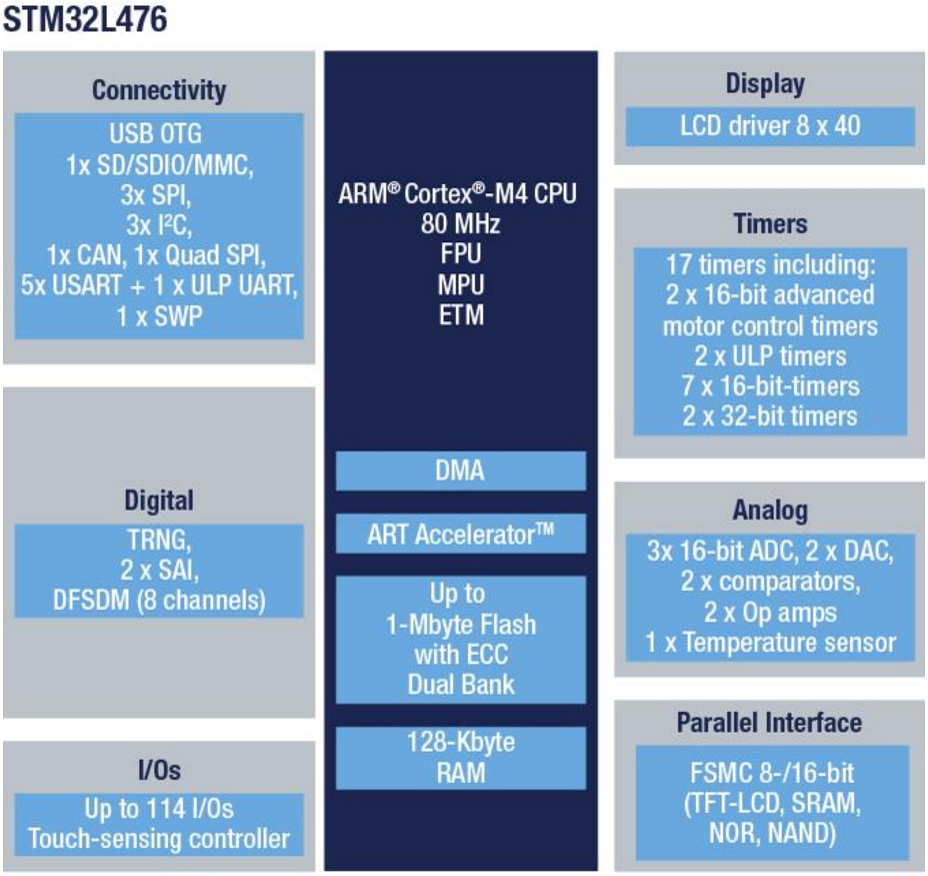
Appendix A: Microcontroller STM32L476

The NUCLEO-L476RG board has a 32-bit Arm Cortex-M4F microcontroller. This is used in our text book.

- Part Number: **STM32L476RGT6**
- Total number of pins: 64
- Total number of I/O pins: 51
- Maximum clock frequency: 80MHz
- Program memory size: 1 MB
- Operating supply voltage: 1.71V to 3.6V
- I/O voltage: 3.3V
- Analog supply voltage: 3.3V

The following figure summarizes the I/O connection supported and internal peripherals.



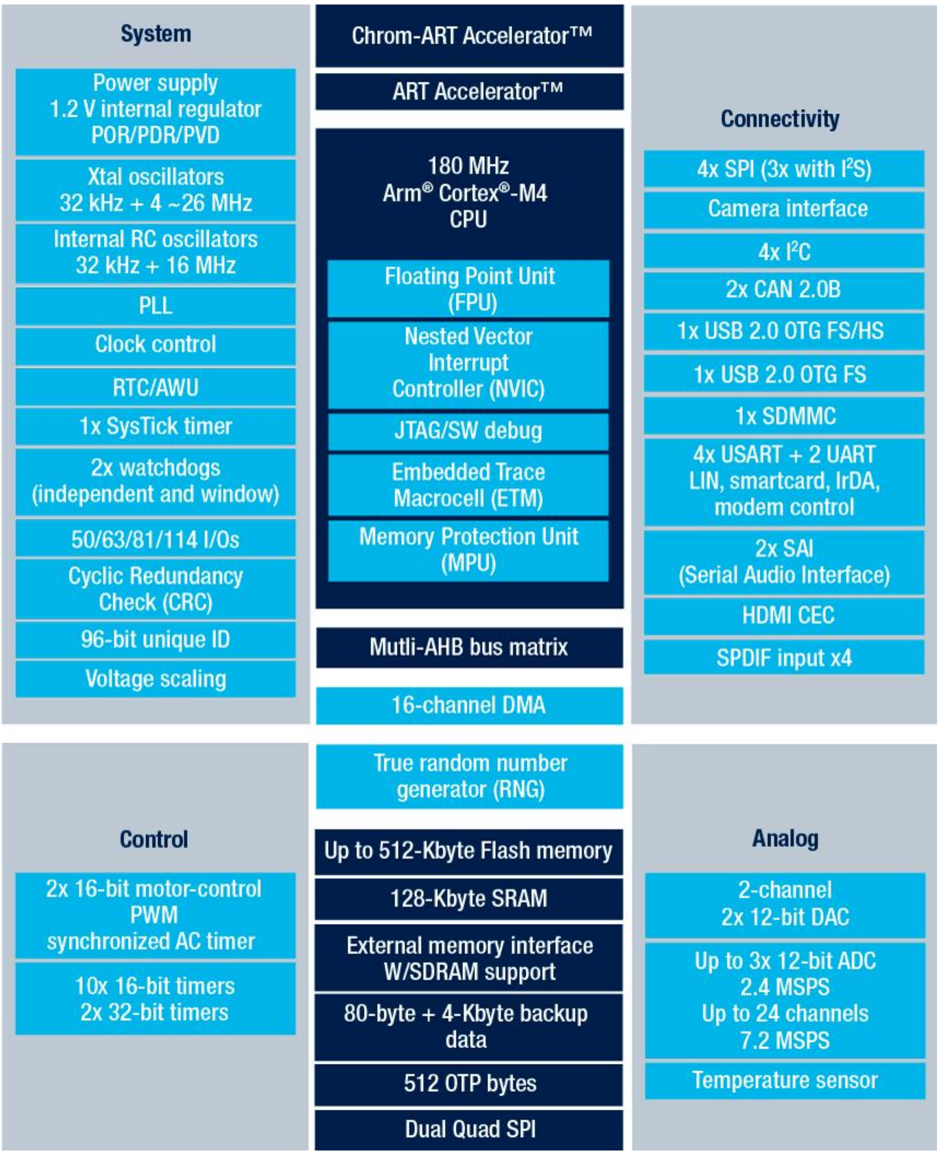
Appendix B: Microcontroller STM32F446

The NUCLEO-L446RE board has a 32-bit Arm Cortex-M4F microcontroller.
This board is used in BUET lab.

- Part Number: **STM32F446RE**
- Total number of pins: 64
- Total number of I/O pins: 51
- Maximum clock frequency: **180MHz**
- Program memory size: **512 kB**
- Operating supply voltage: 1.71V to 3.6V
- I/O voltage: **1.7-3.6V**
- Analog supply voltage: **1.7-3.6V**

The following figure summarizes the I/O connection supported and internal peripherals.

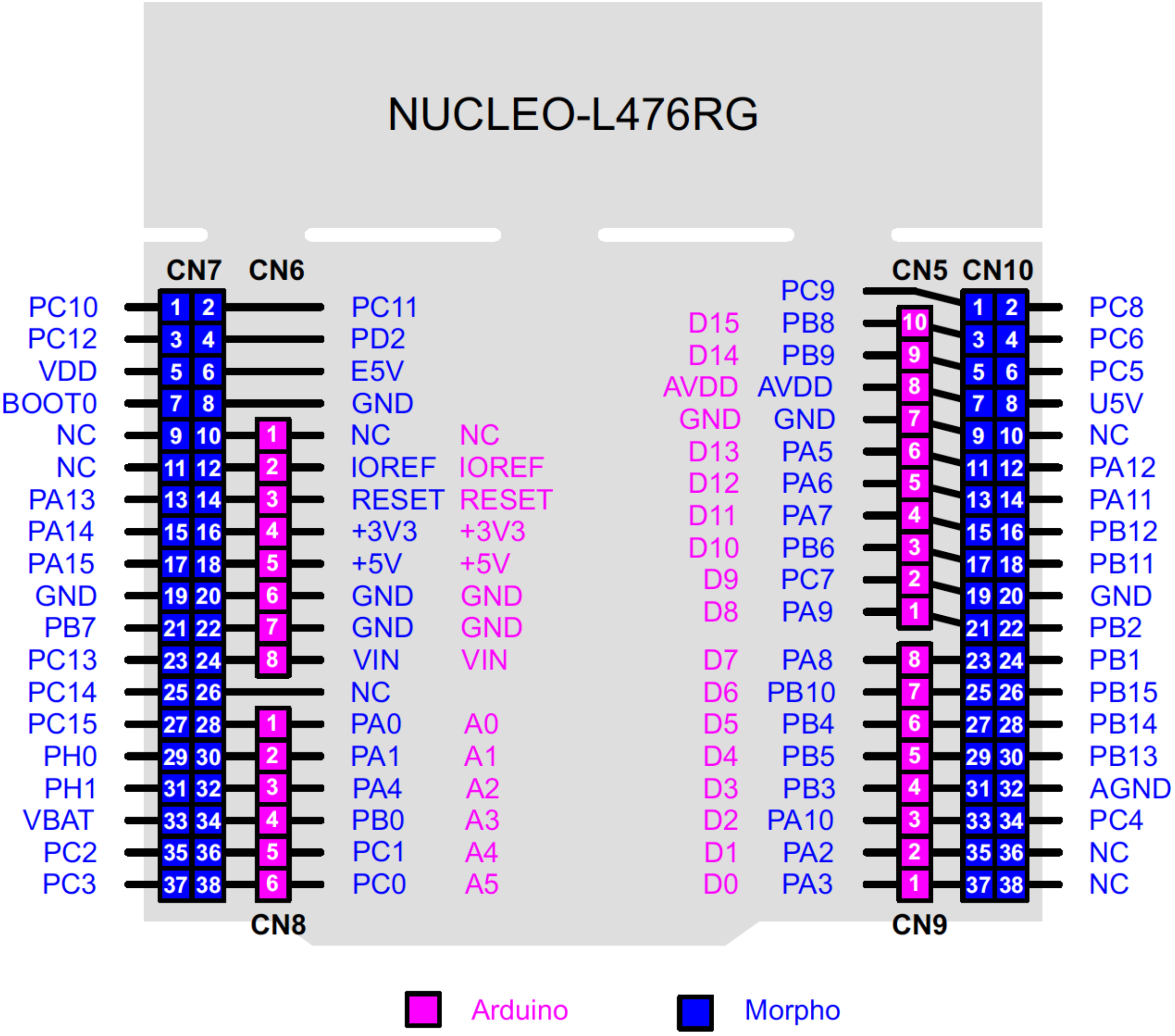
STM32F446


System
Power supply 1.2 V internal regulator POR/PDR/PVD
Xtal oscillators 32 kHz + 4 ~26 MHz
Internal RC oscillators 32 kHz + 16 MHz
PLL
Clock control
RTC/AWU
1x SysTick timer
2x watchdogs (independent and window)
50/63/81/114 I/Os
Cyclic Redundancy Check (CRC)
96-bit unique ID
Voltage scaling
Chrom-ART Accelerator™
ART Accelerator™
180 MHz Arm® Cortex®-M4 CPU
Floating Point Unit (FPU)
Nested Vector Interrupt Controller (NVIC)
JTAG/SW debug
Embedded Trace Macrocell (ETM)
Memory Protection Unit (MPU)
Mutli-AHB bus matrix
16-channel DMA
Connectivity
4x SPI (3x with I²S)
Camera interface
4x I²C
2x CAN 2.0B
1x USB 2.0 OTG FS/HS
1x USB 2.0 OTG FS
1x SDMMC
4x USART + 2 UART LIN, smartcard, IrDA, modem control
2x SAI (Serial Audio Interface)
HDMI CEC
SPDIF input x4
Control
2x 16-bit motor-control PWM synchronized AC timer
10x 16-bit timers 2x 32-bit timers
True random number generator (RNG)
Up to 512-Kbyte Flash memory
128-Kbyte SRAM
External memory interface W/SDRAM support
80-byte + 4-Kbyte backup data
512 OTP bytes
Dual Quad SPI
Analog
2-channel 2x 12-bit DAC
Up to 3x 12-bit ADC 2.4 MSPS Up to 24 channels 7.2 MSPS
Temperature sensor

Appendix C: Pin Connections on Nucleo-64 board

This diagram is common for both Nucleo-L476RG and Nucleo-L446RE

Board Component	Microcontroller Pin	Comment
Green LED	`PA 5`	SB42 closed and SB29 open by default
Blue user button	`PC 13`	Pulled up externally
Black reset button	`NRST`	Connect to ground to reset
ST-Link UART TX	`PA 2`	STLK_TX
ST-Link UART RX	`PA 3`	STLK_RX
ST-Link SWO/TDO	`PB 3`	Trace output pin/Test Data Out pin
ST-Link SWDIO/TMS	`PA 13`	Data I/O pin/Test Mode State pin
ST-Link SWDCLK/TCK	`PA 14`	Clock pin/Test Clock pin



Appendix D: Clock Configuration

There are two major types of clocks: **system clock** and **peripheral clock**. A video tutorial is given here: https://youtu.be/o6ZWD0PAoJk

- ***System Clock***: To meet the requirement of performance and energy-efficiency for different applications, the processor core can be driven by four different clock sources, including ***HSI*** (high-speed internal) oscillator clock, ***HSE*** (high-speed external) oscillator clock, **PLL** clock, and **LSI (**low-speed internal and LSE (low-speed external) oscillator clock. A faster clock provides better performance but usually consumes more power, which is not appropriate for battery-powered systems.

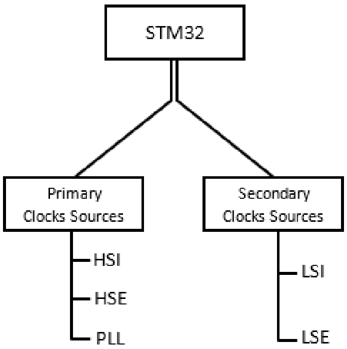


Figure-2: STM32F4xx Clock sources

- ***Peripheral Clock***: All peripherals require to be clocked to function. However, ***clocks of all peripherals are turned off by default to reduce power consumption***.

Overall the primary clock sources on STM32F4xx selection and distribution is shown in bellow figure.

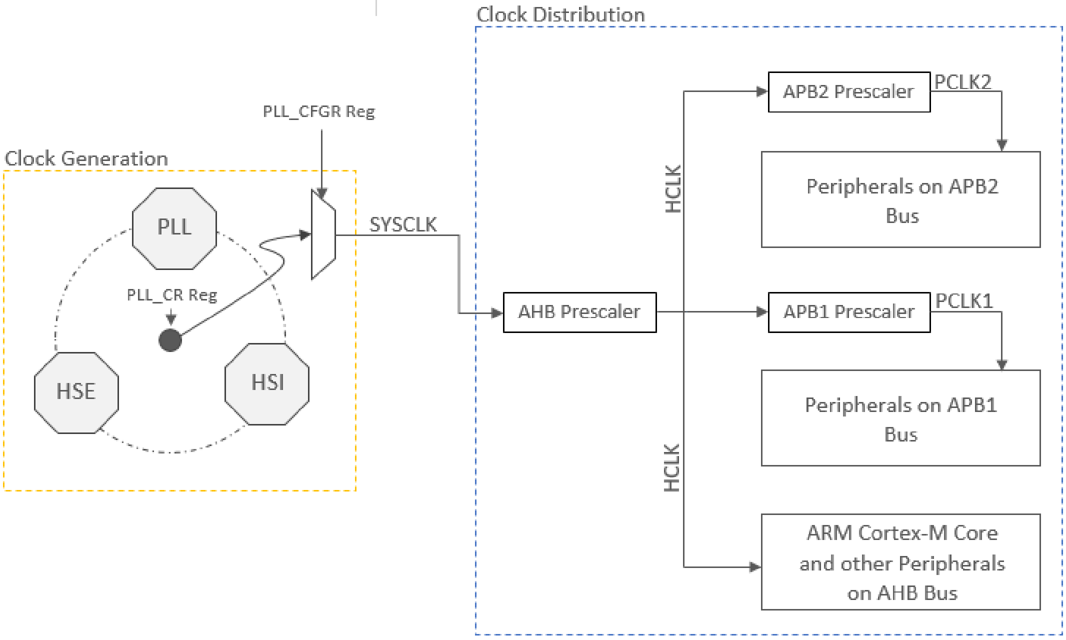
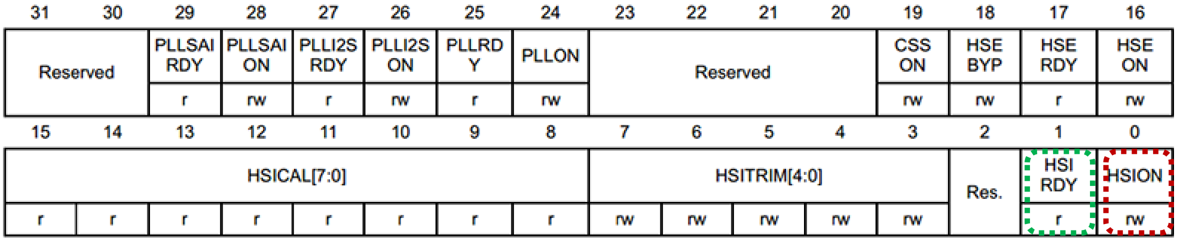


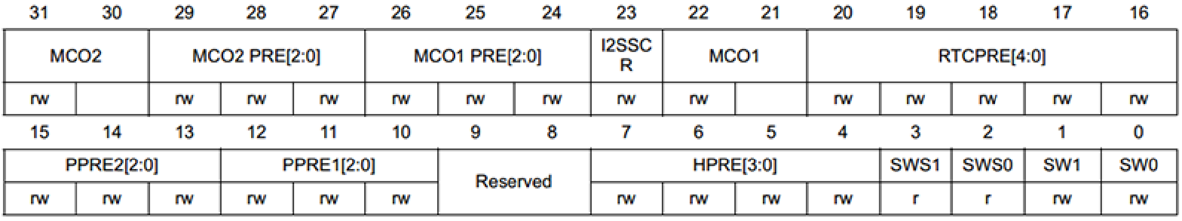
Figure-D1: STM32F4xx Primary clock selection and distribution

In STM32 any of three primary clock sources can be selected/deselected and switched via three RCC Registers i.e. *RCC clock control register (RCC_CR)* – Figure-4, *RCC clock configuration register (RCC_CFGR)*, and RCC PLL configuration register (RCC_PLLCFGR).



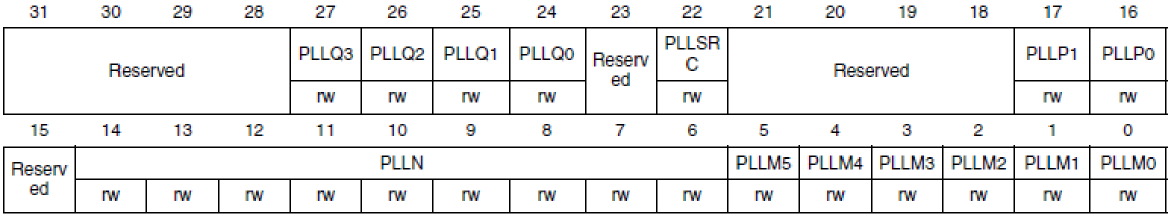
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLLSAI RDY	PLLSAI ON	PLLI2S RDY	PLLI2S ON	PLLRDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
		r	rw	r	rw	r	rw					rw	rw	r	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSICAL[7:0]				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

Figure-D2: STM32 RCC clock control register (RCC_CR)



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

Figure-D3: STM32 RCC clock configuration register (RCC_CFGR)

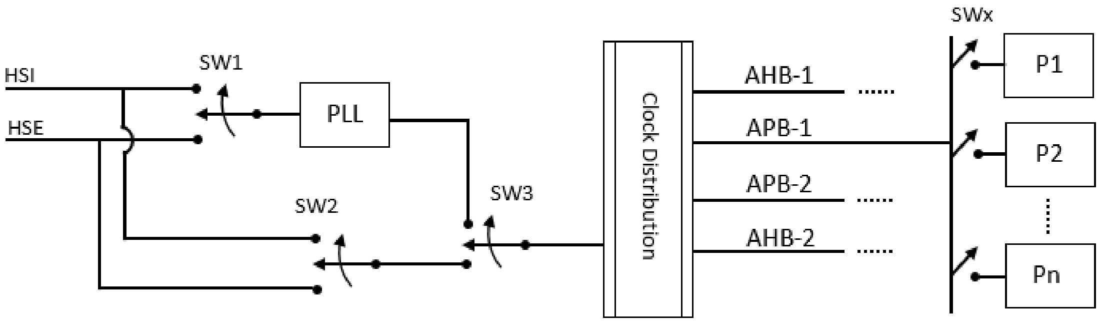


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure-D4: STM32 RCC PLL configuration register (RCC_PLLCFGR)

Following are the steps to select *High Speed Internal (HSI)* as System Clock.

1. Turn ON HSI Oscillator (bit-0, RCC-CR, **HSION**)
2. Wait until the clock get stable (bit-1, RCC-CR, **HSIRDY**)
3. Switch System clock to HSI (bit-0,1 in RCC_CFGR; see Figure-7 SW1,SW2)



- SWx = Switch-X
- Pn = Peripheral-n

Figure-D5: STM32F4xx RCC functional diagram

PLL Configuration

STM32F uses PLLs to take the input clock and adjust/multiply it and emit on an output pin. It acts as a input frequency amplifier.

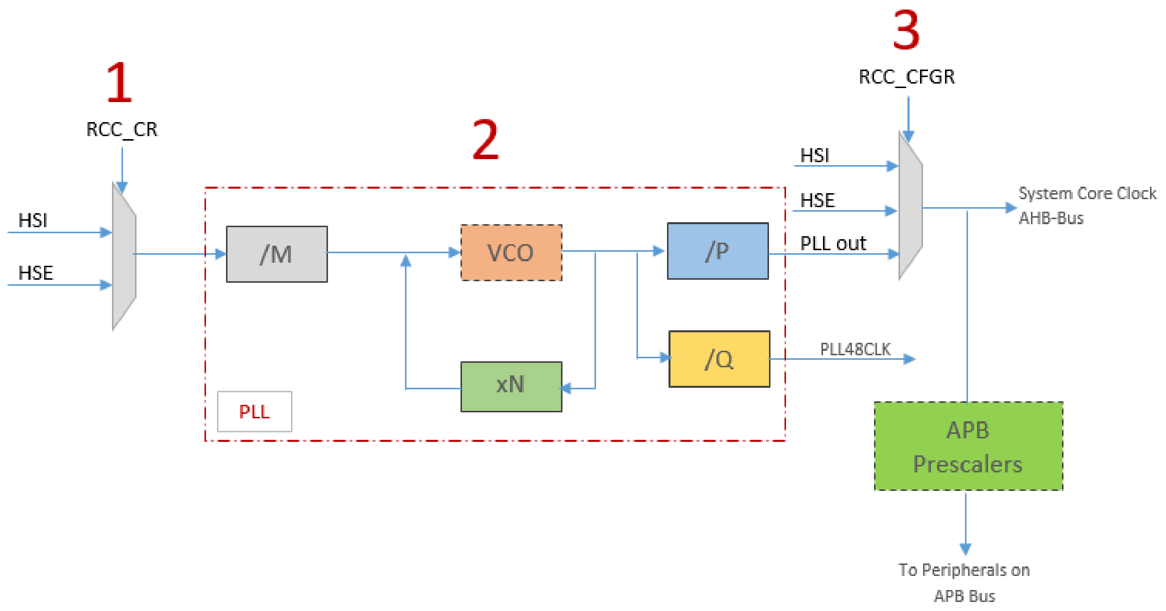


Figure-D6: STM32F4xx main PLL functional diagram

1. PLL input Clock Source: As mentioned earlier, PLL is not a clock source itself, it takes input clock from a clock source, adjust/multiply it, and emit it on output pin. It acts like an amplifier that amplifies input frequency. So before we can use PLL, we must feed it from some clock source – Figure-8.

2. Configuring PLL: The next step is to configure PLL main Logic as shown in Figure-8. The PLL output frequency is calculated as:

$$f_{(\text{VCO clock})} = \frac{f_{(\text{PLL clock input})} \times \text{PLLN}}{\text{PLLM}}$$

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

$$f_{(\text{USB OTG FS, SDIO, RNG Clock output})} = \frac{f_{(\text{VCO clock})}}{\text{PLLQ}}$$

While configuring PLL, following conditions need to meet otherwise PLL may not lock. Reserved bits must be kept at their default values in PLL registers.

PLL bits can be written only when PLL is disabled.

VCO input frequency ranges from 1 to 2 MHz. It is recommended to select a frequency of 2 MHz to limit PLL jitter.

VCO output frequency must be in range 192 to 432 MHz.

3. Switching to PLL: The last step is to switch system to PLL clock.

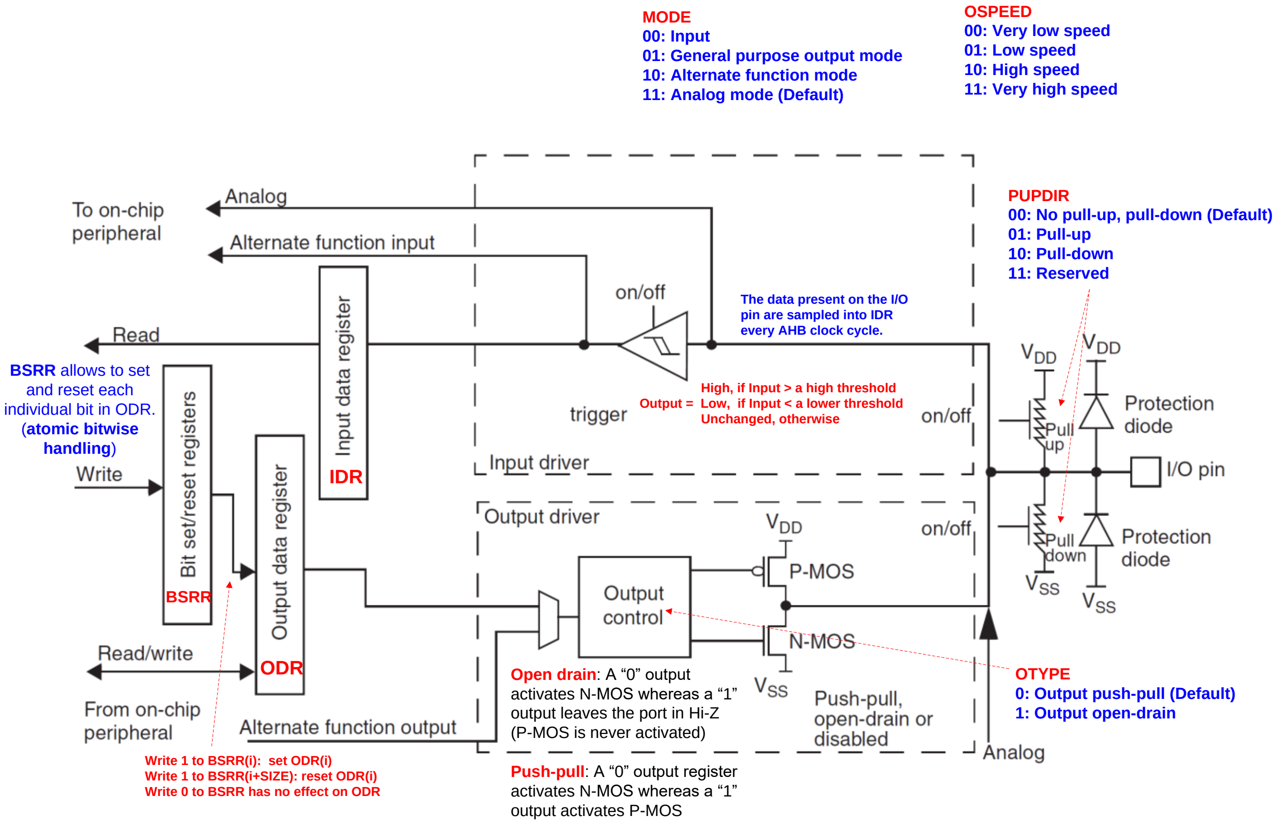
The software provided in this lab uses the 16MHz HSI as the input to the PLL clock. Appropriate scaling factors have been selected to achieve the maximum allowed clock speed (80 MHz). See the function void **System_Clock_Init**() for details.

```
static void enable_HSI(){

    /* Enable Power Control clock */
    /* RCC->APB1ENR |= RCC_APB1LPENR_PWRLPEN; */

    // Regulator voltage scaling output selection: Scale 2
    // PWR->CR |= PWR_CR_VOS_1;

    // Enable High Speed Internal Clock (HSI = 16 MHz)
    RCC->CR |= ((uint32_t)RCC_CR_HSION);
    while ((RCC->CR & RCC_CR_HSIRDY) == 0); // Wait until HSI ready

    // Store calibration value
    PWR->CR |= (uint32_t)(16 << 3);

    // Reset CFGR register
    RCC->CFGR = 0x00000000;

    // Reset HSEON, CSSON and PLLON bits
    RCC->CR &= ~(RCC_CR_HSEON | RCC_CR_CSSON | RCC_CR_PLLON);
    while ((RCC->CR & RCC_CR_PLLRDY) != 0); // Wait until PLL disabled

    // Programming PLLCFGR register
    // RCC->PLLCFGR = 0x24003010; // This is the default value

    // Tip:
    // Recommended to set VOC Input f(PLL clock input) / PLLM to 1-2MHz
    // Set VCO output between 192 and 432 MHz,
    // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
    // f(PLL general clock output) = f(VCO clock) / PLLP
    // f(USB OTG FS, SDIO, RNG clock output) = f(VCO clock) / PLLQ

    RCC->PLLCFGR = 0;
    RCC->PLLCFGR &= ~(RCC_PLLCFGR_PLLSRC);                  // PLLSRC = 0 (HSI 16 Mhz clock
selected as clock source)
    RCC->PLLCFGR |= 16 << RCC_PLLCFGR_PLLN_Pos;     // PLLM = 16, VCO input clock = 16 MHz /
PLLM = 1 MHz
    RCC->PLLCFGR |= 336 << RCC_PLLCFGR_PLLN_Pos;    // PLLN = 336, VCO output clock = 1 MHz *
336 = 336 MHz
    RCC->PLLCFGR |= 4 << RCC_PLLCFGR_PLLP_Pos;      // PLLP = 4, PLLCLK = 336 Mhz / PLLP = 84
MHz
    RCC->PLLCFGR |= 7 << RCC_PLLCFGR_PLLQ_Pos;      // PLLQ = 7, USB Clock = 336 MHz / PLLQ =
48 MHz

    // Enable Main PLL Clock
    RCC->CR |= RCC_CR_PLLON;
    while ((RCC->CR & RCC_CR_PLLRDY) == 0);  // Wait until PLL ready

    // FLASH configuration block
    // enable instruction cache, enable prefetch, set latency to 2WS (3 CPU cycles)
    FLASH->ACR |= FLASH_ACR_ICEN | FLASH_ACR_PRFTEN | FLASH_ACR_LATENCY_2WS;

    // Configure the HCLK, PCLK1 and PCLK2 clocks dividers
    // AHB clock division factor


    RCC->CFGR &= ~RCC_CFGR_HPRE; // 84 MHz, not divided
    // PPRE1: APB Low speed prescaler (APB1)
```

```
            RCC->CFGR &= ~RCC_CFGR_PPRE1;
            RCC->CFGR |= RCC_CFGR_PPRE1_DIV2; // 42 MHz, divided by 2
            // PPRE2: APB high-speed prescaler (APB2)
            RCC->CFGR &= ~RCC_CFGR_PPRE2; // 84 MHz, not divided

            // Select PLL as system clock source
            // 00: HSI oscillator selected as system clock
            // 01: HSE oscillator selected as system clock
            // 10: PLL selected as system clock
            RCC->CFGR &= ~RCC_CFGR_SW;
            RCC->CFGR |= RCC_CFGR_SW_1;
            // while ((RCC->CFGR & RCC_CFGR_SWS_PLL) != RCC_CFGR_SWS_PLL);

            // Configure the Vector Table location add offset address
//          VECT_TAB_OFFSET  = 0x00UL; // Vector Table base offset field.
                        // This value must be a multiple of 0x200.
            SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; // Vector Table Relocation in Internal FLASH

}
```

See this tutorial for a comprehensive guide on how to set different clock sources: https://ecoderlenz.com/?p=830

Appendix E: General Purpose I/O



MODE
00: Input
01: General purpose output mode
10: Alternate function mode
11: Analog mode (Default)

OSPEED
00: Very low speed
01: Low speed
10: High speed
11: Very high speed

PUPDIR
00: No pull-up, pull-down (Default)
01: Pull-up
10: Pull-down
11: Reserved

The data present on the I/O pin are sampled into IDR every AHB clock cycle.

Output = High, if Input > a high threshold
Low, if Input < a lower threshold
Unchanged, otherwise

BSRR allows to set and reset each individual bit in ODR. (**atomic bitwise handling**)

Write 1 to BSRR(i): set ODR(i)
Write 1 to BSRR(i+SIZE): reset ODR(i)
Write 0 to BSRR has no effect on ODR

Open drain: A “0” output activates N-MOS whereas a “1” output leaves the port in Hi-Z (P-MOS is never activated)

Push-pull: A “0” output register activates N-MOS whereas a “1” output activates P-MOS

OTYPE
0: Output push-pull (Default)
1: Output open-drain