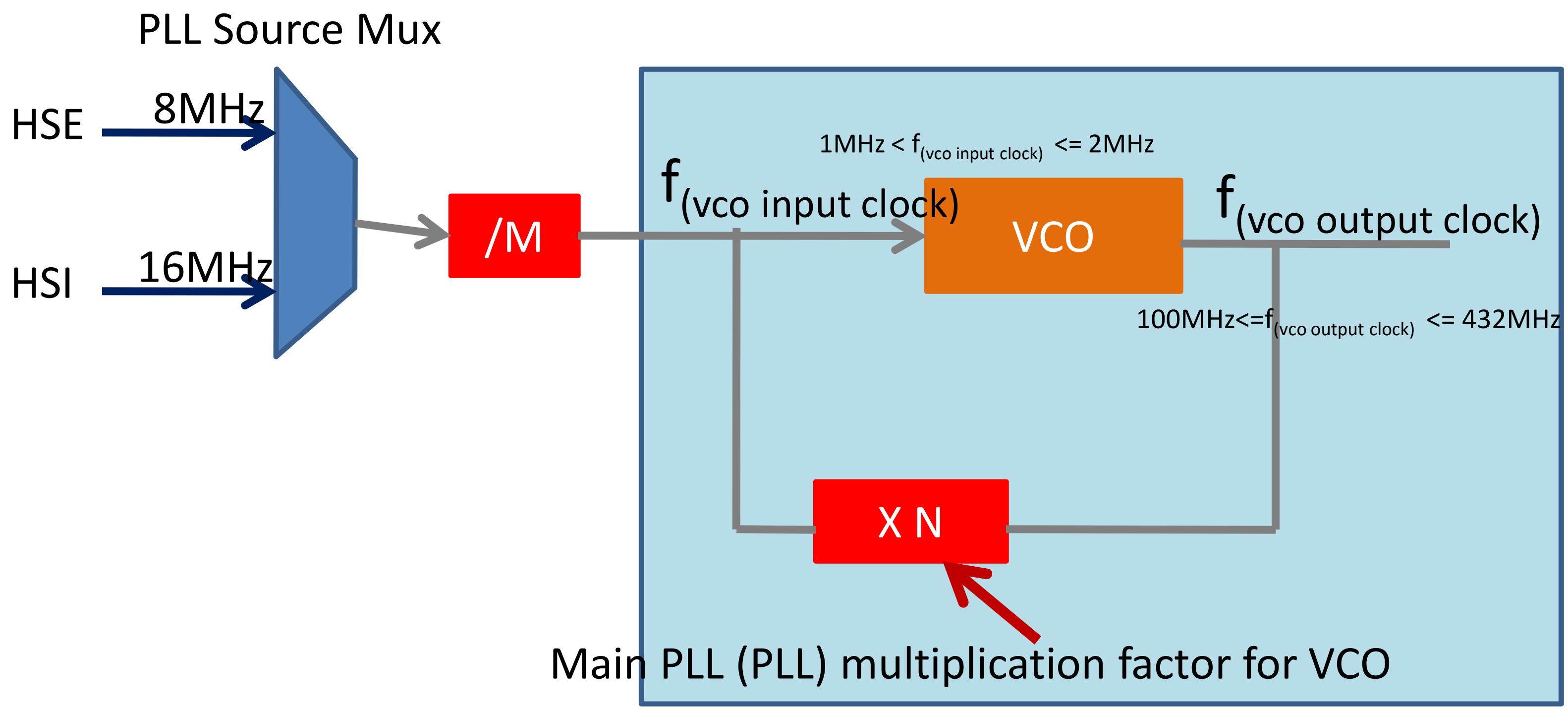


PLLCLK
PLL Source Mux
HSE
8MHz
HSI
16MHz
/M
f(vco input clock)
VCO
f(vco output clock)
/P
/Q
/R
X N

# Main PLL


PLL Source Mux
HSE
8MHz
HSI
16MHz
/M
f(vco input clock)
VCO
f(vco output clock)
1MHz < f(vco input clock) <= 2MHz
Division factor for the main PLL (PLL) input clock

# Main PLL


PLL Source Mux
HSE
8MHz
HSI
16MHz
/M
1MHz < f(vco input clock) <= 2MHz
f(vco input clock)
VCO
f(vco output clock)
100MHz<=f(vco output clock) <= 432MHz
X N
Main PLL (PLL) multiplication factor for VCO

# PLL Formulas

$$f_{(\text{vco output clock})} = \left[\frac{f_{(\text{vco input clock})}}{\text{PLLM}}\right] \times \text{PLLN}$$

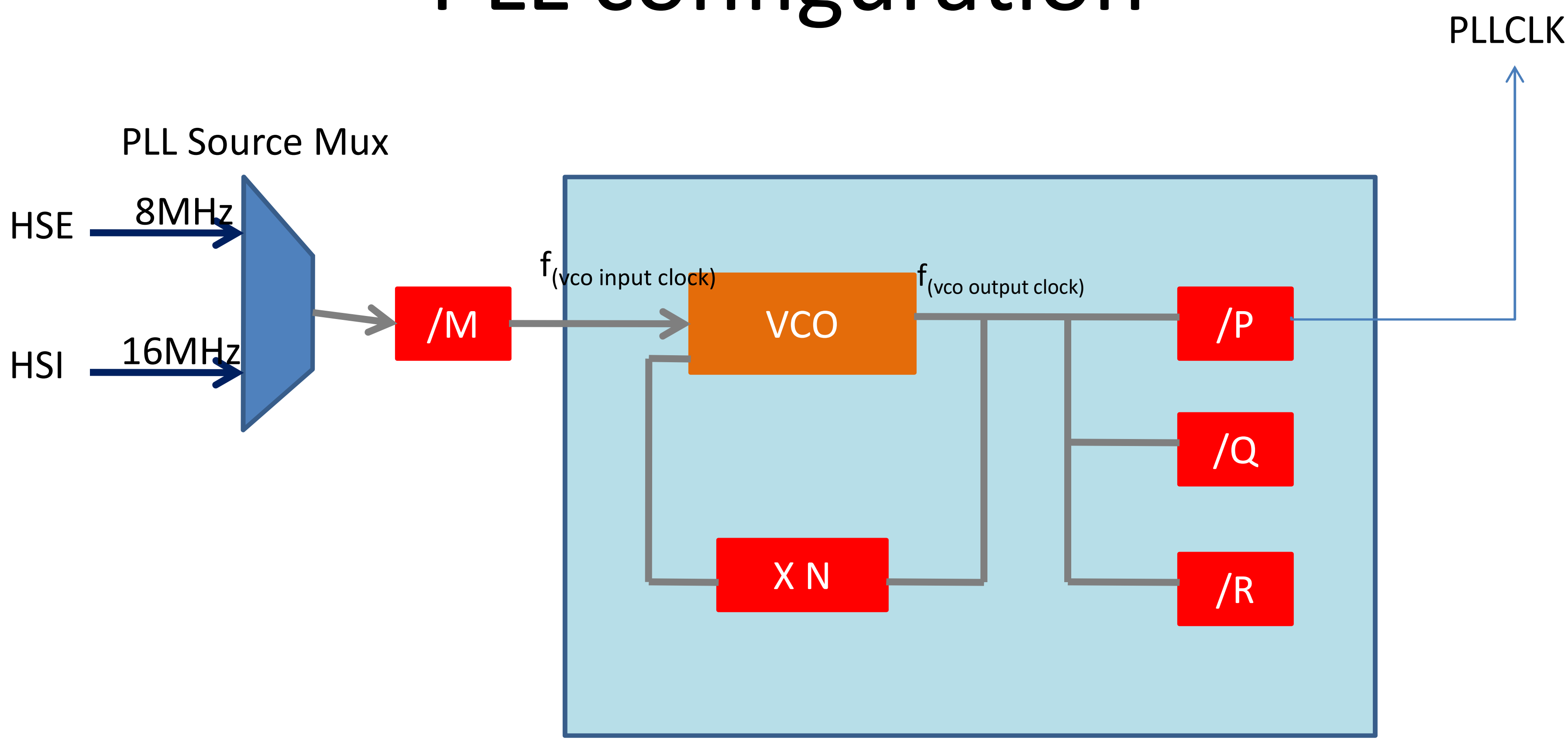
$$f_{(\text{PLL general clock output})} = \frac{f_{(\text{vco output clock})}}{\text{PLLP}}$$

**PLLCLK**

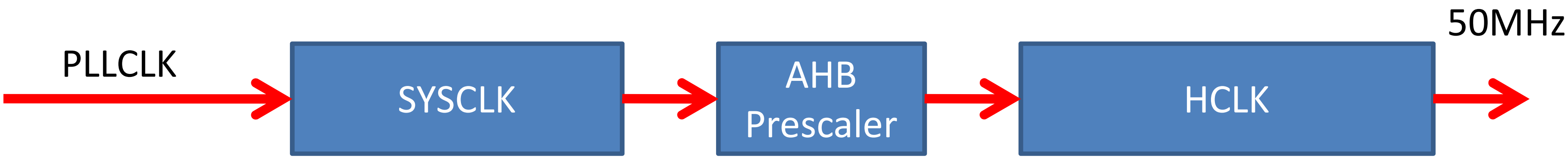
# Exercise

- Write an application to generate below HCLK Frequencies  using PLL . Use HSI as PLL's input source and repeat the exercise using HSE as input source.
  - 50MHz
  - 84MHz
  - 120MHz

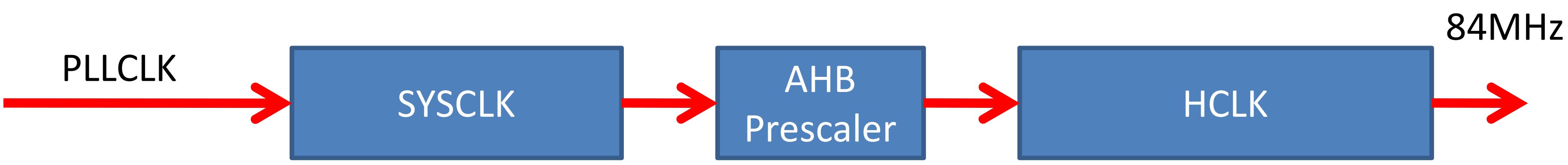
# PLL configuration



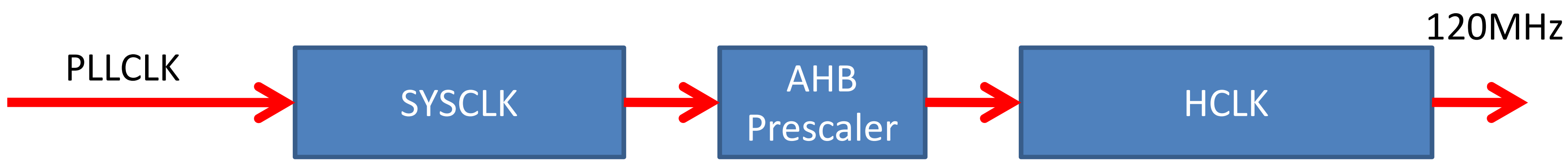
# PLL configuration



# PLL configuration



# PLL configuration



# Exercise

- Write an application which does PLL configuration to boost the HCLK to maximum capacity (for STM32F446RE it is:180MHz). Use HSE as PLL Source.