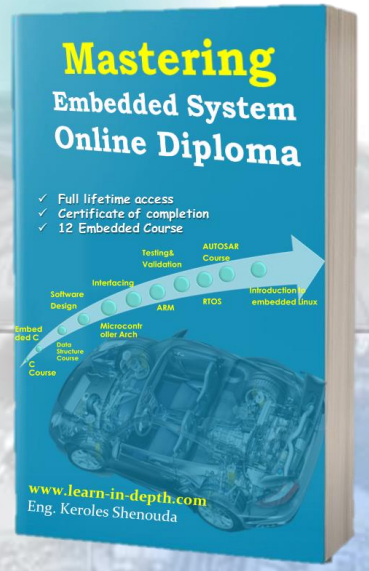
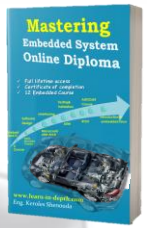
Mastering Embedded System

Online Diploma

- ✓ Full lifetime access
- ✓ Access on Android mobile and PC (Windows)
- ✓ Certificate of completion
- ✓ 12 Embedded Course



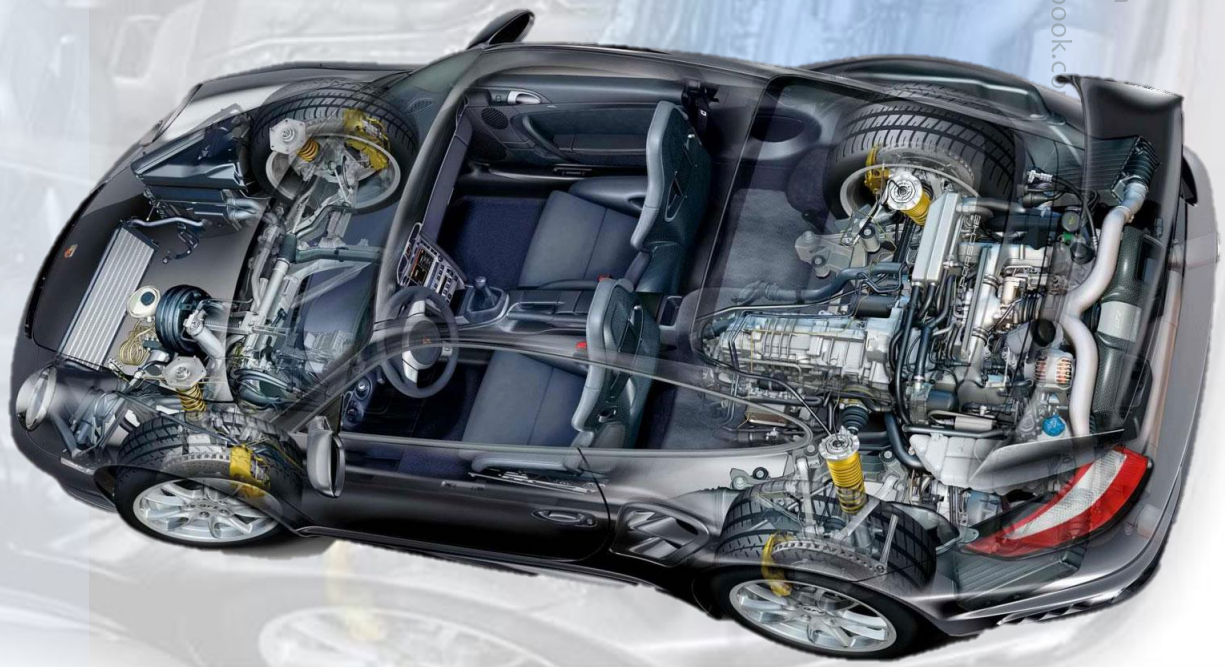


#LEARN_IN DEPTH

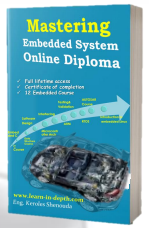
#Be_professional_in_embedded_system

Unit 6 (MCU Fundamentals) . lesson 3 MCU Clocks

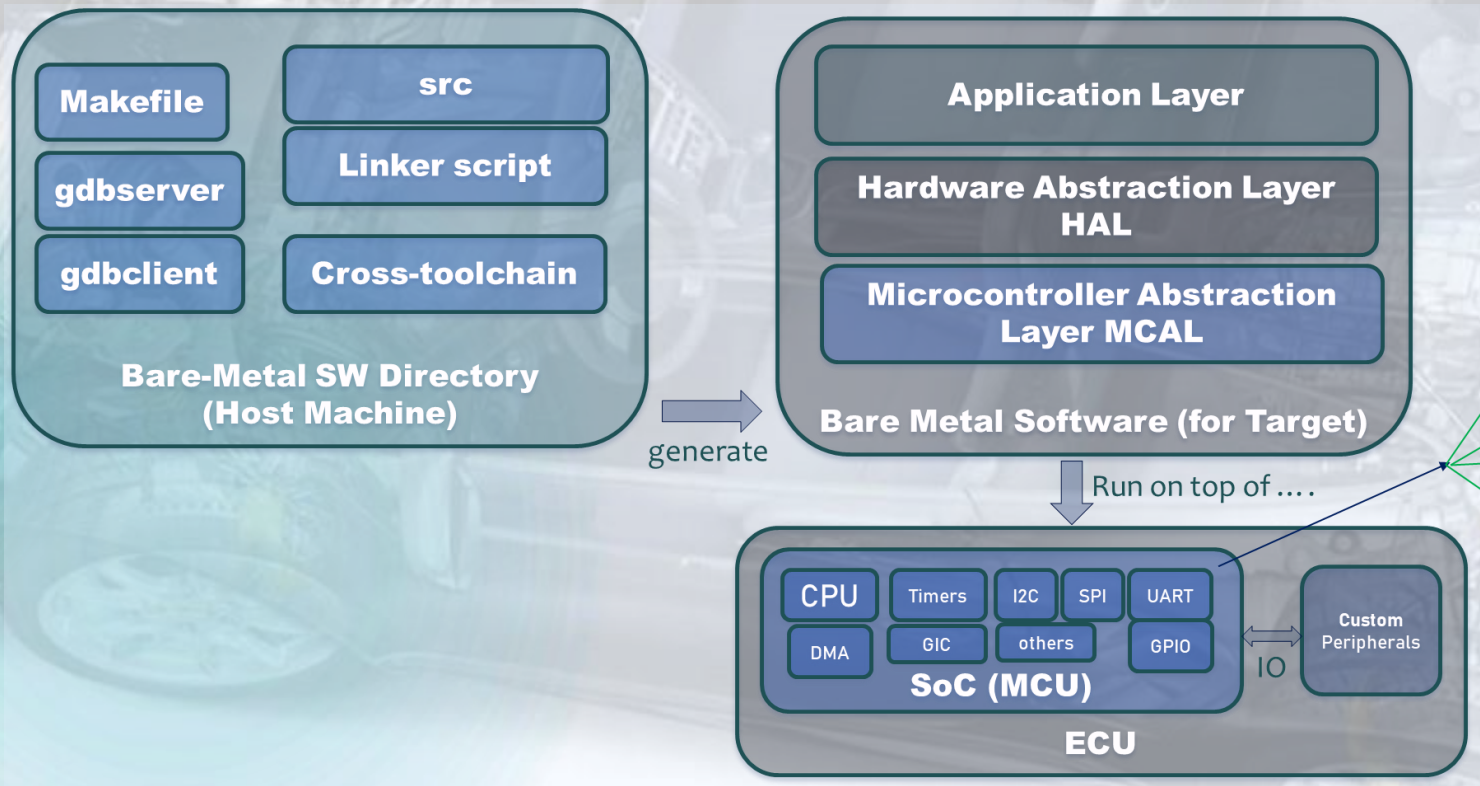
- **MCU Clocking**
- **General Topology of the Clock Architecture** system clock (SYSCLK)
- **General Topology of the Clock Architecture** system clock (SYSCLK)
- System Control and Clocks **in TM4C123 MCU**
- MCU Clock Sources
- Steps to find out the Clock tree in any MCU
- Understanding Clock Tree
- external crystal to SYSCLK examples
- High SPEED INTERNAL CLOCK SIGNAL (HIS)
- (Clock Design) failures example
- Peripherals Clocks
- LAB1: Practical lab on STM32F103XX
- Lab2: Change SYSCLK, HCLK, PCLK1 and PCLK2 wit different frequencies

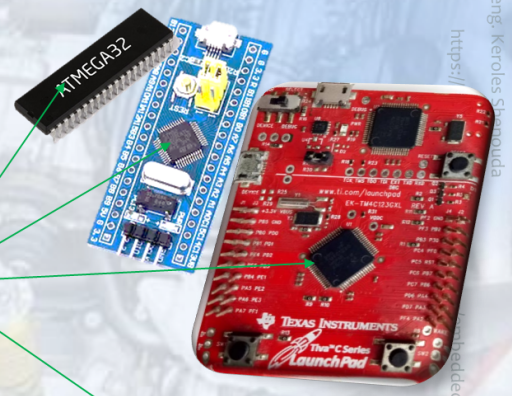


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Big Picture

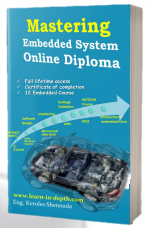




X – SoC

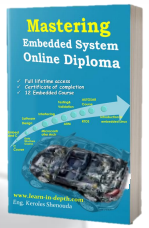
Read Specs and Program X - SoC

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



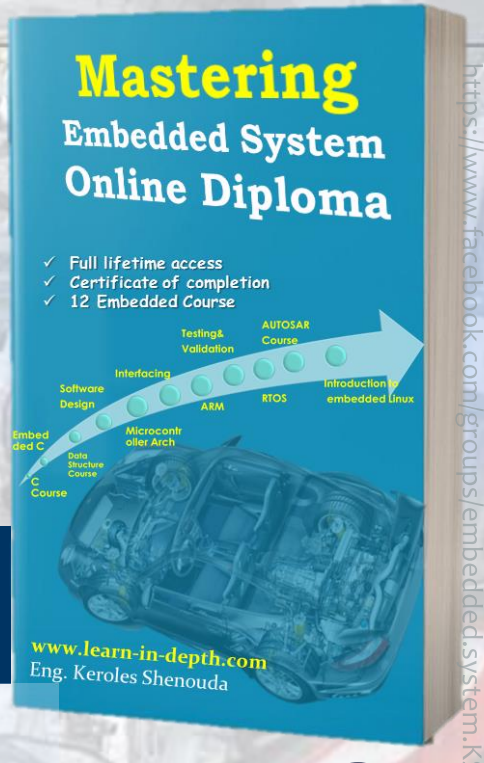
Unit 6 (Big Picture) Cont.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

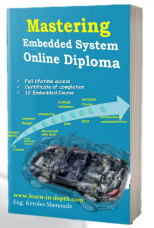
#Be_professional_in_embedded_system



MCU Clocking

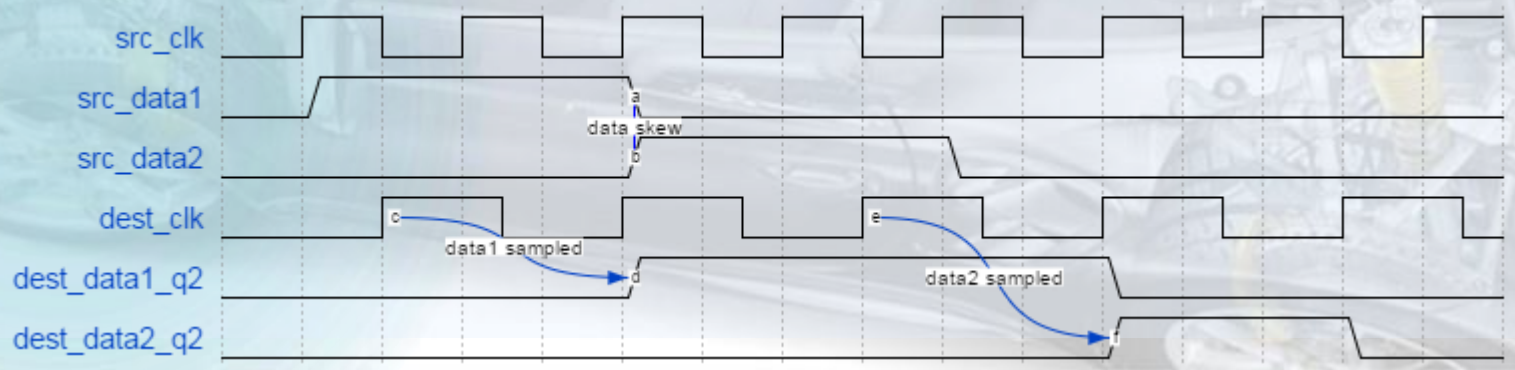
LEARN-IN-DEPTH
Be professional in embedded system

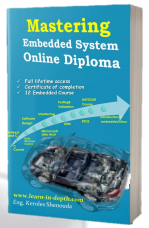
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



MCU Clocking

- Clocks used inside MCU in the **core** and by **peripherals**
- A clock in hardware is used to give the electronic circuits a **heartbeat** so that everything works in lock step.
- Clocks are used to **adjust the operating speed** of the part.
- **A peripheral with a slow clock** uses the **least amount of power**.
- **Higher frequencies** result **in more power consumption**.

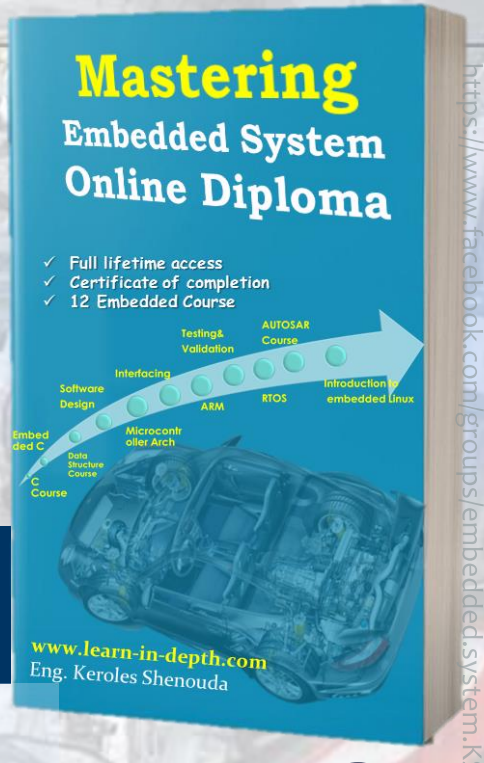




#LEARN_IN DEPTH

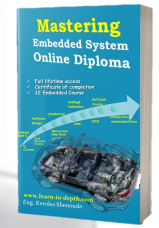
#Be_professional_in_embedded_system

General Topology of the Clock Architecture
system clock (SYSCLK)

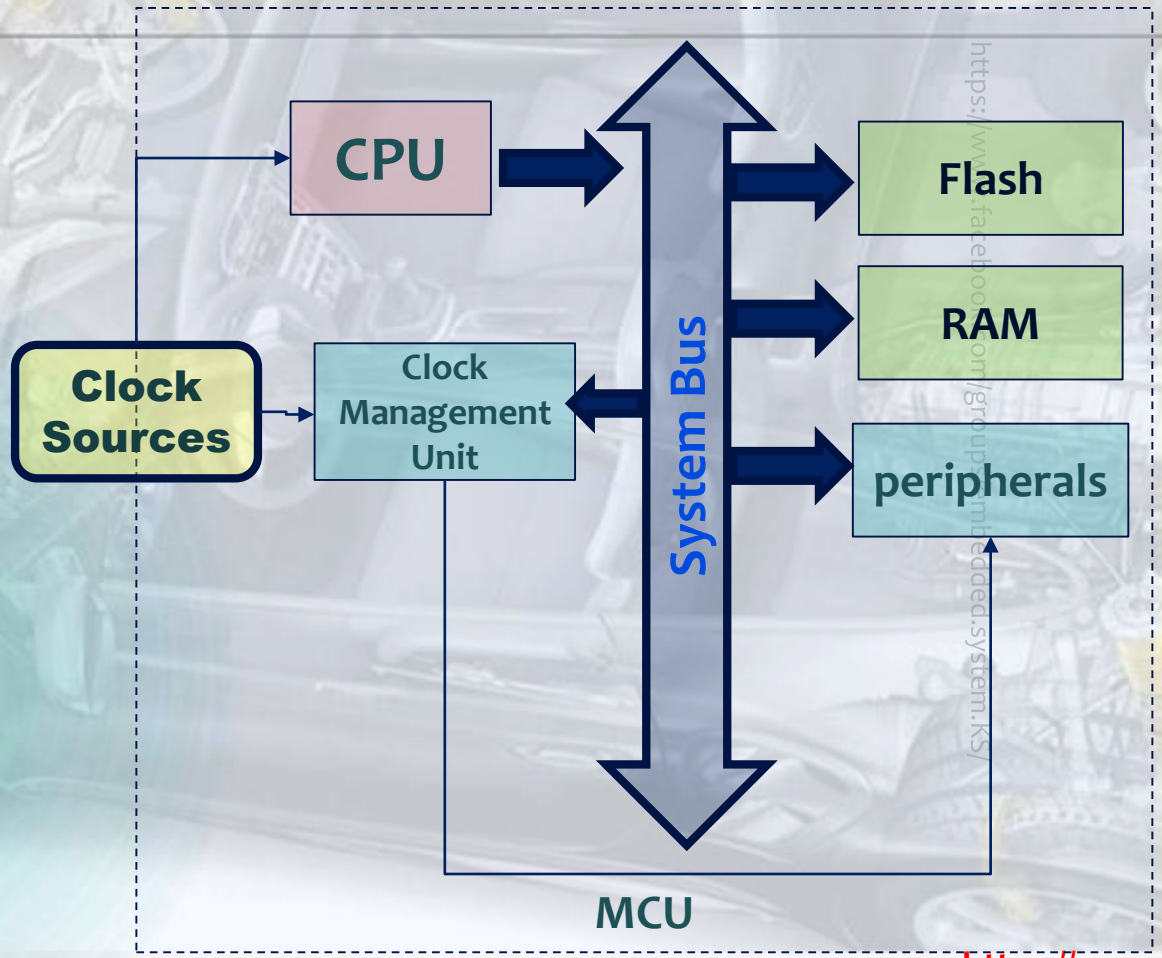


LEARN-IN-DEPTH
Be professional in embedded system

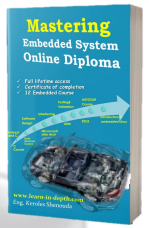
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



General Topology of the Clock Architecture



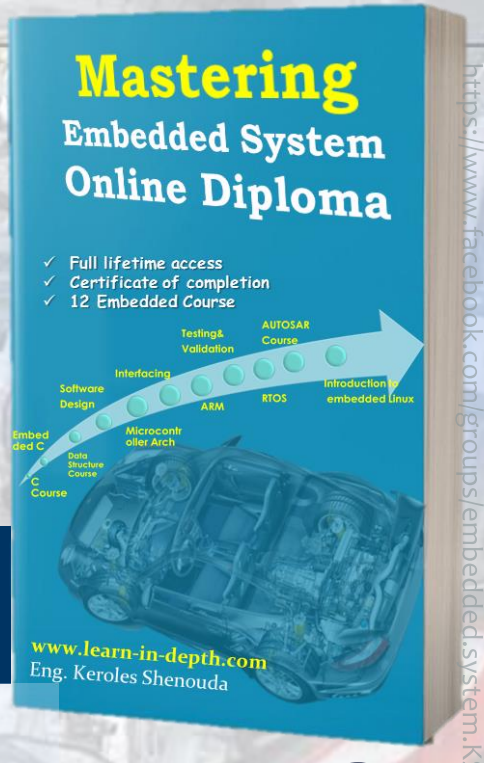
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

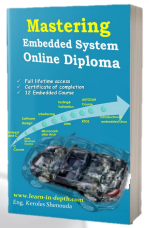
#Be_professional_in_embedded_system

Navigate Clock Management Unit on different SoCs
Tm4c123, Stm32F103X and …. From TRM



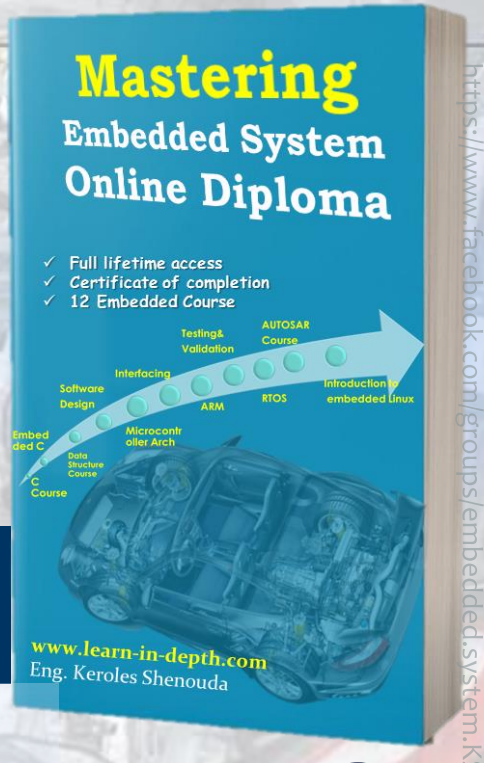
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

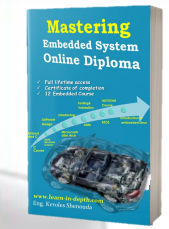
#Be_professional_in_embedded_system



Reset & clock control (RCC) in Stm32F103XX

LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



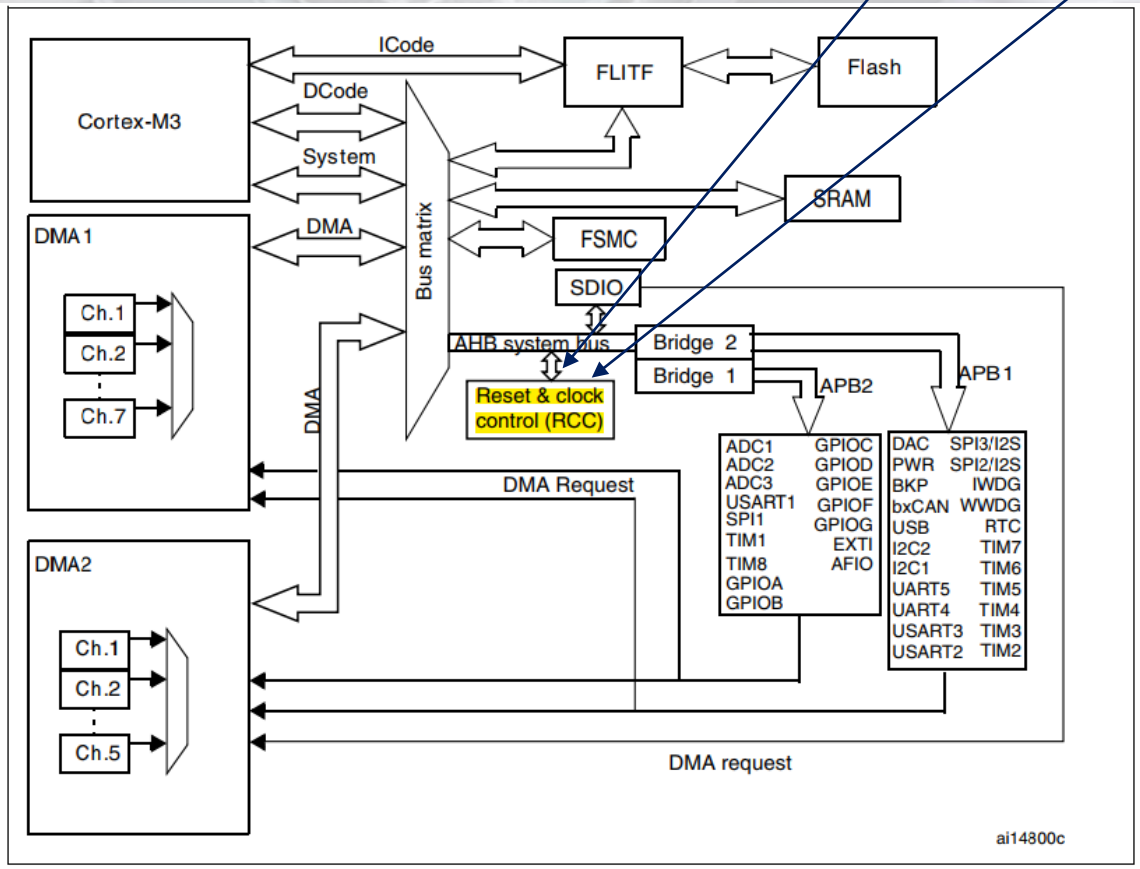
#LEARN_IN DEPTH

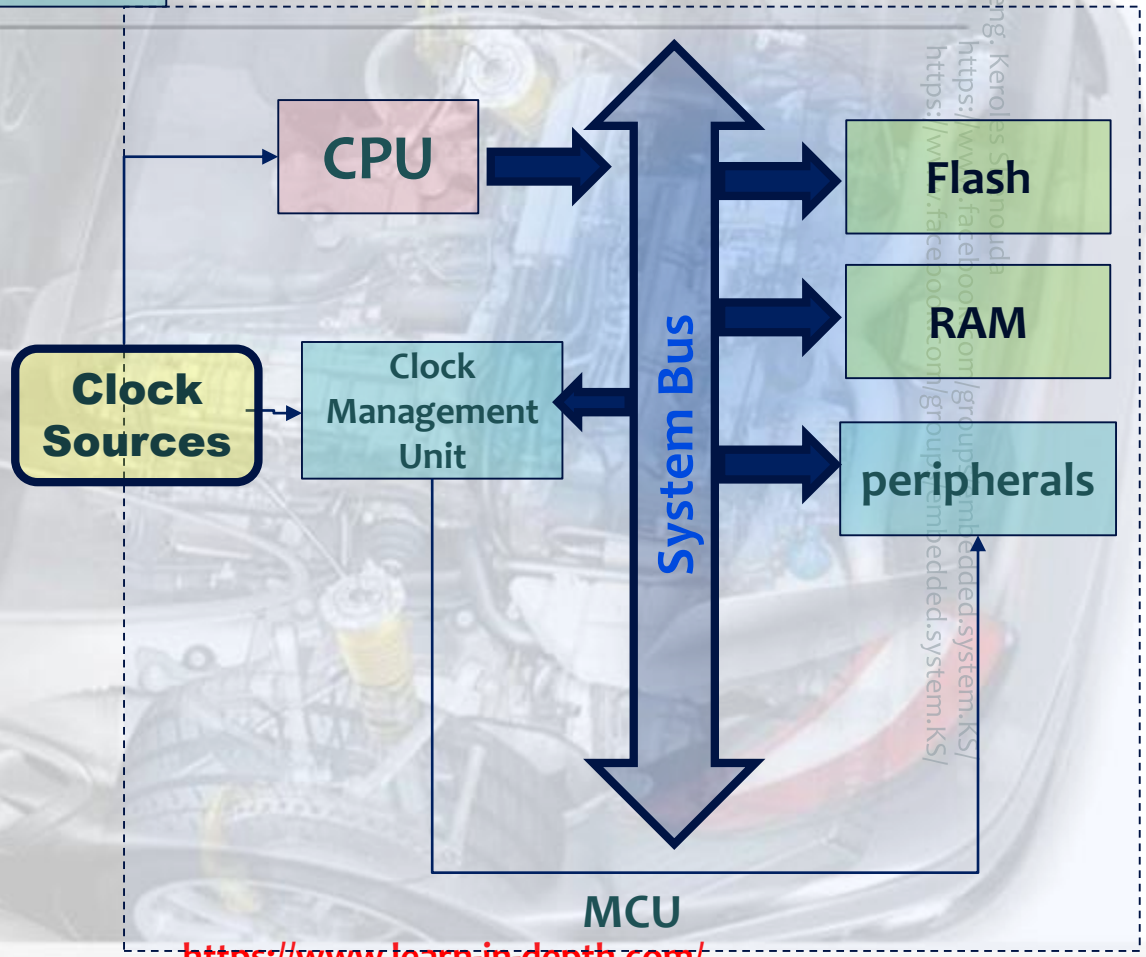
#Be_professional_in_embedded_system

STM32F103X

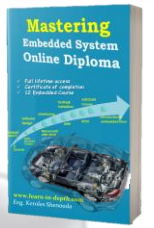
Connect with AHB Bus

Clock Management Unit





https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

STM32F103X

- 7 Low-, medium-, high- and XL-density reset and clock control (RCC)
  - 7.1 Reset
  - 7.2 Clocks
  - 7.3 RCC registers
- 8 Connectivity line devices: reset and clock control (RCC)
  - 8.1 Reset
  - 8.2 Clocks
  - 8.3 RCC registers

Low-, medium-, high- and XL-density reset and clock control (RCC)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

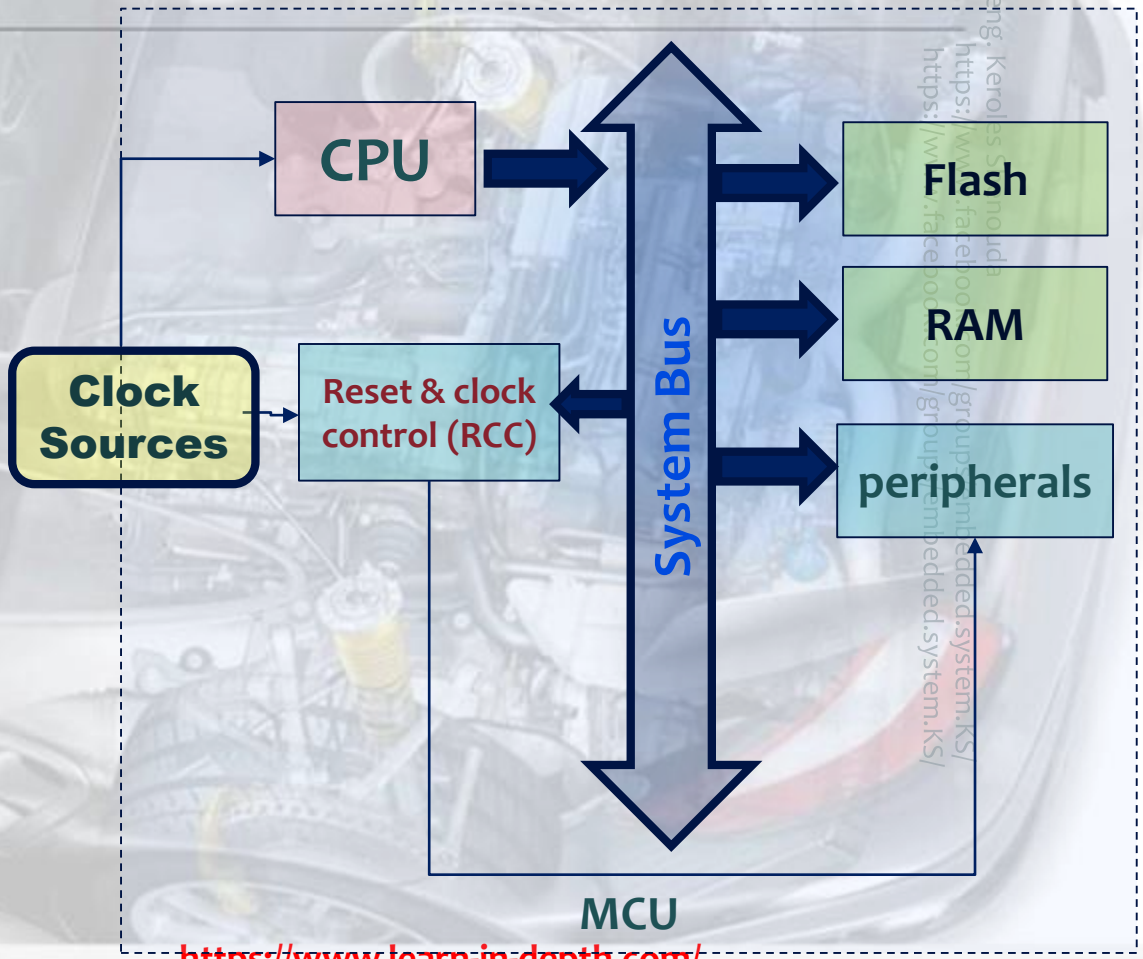
Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

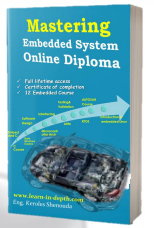
XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

This Section applies to low-, medium-, high- and XL-density STM32F10xxx devices. Connectivity line devices are discussed in a separate section (refer to *Connectivity line devices: reset and clock control (RCC)*).



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



STM32F103X

Clocks

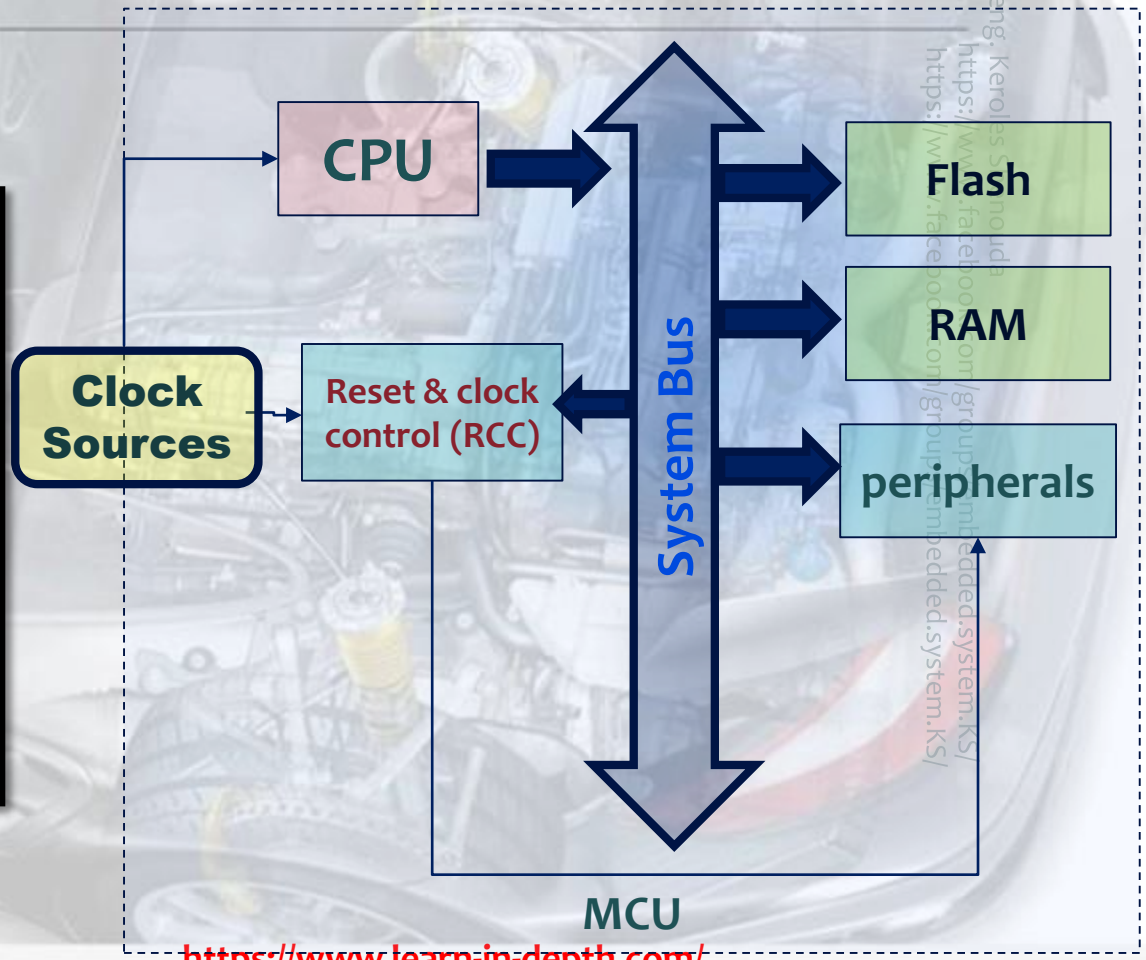
Three different clock sources can be used to drive the system clock (SYSCLK):

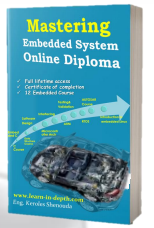
- HSI oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following two secondary clock sources:

- 40 kHz low speed internal RC (LSI RC), which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
- 32.768 kHz low speed external crystal (LSE crystal), which optionally drives the real-time clock (RTCCLK)

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





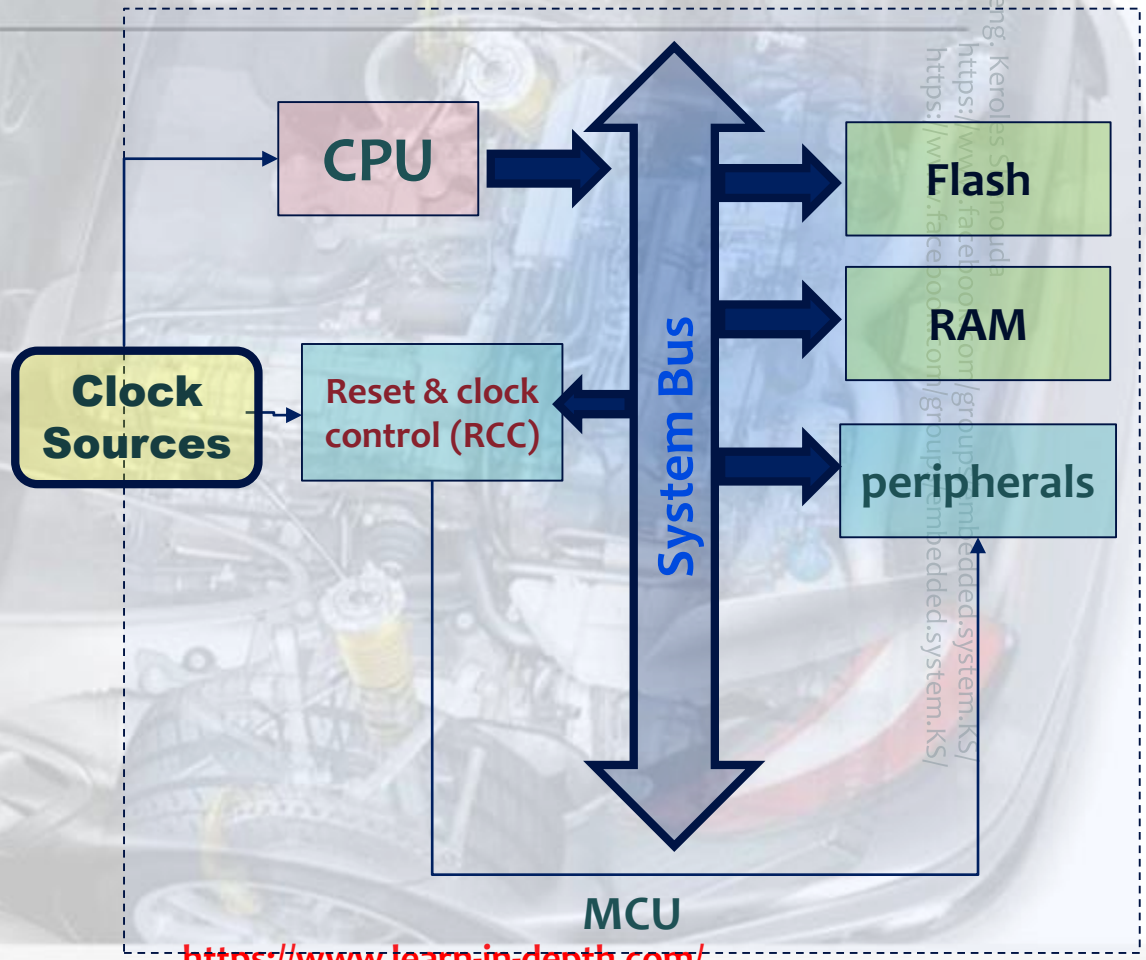
STM32F103X

3.3 Memory map

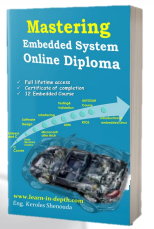
See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 3* gives the boundary addresses of the peripherals available in all STM32F10xxx devices.

Table 3. Register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC	AHB	*Section 21.6.9 on page 564*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*Section 28.16.6 on page 913*
0x4003 0000 - 0x4FFF FFFF	Reserved		-
0x4002 8000 - 0x4002 9FFF	Ethernet		*Section 29.8.5 on page 1069*
0x4002 3400 - 0x4002 7FFF	Reserved		-
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4 on page 65*
0x4002 2000 - 0x4002 23FF	Flash memory interface		-
0x4002 1400 - 0x4002 1FFF	Reserved		-
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		*Section 7.3.11 on page 121*
0x4002 0800 - 0x4002 0FFF	Reserved		-
0x4002 0400 - 0x4002 07FF	DMA2		*Section 13.4.7 on page 289*
0x4002 0000 - 0x4002 03FF	DMA1		
0x4001 8400 - 0x4001 FFFF	Reserved		-
0x4001 8000 - 0x4001 83FF	SDIO		*Section 22.9.16 on page 621*



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

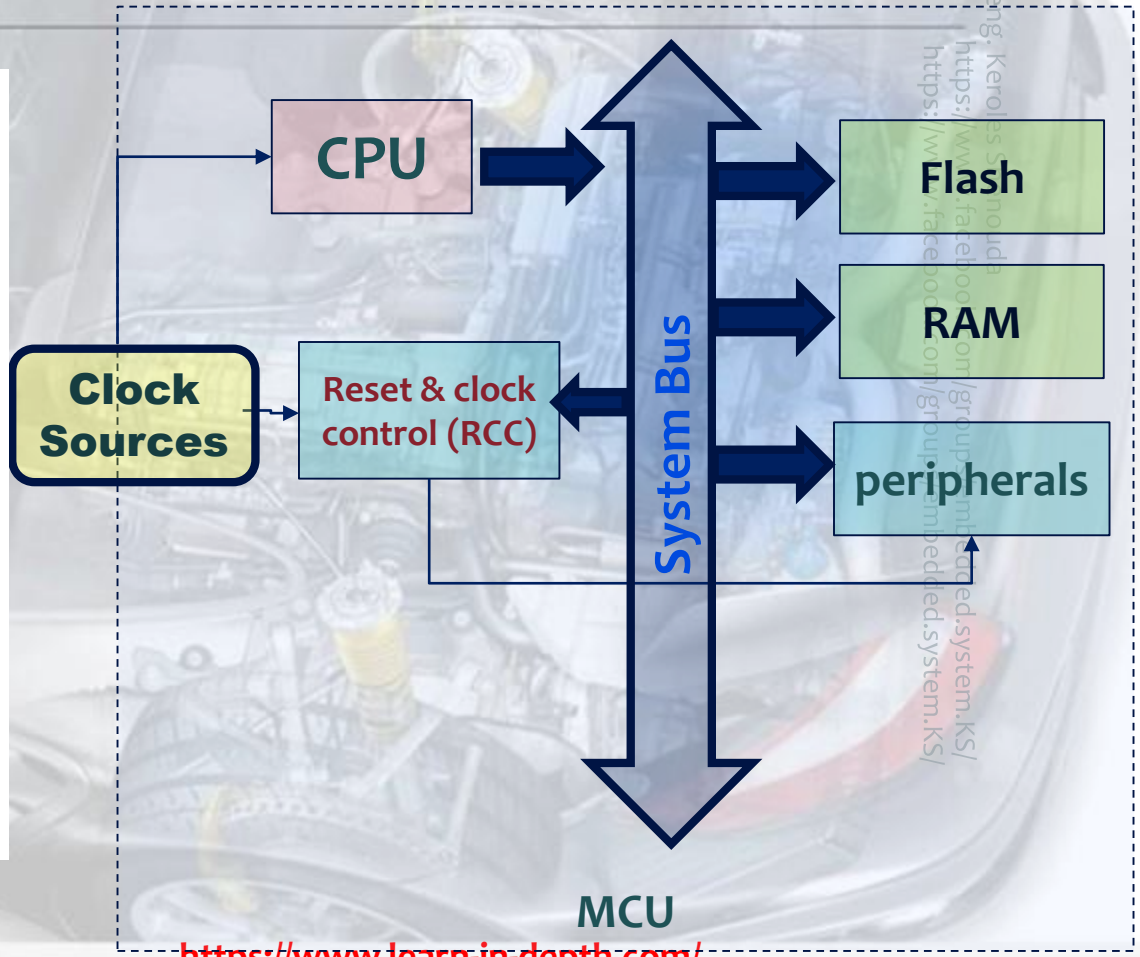


#LEARN_IN DEPTH

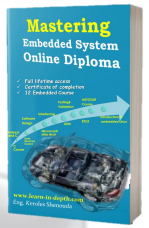
#Be_professional_in_embedded_system

STM32F103X

- 7 Low-, medium-, high- and XL-density reset and clock control (RCC)
  - 7.1 Reset
  - 7.2 Clocks
  - 7.3 RCC registers
    - 7.3.1 Clock control register (RCC_CR)
    - 7.3.2 Clock configuration register (RCC_CFGR)
    - 7.3.3 Clock interrupt register (RCC_CIR)
    - 7.3.4 APB2 peripheral reset register (RCC_APB2RSTR)
    - 7.3.5 APB1 peripheral reset register (RCC_APB1RSTR)
    - 7.3.6 AHB peripheral clock enable register (RCC_AHBENR)
    - 7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)
    - 7.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)
    - 7.3.9 Backup domain control register (RCC_BDCR)
    - 7.3.10 Control/status register (RCC_CSR)
    - 7.3.11 RCC register map



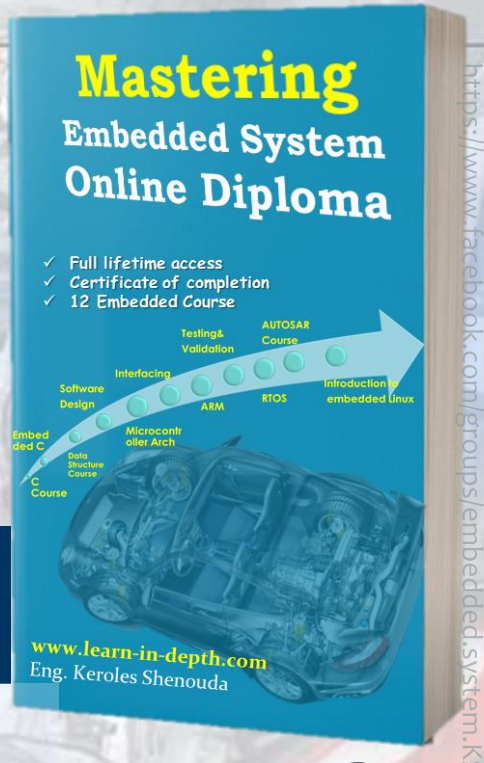
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

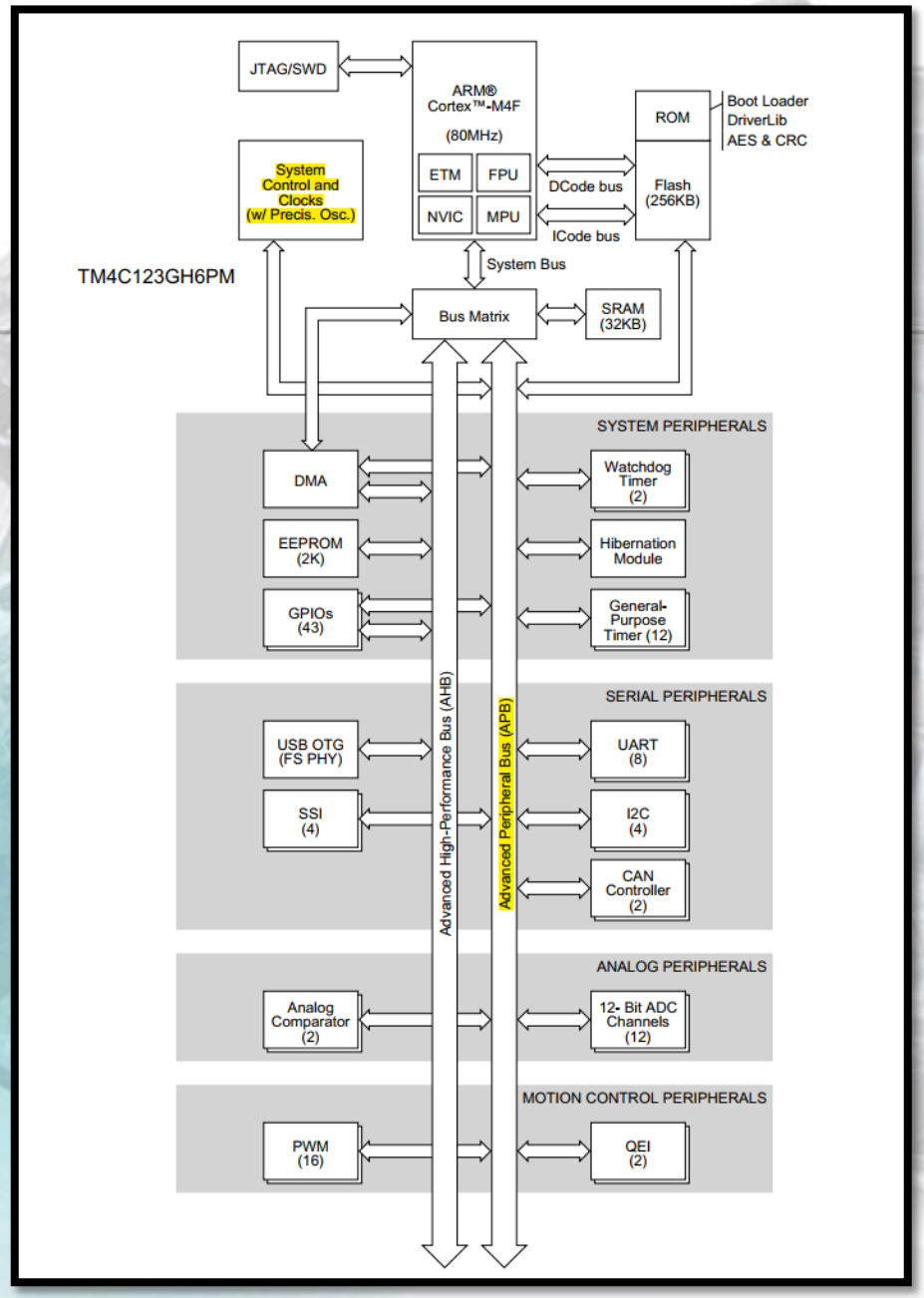
#Be_professional_in_embedded_system

System Control and Clocks in TM4C123 MCU

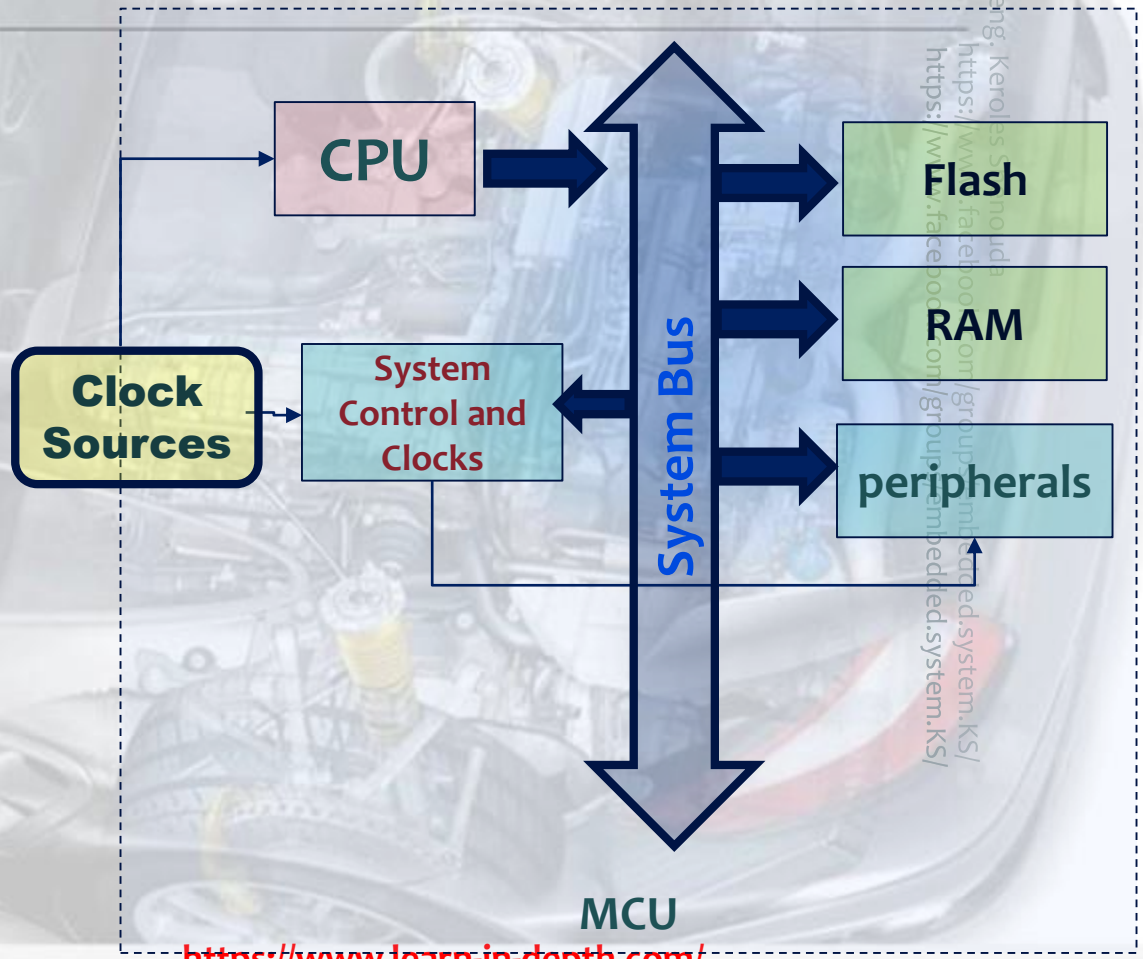


LEARN-IN-DEPTH
Be professional in embedded system

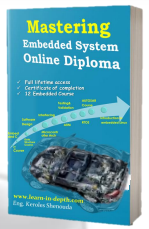
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



TM4c123



FOLLOW US
Press here
16



#LEARN_IN DEPTH

#Be_professional_in embedded_system

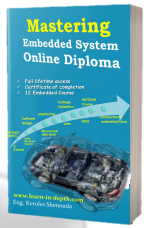
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

- Interrupt on transfer completion, with a separate interrupt per channel

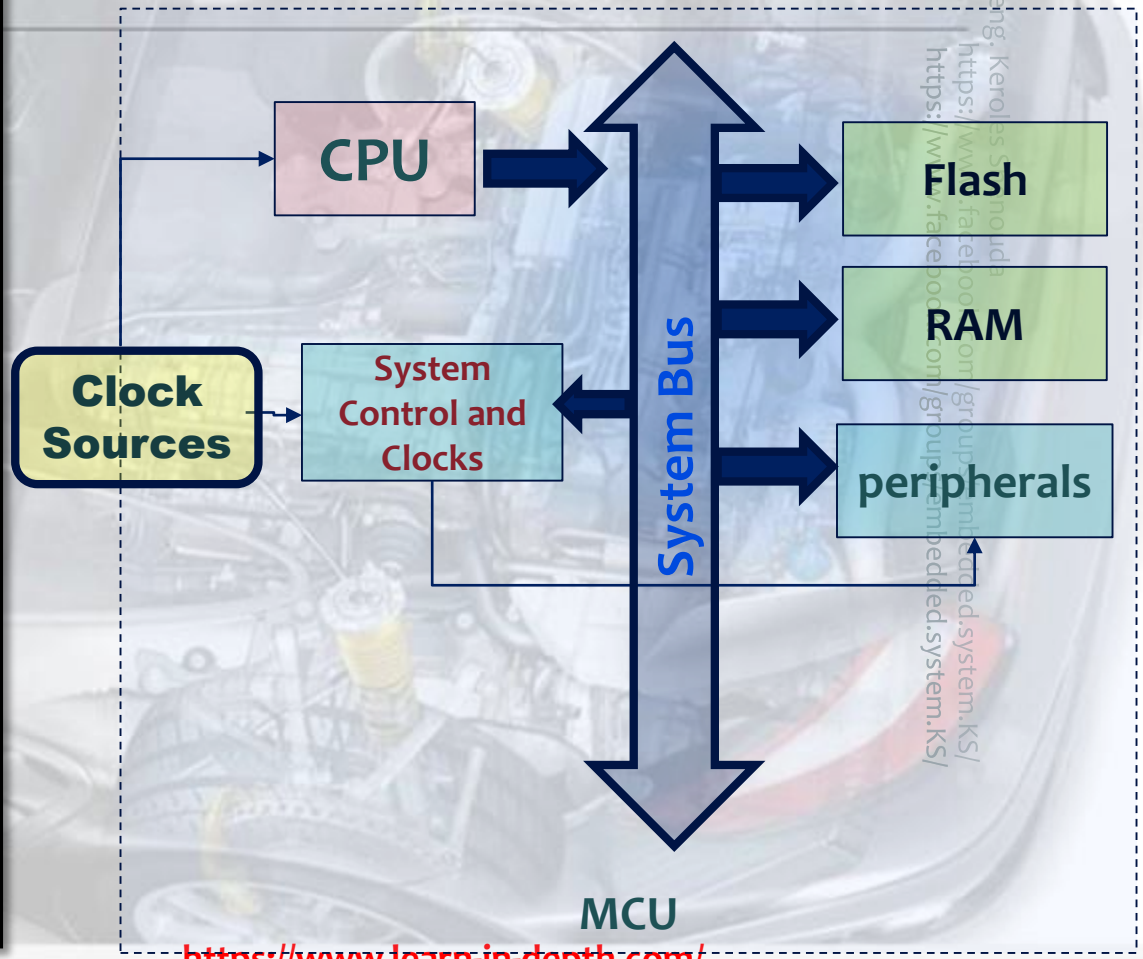
1.3.4.2 System Control and Clocks (see page 212)

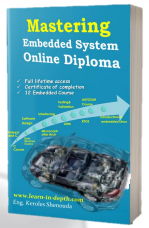
System control determines the overall operation of the device. It provides information about the device, controls power-saving features, controls the clocking of the device and individual peripherals, and handles reset detection and reporting.

- Device identification information: version, part number, SRAM size, Flash memory size, and so on
- Power control
  - On-chip fixed Low Drop-Out (LDO) voltage regulator
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options for microcontroller: Sleep and Deep-Sleep modes with clock gating
  - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock. The following clock sources are provided to the TM4C123GH6PM microcontroller:
  - Precision Internal Oscillator (PIOSC) providing a 16-MHz frequency
    - 16 MHz ±3% across temperature and voltage
    - Can be recalibrated with 7-bit trim resolution to achieve better accuracy (16 MHz ±1%)
    - Software power down control for low power modes
  - Main Oscillator (MOSC): A frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins.
  - Low Frequency Internal Oscillator (LFIOSC): On-chip resource used during power-saving modes
  - Hibernate RTC oscillator (RTCOSC) clock that can be configured to be the 32.768-kHz external oscillator source from the Hibernation (HIB) module or the HIB Low Frequency clock source (HIB LFIOSC), which is located within the Hibernation Module.
- Flexible reset sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out reset (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset



TM4c123



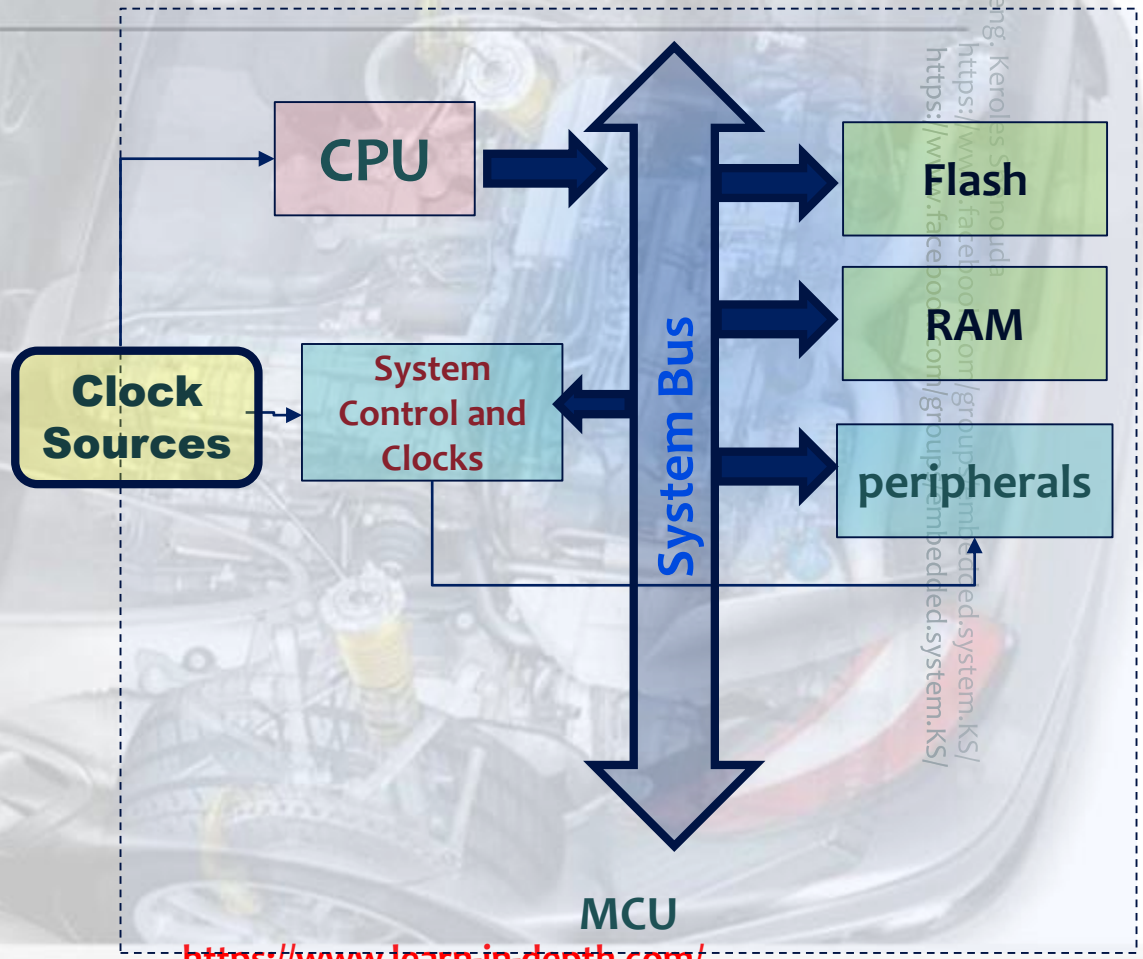


#LEARN_IN DEPTH

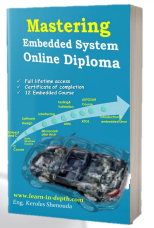
#Be_professional_in_embedded_system

TM4c123

- Tiva™ TM4C123GH6PM Microcontroller
- Table of Contents
- Revision History
- About This Document
- 1. Architectural Overview
- 2. The Cortex-M4F Processor
- 3. Cortex-M4 Peripherals
- 4. JTAG Interface
- 5. System Control
  - 5.1. Signal Description
  - 5.2. Functional Description
    - 5.2.1. Device Identification
    - 5.2.2. Reset Control
    - 5.2.3. Non-Maskable Interrupt
    - 5.2.4. Power Control
    - 5.2.5. Clock Control
    - 5.2.6. System Control
  - 5.3. Initialization and Configuration
  - 5.4. Register Map
  - 5.5. System Control Register Descriptions
  - 5.6. System Control Legacy Register Descriptions
- 6. System Exception Module
- 7. Hibernation Module
- 8. Internal Memory



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

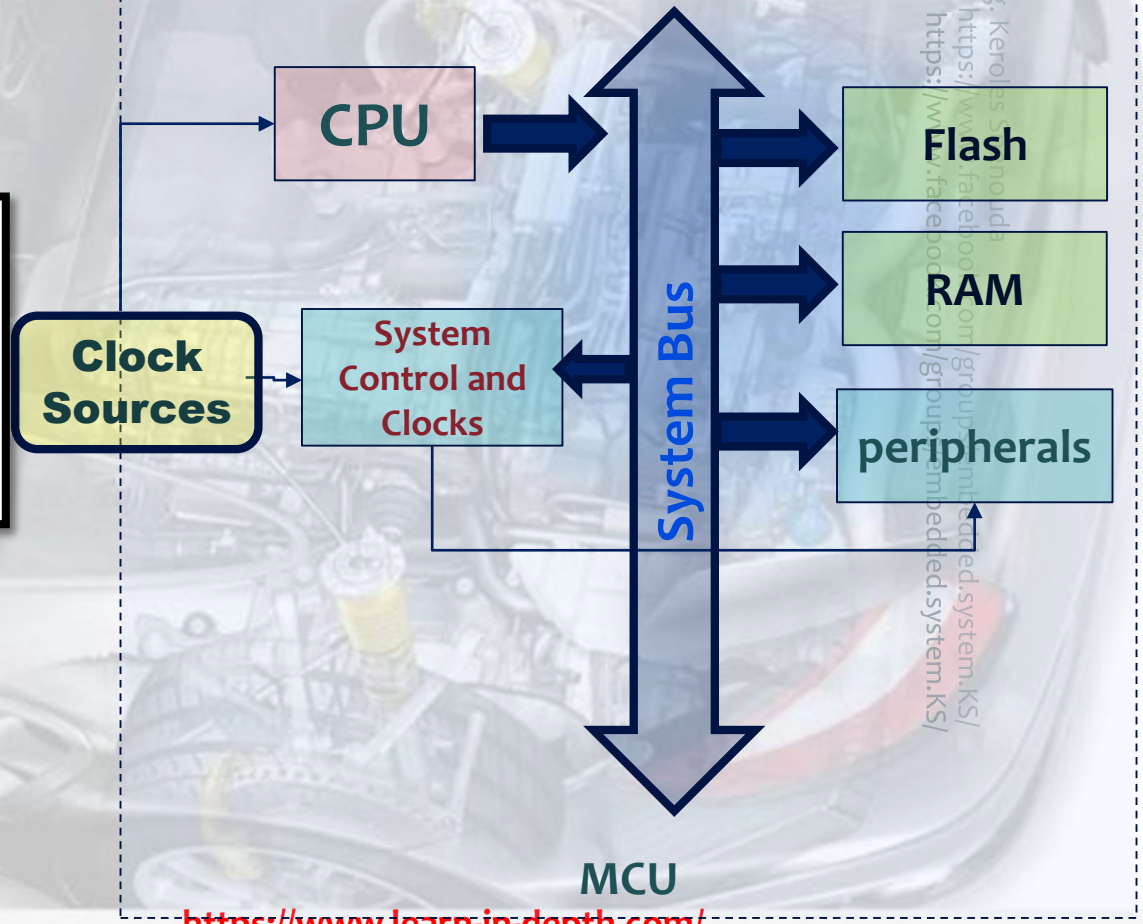


TM4c123

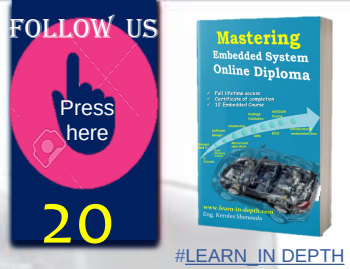
5.4 Register Map

Table 5-7 on page 232 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

Note: Spaces in the System Control register space that are not used are reserved for future or internal use. Software should not modify any reserved memory address.



TM4c123

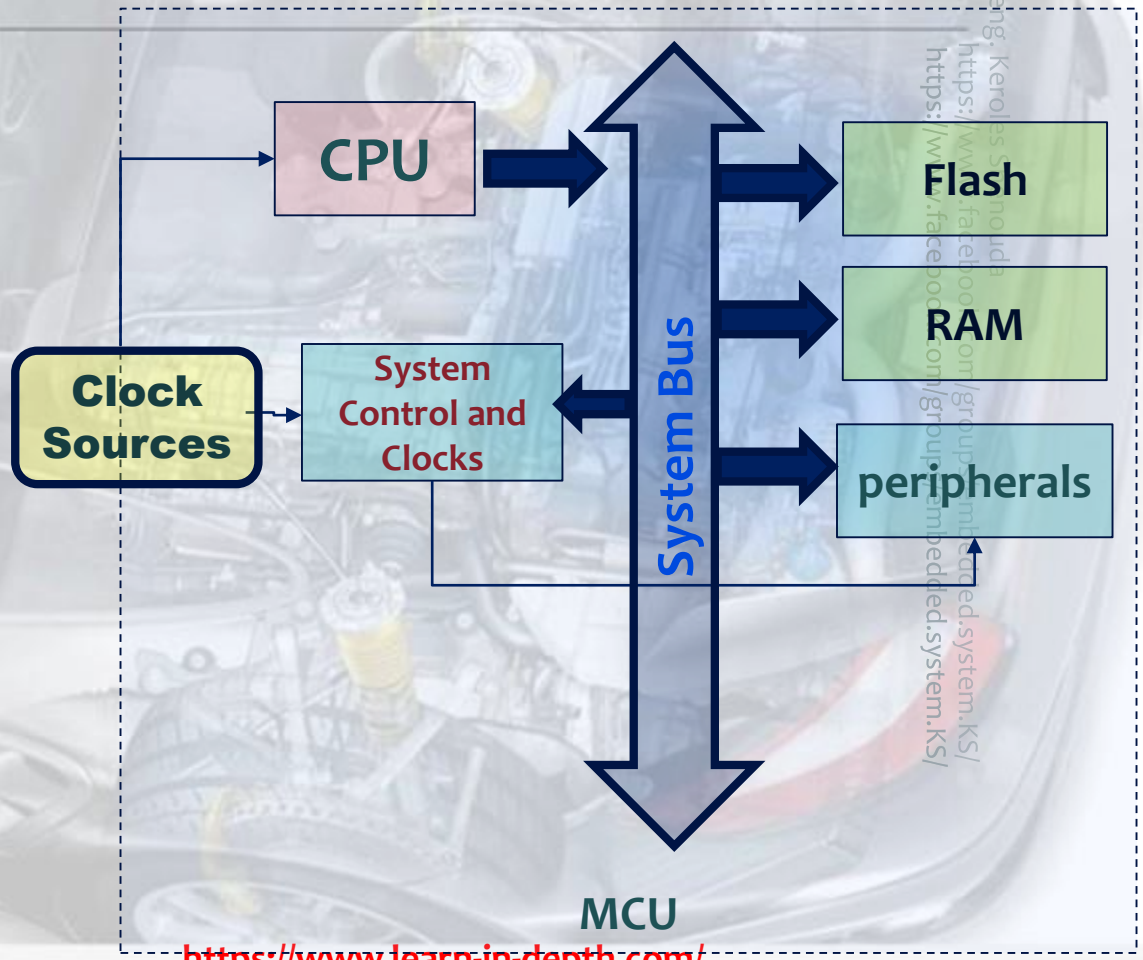


#LEARN_IN DEPTH

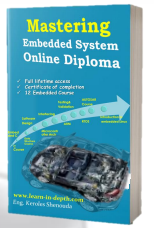
#Be_professional_in_embedded_system

Table 5-7. System Control Register Map

Offset	Name	Type	Reset	Description	See page
System Control Registers					
0x000	DID0	RO	-	Device Identification 0	238
0x004	DID1	RO	0x10A1.606E	Device Identification 1	240
0x030	PBORCTL	RW	0x0000.7FFF	Brown-Out Reset Control	243
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	244
0x054	IMC	RW	0x0000.0000	Interrupt Mask Control	247
0x058	MISC	RW1C	0x0000.0000	Masked Interrupt Status and Clear	249
0x05C	RESC	RW	-	Reset Cause	252
0x060	RCC	RW	0x078E.3AD1	Run-Mode Clock Configuration	254
0x06C	GPIOHBCTL	RW	0x0000.7E00	GPIO High-Performance Bus Control	258
0x070	RCC2	RW	0x07C0.6810	Run-Mode Clock Configuration 2	260
0x07C	MOSCCTL	RW	0x0000.0000	Main Oscillator Control	263
0x144	DSLPCLKCFG	RW	0x0780.0000	Deep Sleep Clock Configuration	264
0x14C	SYSPROP	RO	0x0000.1D31	System Properties	266
0x150	PIOSCCAL	RW	0x0000.0000	Precision Internal Oscillator Calibration	268
0x154	PIOSCSTAT	RO	0x0000.0040	Precision Internal Oscillator Statistics	270
0x160	PLLFREQ0	RO	0x0000.0032	PLL Frequency 0	271
0x164	PLLFREQ1	RO	0x0000.0001	PLL Frequency 1	272
0x168	PLLSTAT	RO	0x0000.0000	PLL Status	273
0x188	SLPPWRCFG	RW	0x0000.0000	Sleep Power Configuration	274
0x18C	DSLPPWRCFG	RW	0x0000.0000	Deep-Sleep Power Configuration	276
0x1B4	LDOSPCTL	RW	0x0000.0018	LDO Sleep Power Control	278
0x1B8	LDOSPCAL	RO	0x0000.1818	LDO Sleep Power Calibration	280
0x1BC	LDODPCTL	RW	0x0000.0012	LDO Deep-Sleep Power Control	281
0x1C0	LDODPCAL	RO	0x0000.1212	LDO Deep-Sleep Power Calibration	283
0x1CC	SDPMST	RO	0x0000.0000	Sleep / Deep-Sleep Power Mode Status	284
0x300	PPWD	RO	0x0000.0003	Watchdog Timer Peripheral Present	287
0x304	PPTIMER	RO	0x0000.003F	16/32-Bit General-Purpose Timer Peripheral Present	288
0x308	PPGPIO	RO	0x0000.003F	General-Purpose Input/Output Peripheral Present	290
0x30C	PPDMA	RO	0x0000.0001	Micro Direct Memory Access Peripheral Present	293



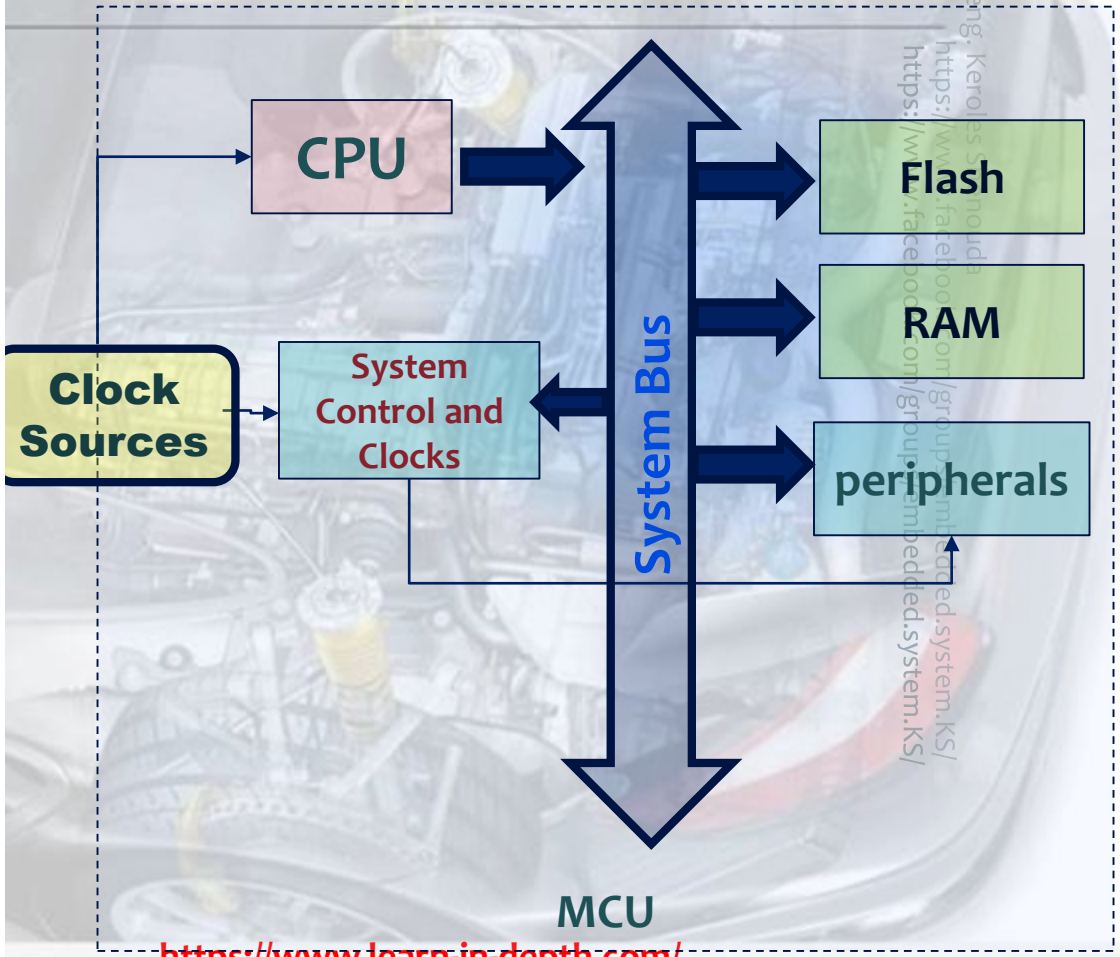
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



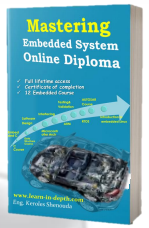
TM4c123

Table 2-4. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4003.D000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	1067
0x4004.1000	0x4004.1FFF	CAN1 Controller	1067
0x4004.2000	0x4004.BFFF	Reserved	-
0x4004.C000	0x4004.CFFF	32/64-bit Timer 2	725
0x4004.D000	0x4004.DFFF	32/64-bit Timer 3	725
0x4004.E000	0x4004.EFFF	32/64-bit Timer 4	725
0x4004.F000	0x4004.FFFF	32/64-bit Timer 5	725
0x4005.0000	0x4005.0FFF	USB	1114
0x4005.1000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	658
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	658
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	658
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	658
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	658
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	658
0x4005.E000	0x400A.EFFF	Reserved	-
0x400A.F000	0x400A.FFFF	EEPROM and Key Locker	540
0x400B.0000	0x400F.8FFF	Reserved	-
0x400F.9000	0x400F.9FFF	System Exception Module	485
0x400F.A000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	505
0x400F.D000	0x400F.DFFF	Flash memory control	540
0x400F.E000	0x400F.EFFF	System control	231
0x400F.F000	0x400F.FFFF	µDMA	606
0x4010.0000	0x41FF.FFFF	Reserved	-



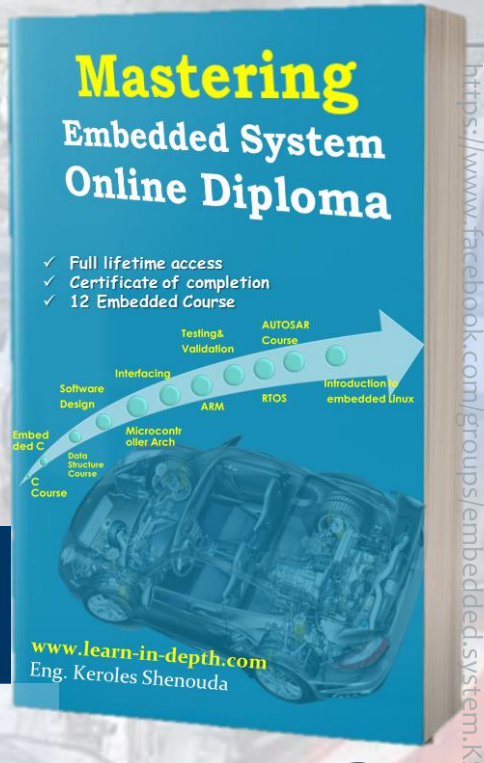
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

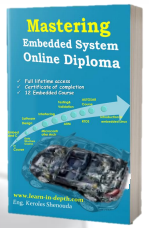
#Be_professional_in_embedded_system

MCU Clock Sources



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

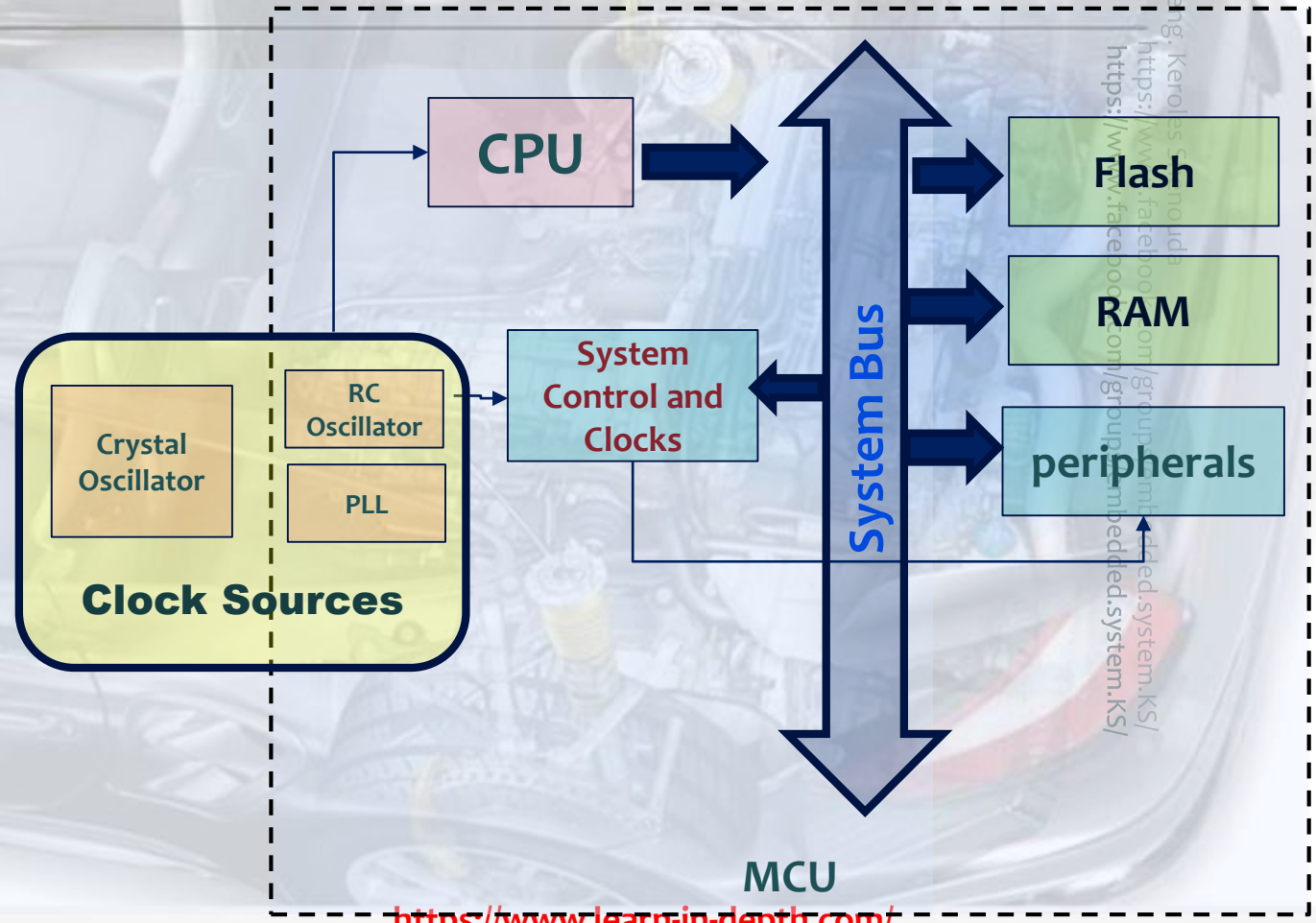


#LEARN_IN DEPTH

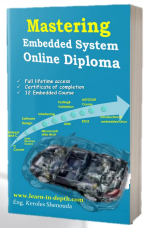
#Be_professional_in_embedded_system

MCU Clock Sources

- MCU External
  - Crystal Oscillator
- MCU Internal
  - The RC Oscillator
  - The PLL (Phase Locked Loop)



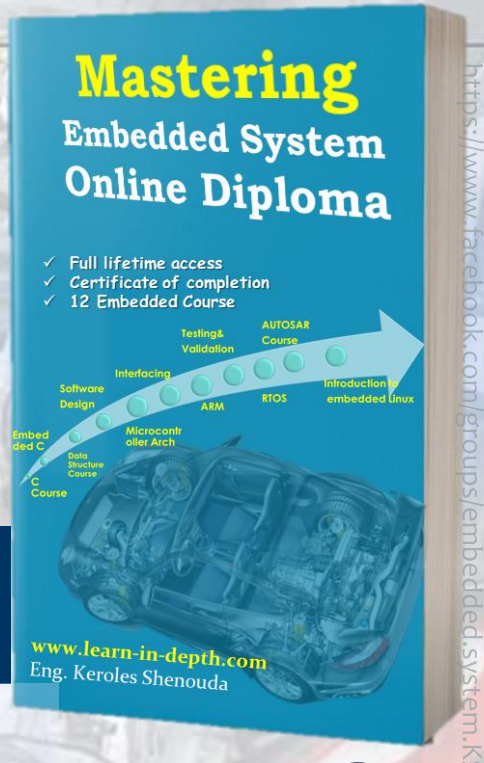
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

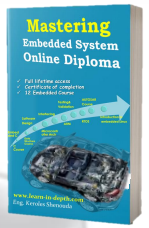
#Be_professional_in_embedded_system

Steps to find out the Clock tree in any MCU



LEARN-IN-DEPTH
Be professional in embedded system

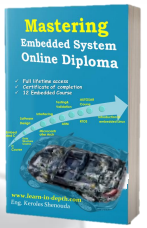
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Steps to find out the Clock tree in any MCU

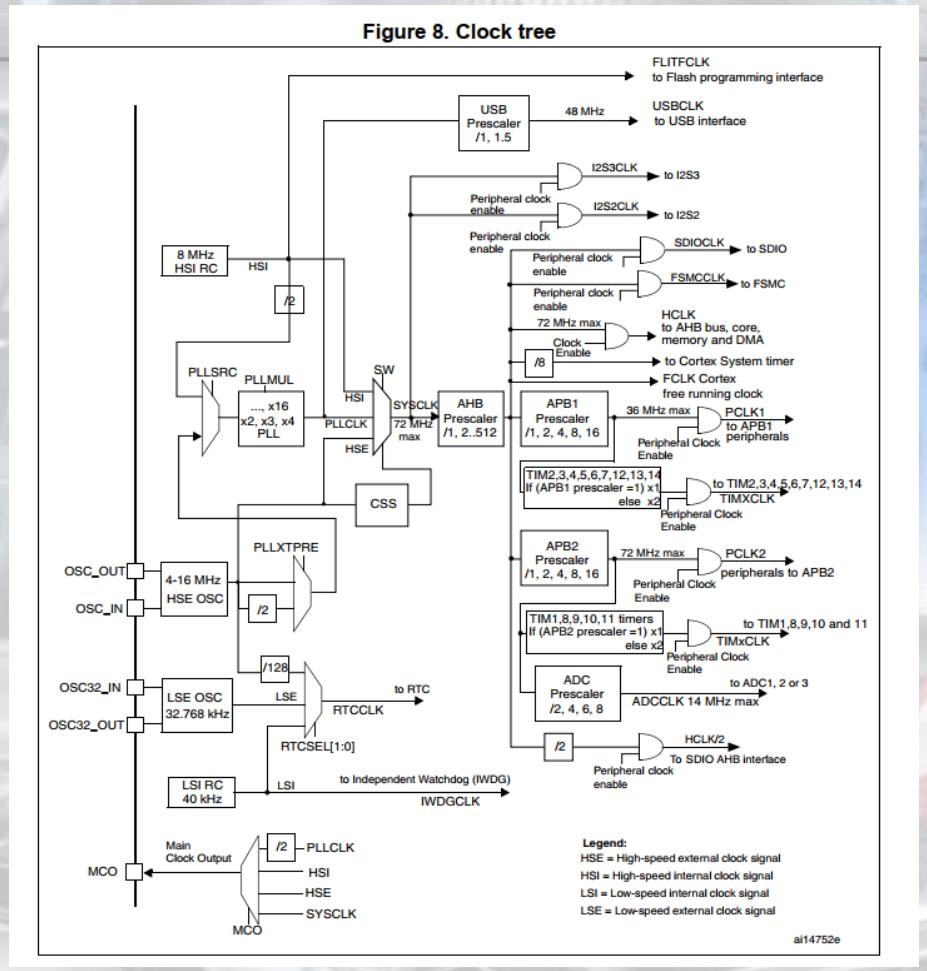
- 1. Figure out the "Clock Management Unit" for your MCU
- 2. Figure out the Base address and "Clock Management Unit" Section
- 3. Navigate inside this Section the " Clock tree " Diagram

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

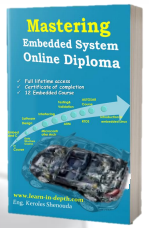


Steps to find out the Clock tree in any MCU Stm32F103XX

- 1. Figure out the "Clock Management Unit" for your MCU
- 2. Figure out the Base address and "Clock Management Unit" Section
- 3. Navigate inside this Section the " Clock tree " Diagram



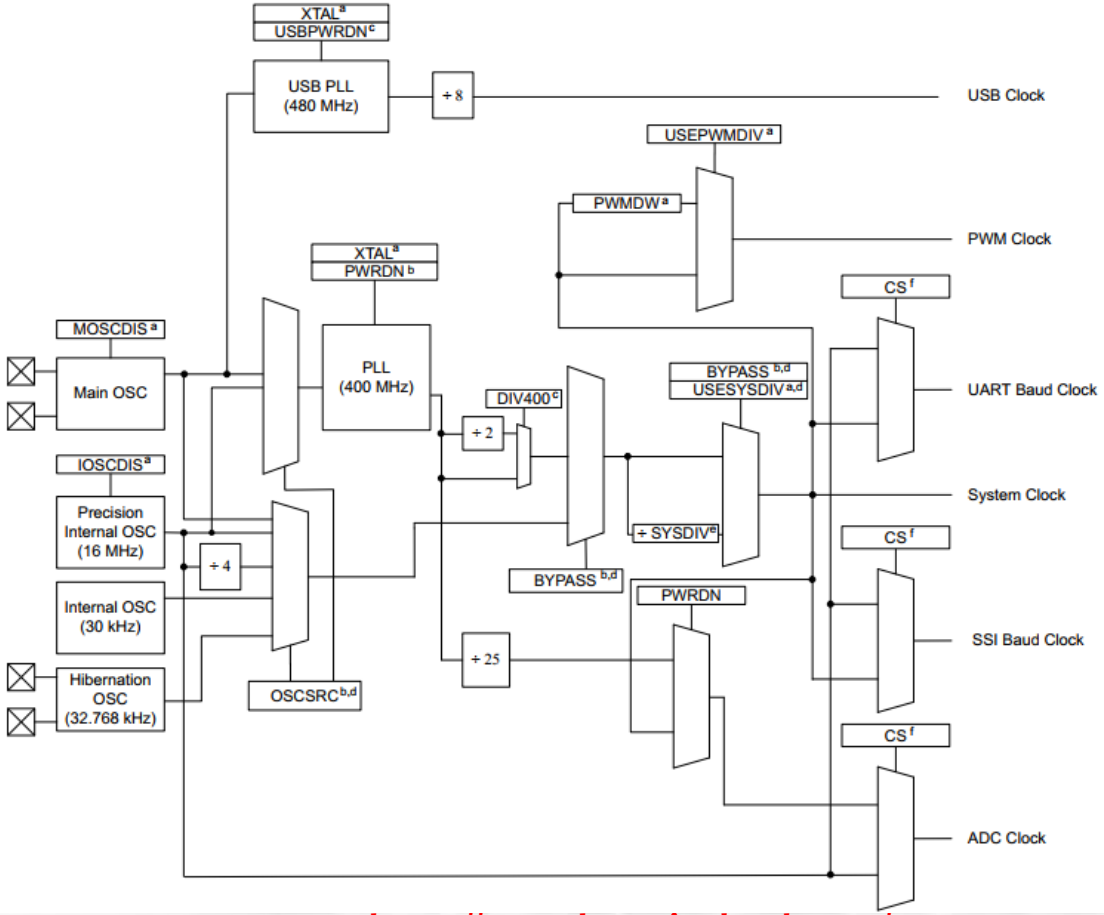
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



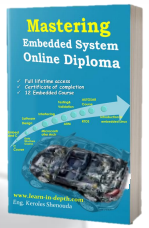
Steps to find out the Clock tree in any MCU TM4C123

- 1. Figure out the "Clock Management Unit" for your MCU
- 2. Figure out the Base address and "Clock Management Unit" Section
- 3. Navigate inside this Section the " Clock tree " Diagram

Figure 5-5. Main Clock Tree



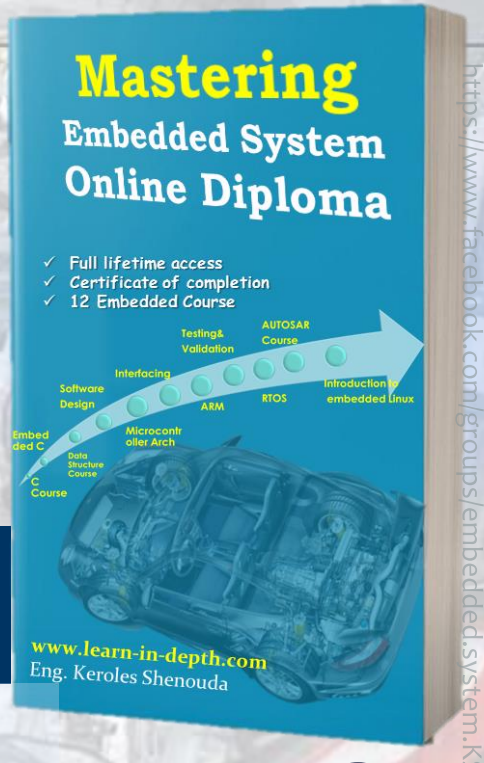
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

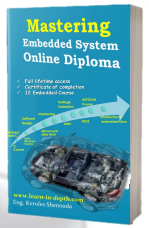
Understanding Clock Tree

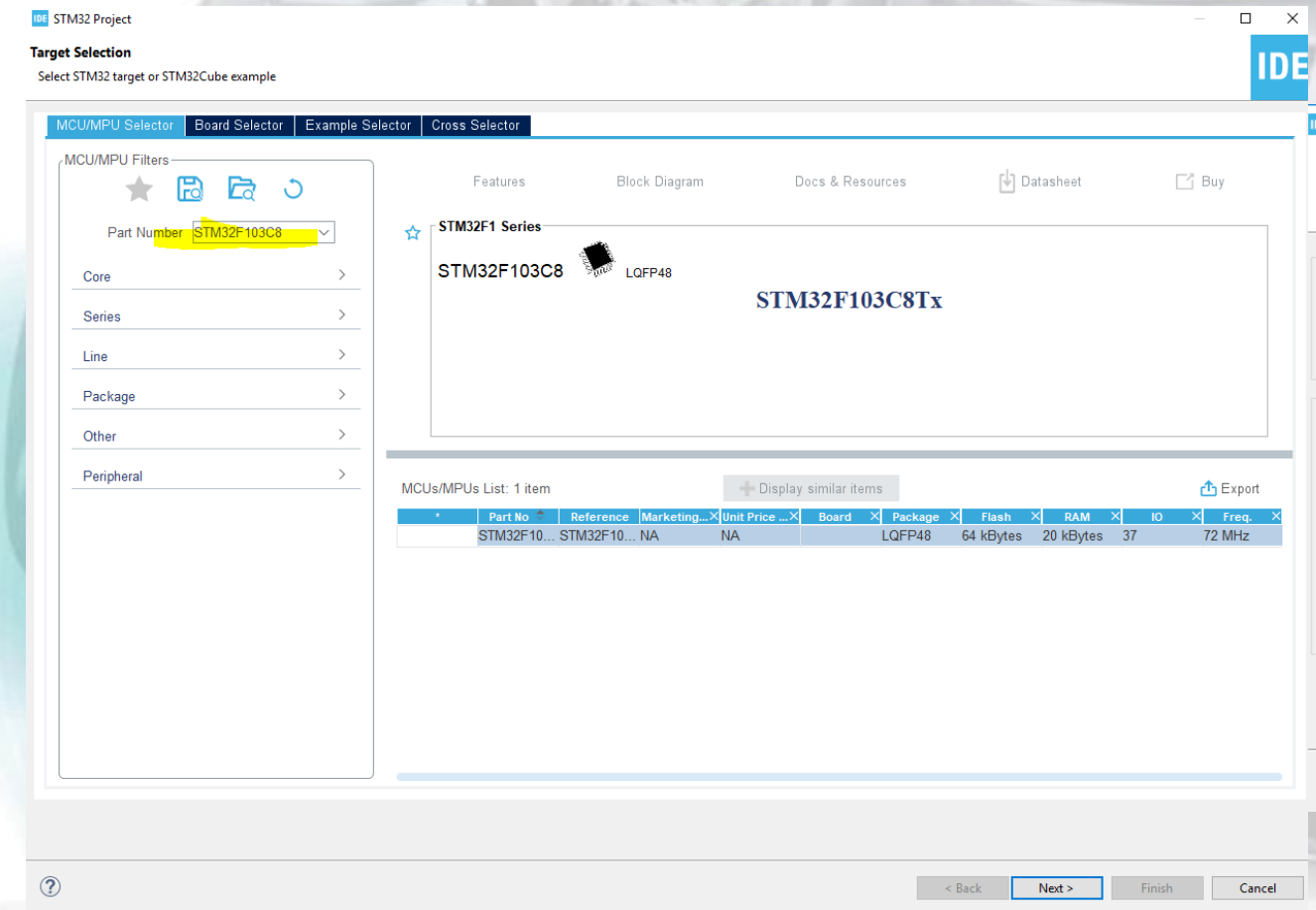


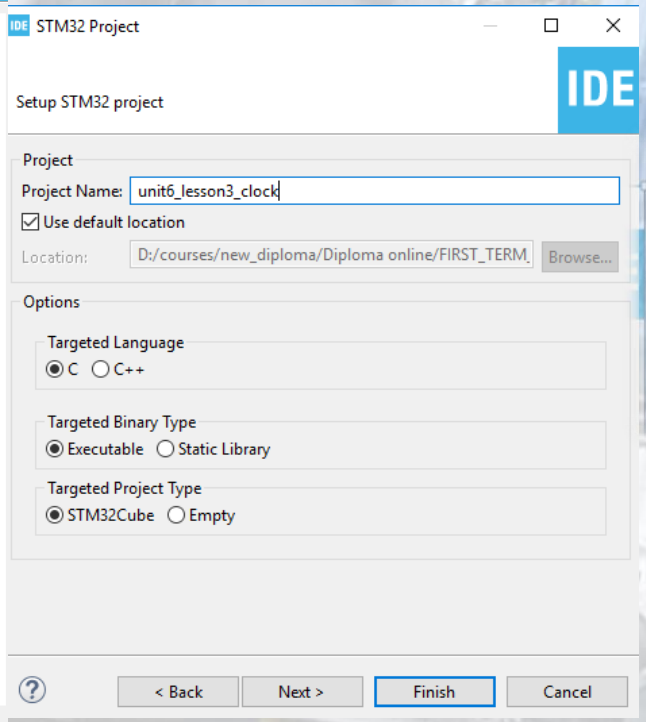
LEARN-IN-DEPTH
Be professional in embedded system

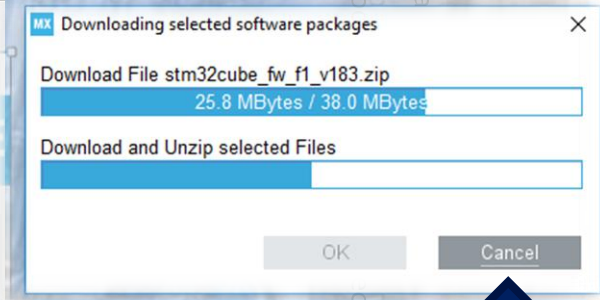
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Open Stm32CubeIDE



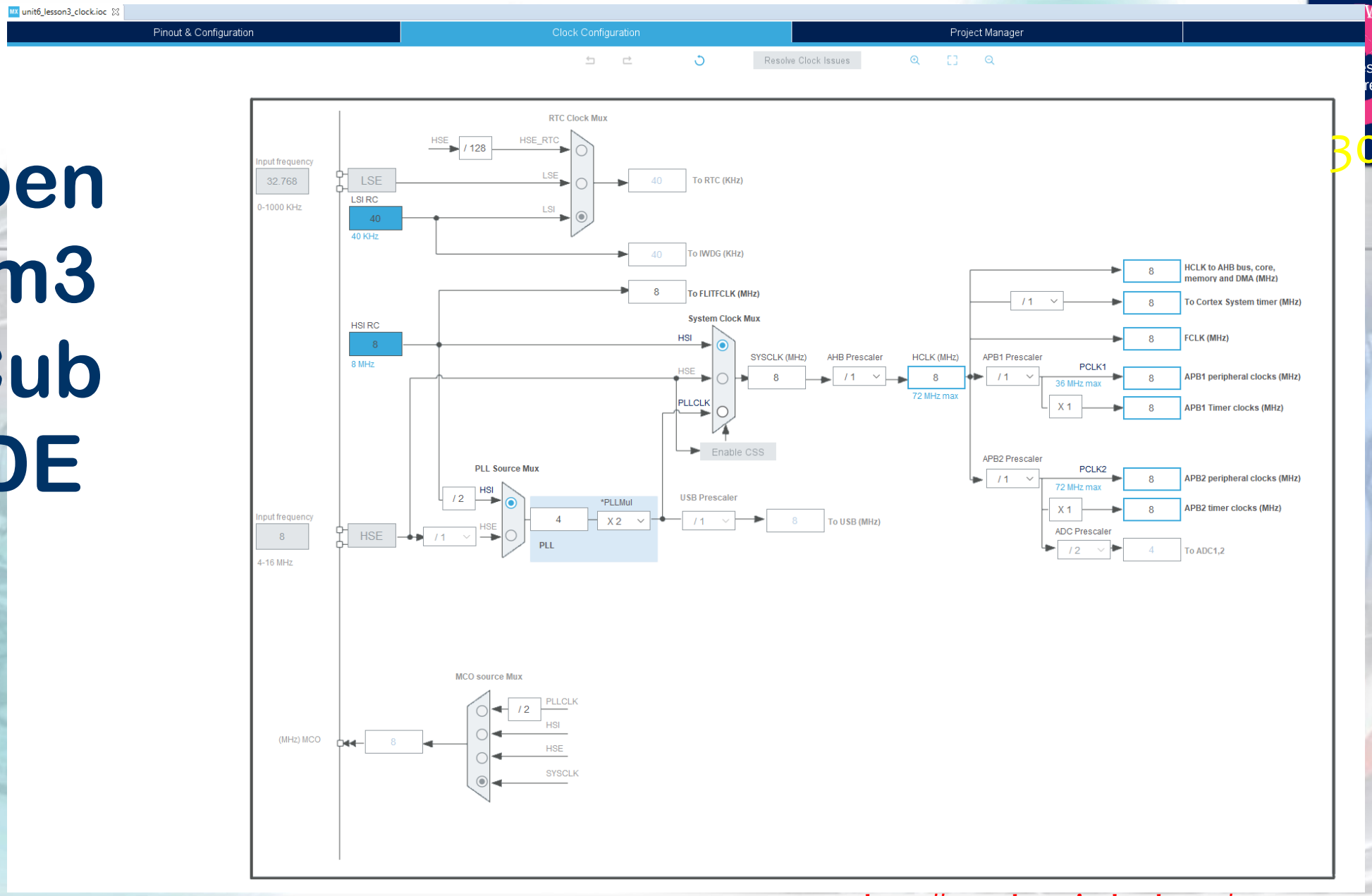


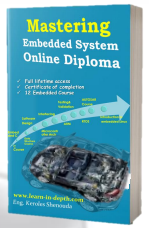




Click Cancel

Open Stm3 2Cub eIDE





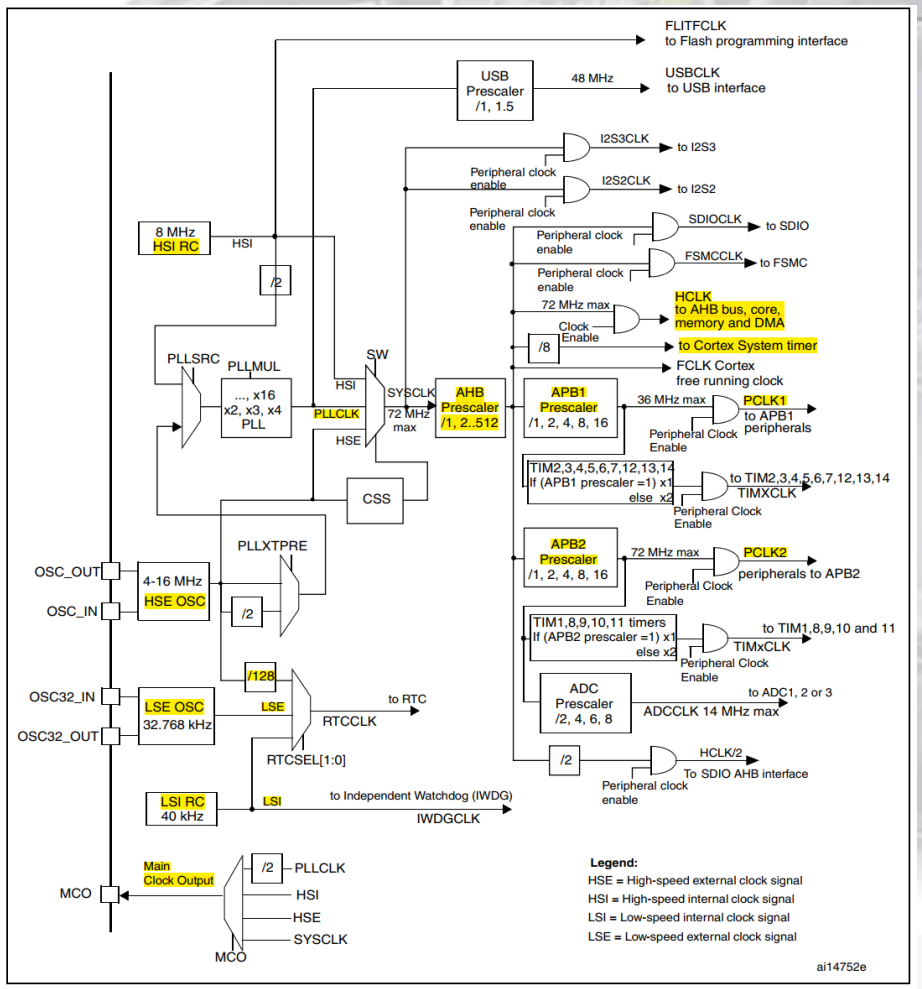
#LEARN_IN DEPTH

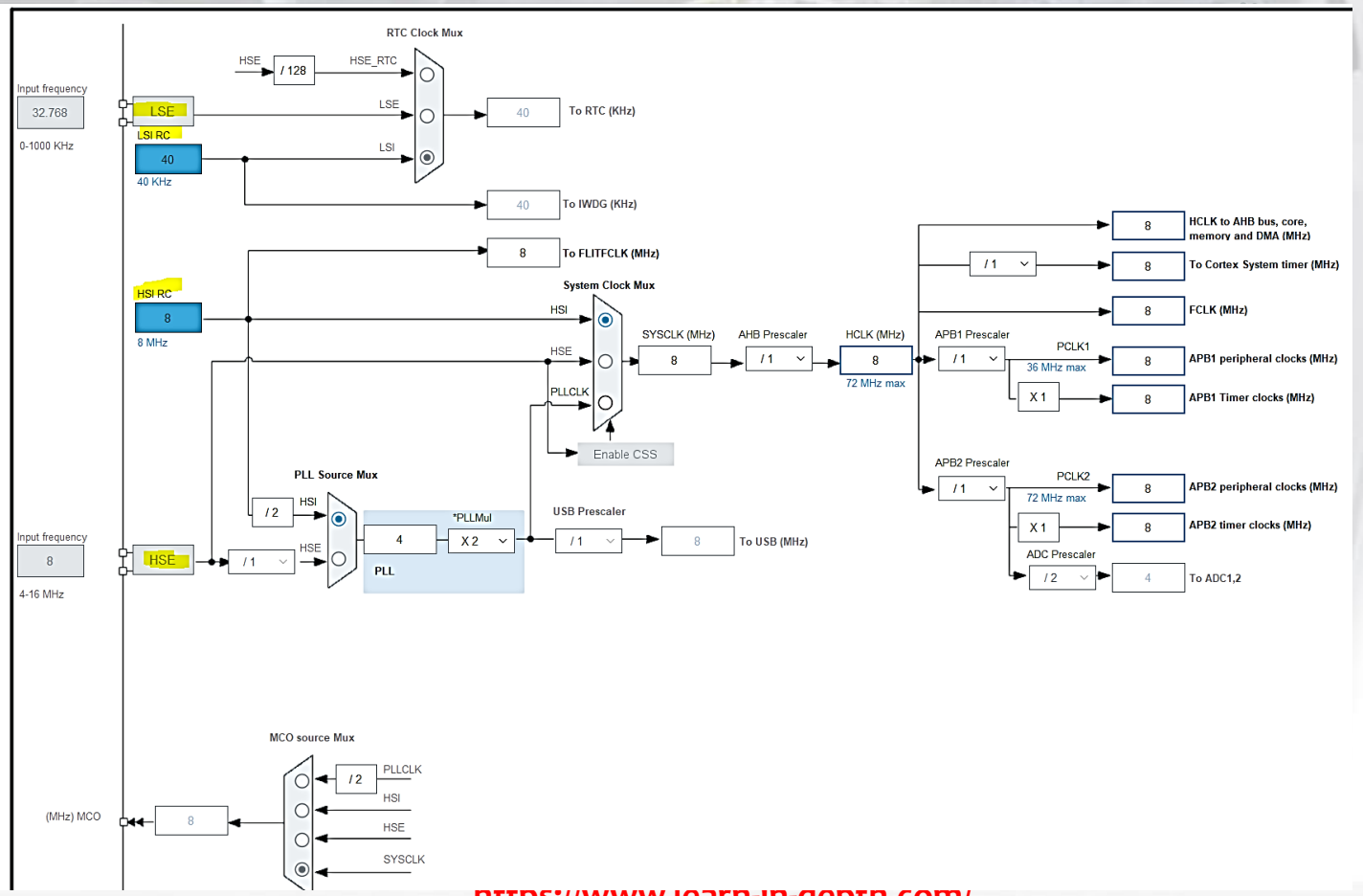
#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

You have to calculate Clocks before programming any module

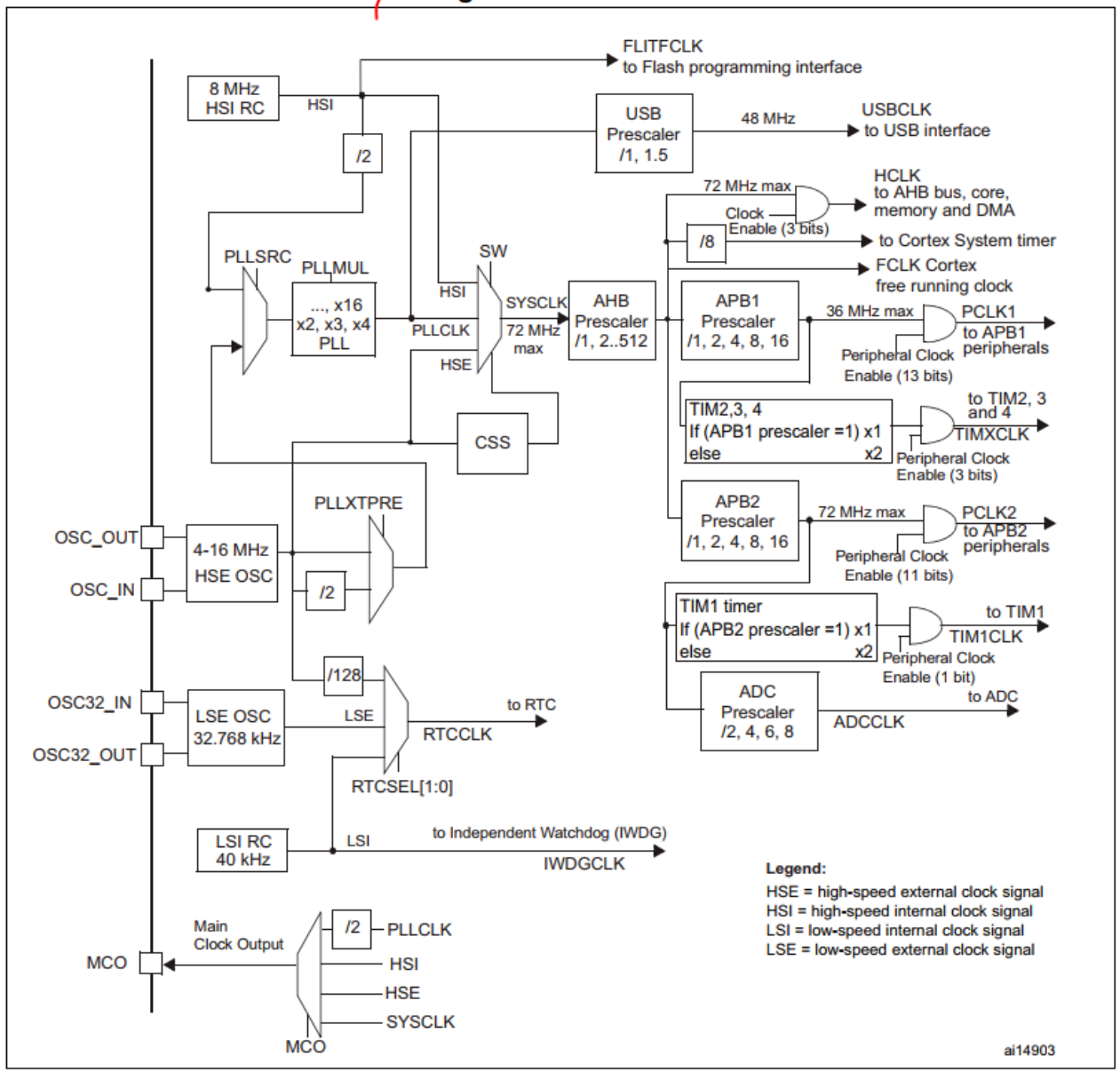
```
void main()
{ Clock_init(); ….. }
```

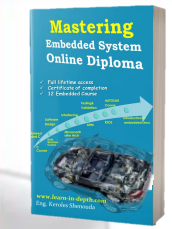




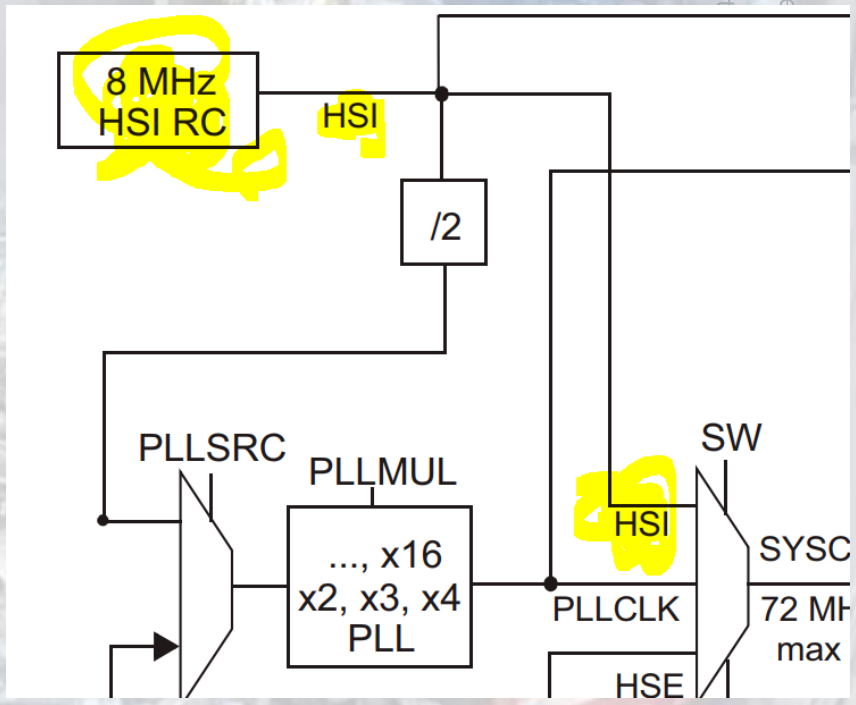
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Figure 2. Clock tree

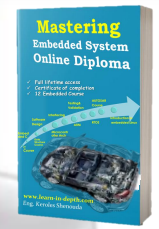




The RC Oscillator (HSI)

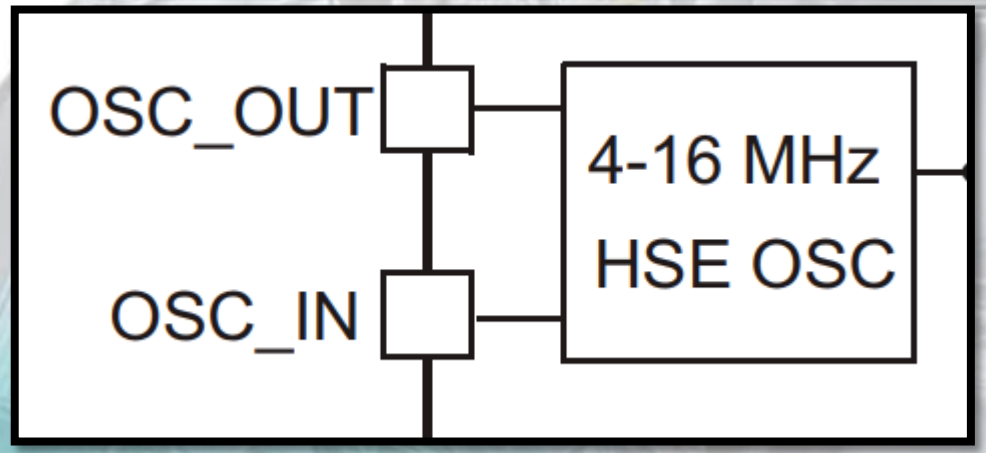


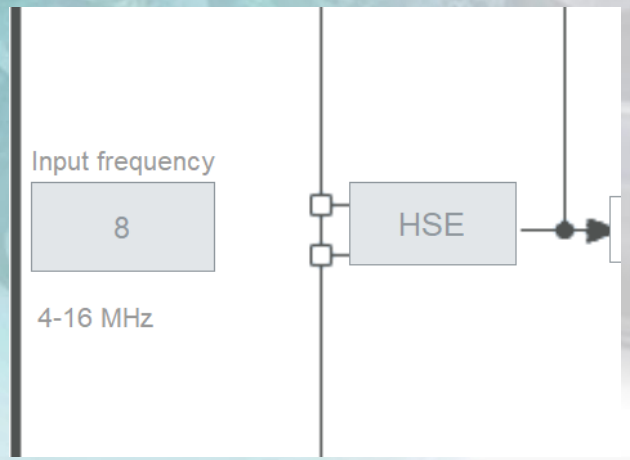
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

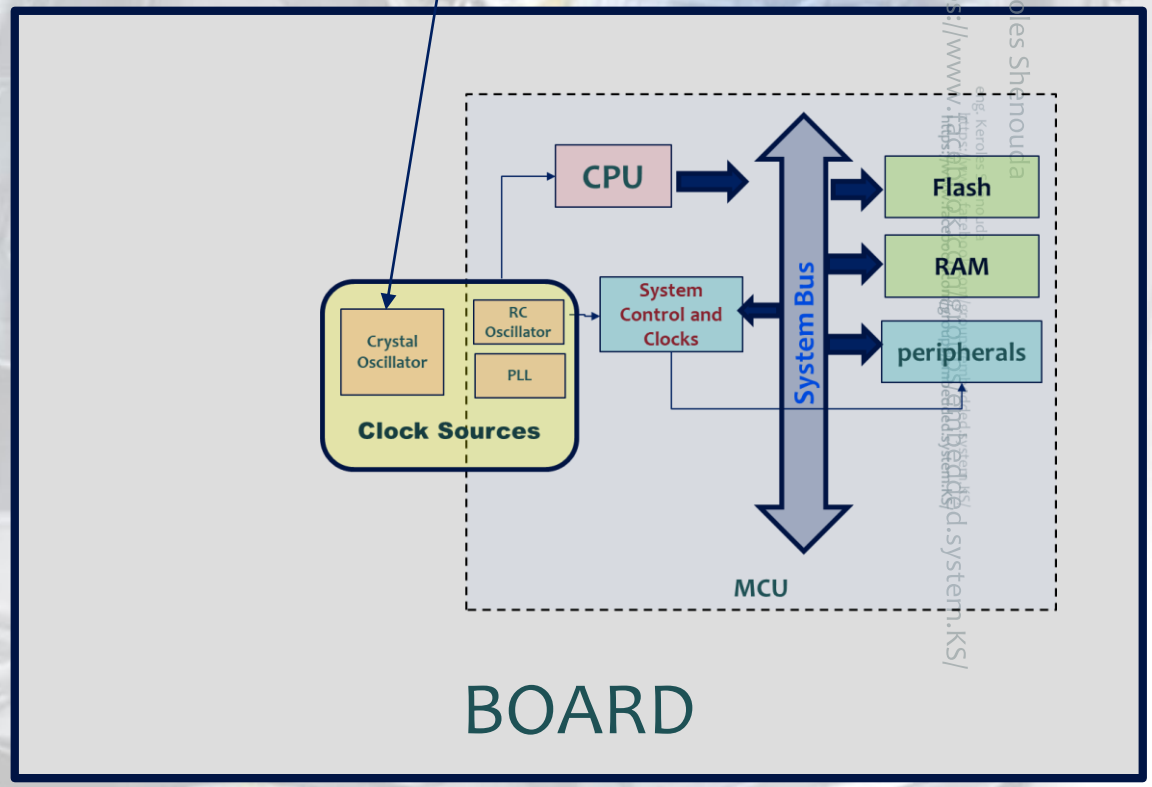


Crystal Oscillator (HSE)

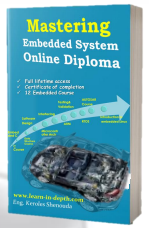
HSE Values is Specified in the Board Specs Not In the MCU TRM







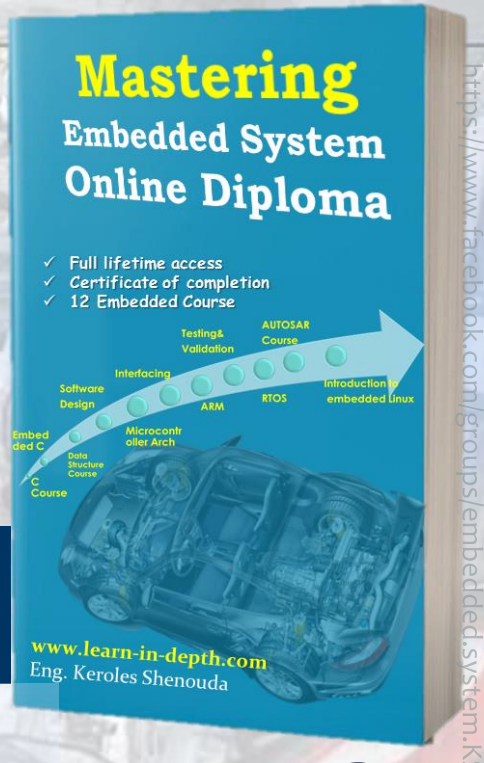
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

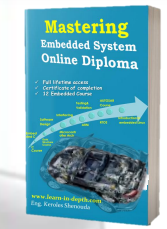
#Be_professional_in_embedded_system

external crystal to SYSCLK examples

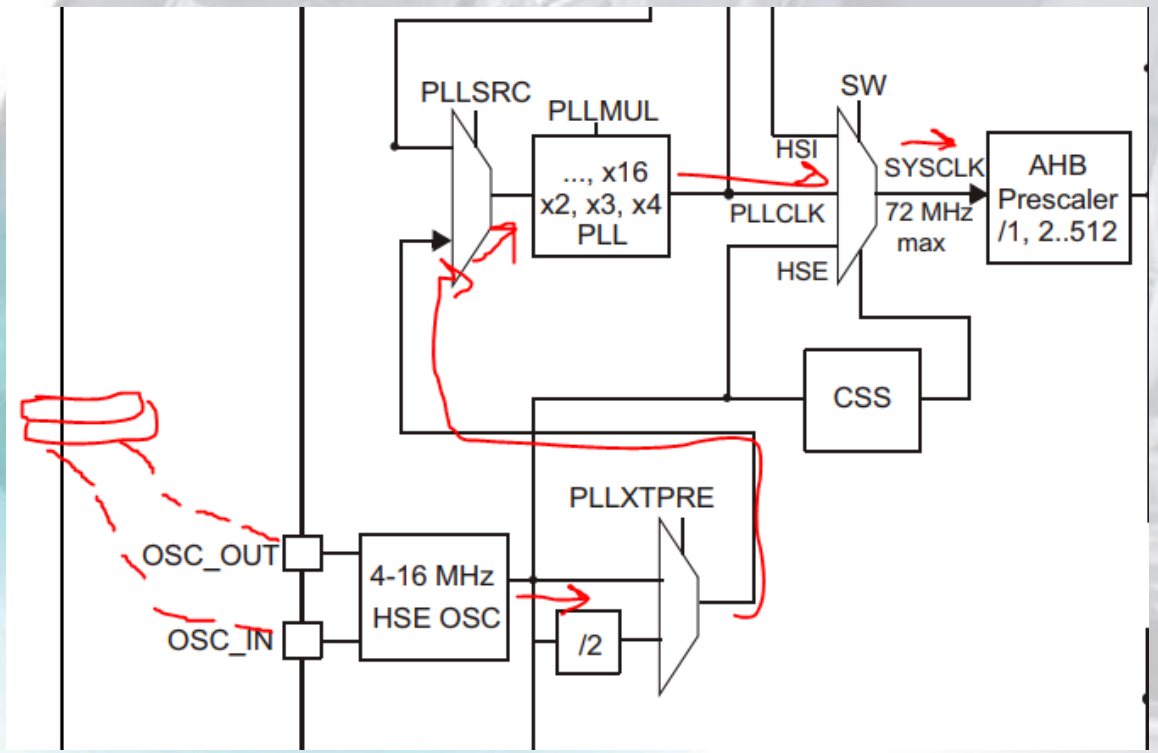


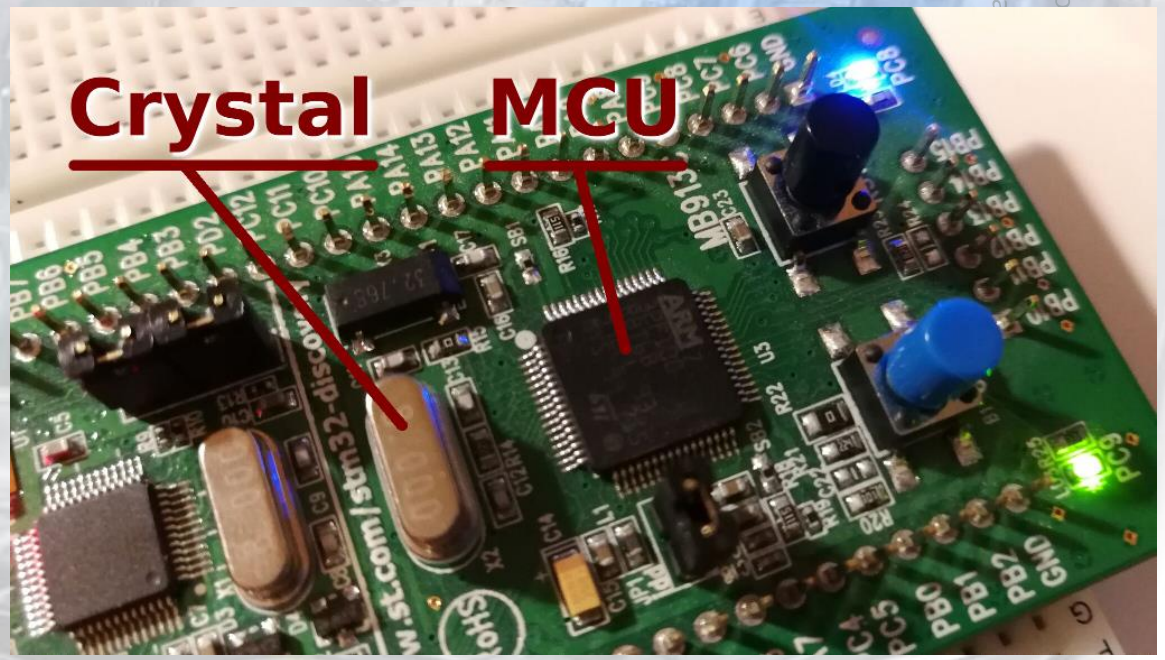
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

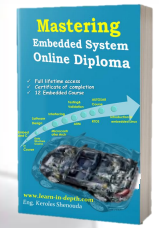


Crystal Oscillator (HSE)
Here, we can plot the path from the external crystal to SYSCLK

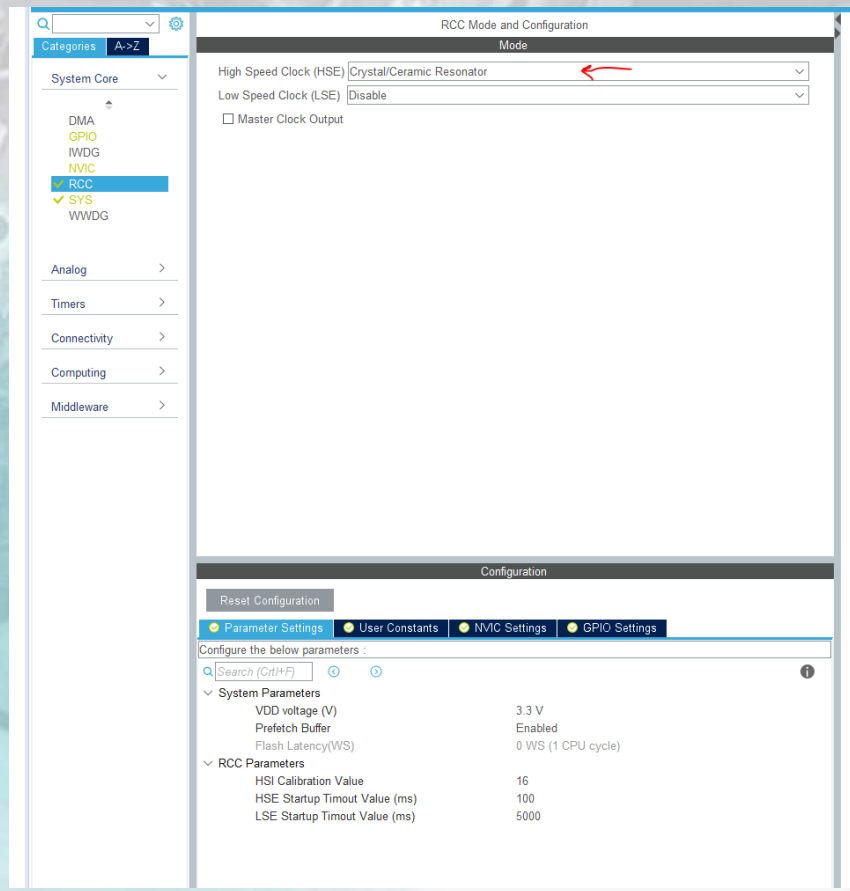


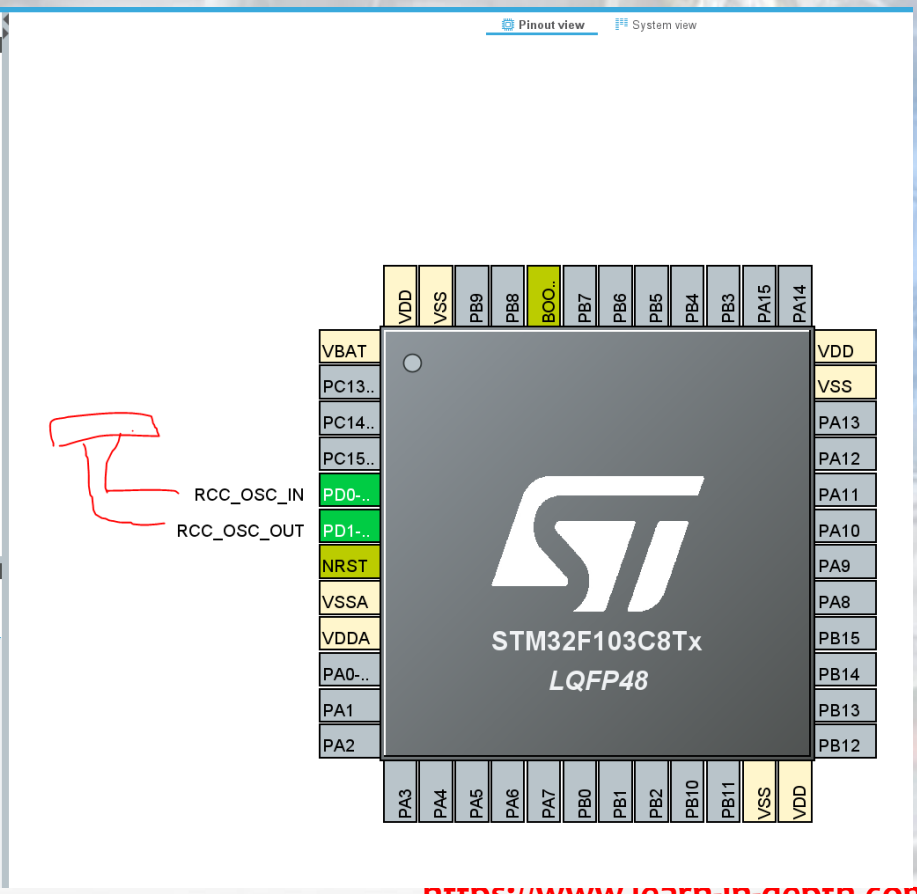


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



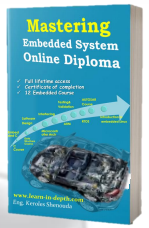
Enable crystal





https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

In this Board HSE: Crystal resonators is Used



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

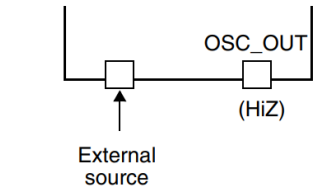
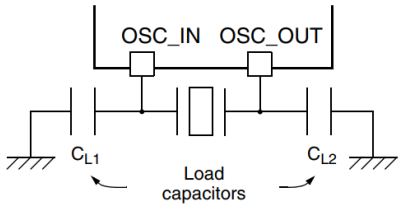
7.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

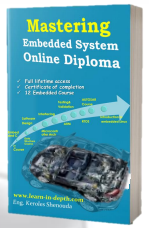
- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

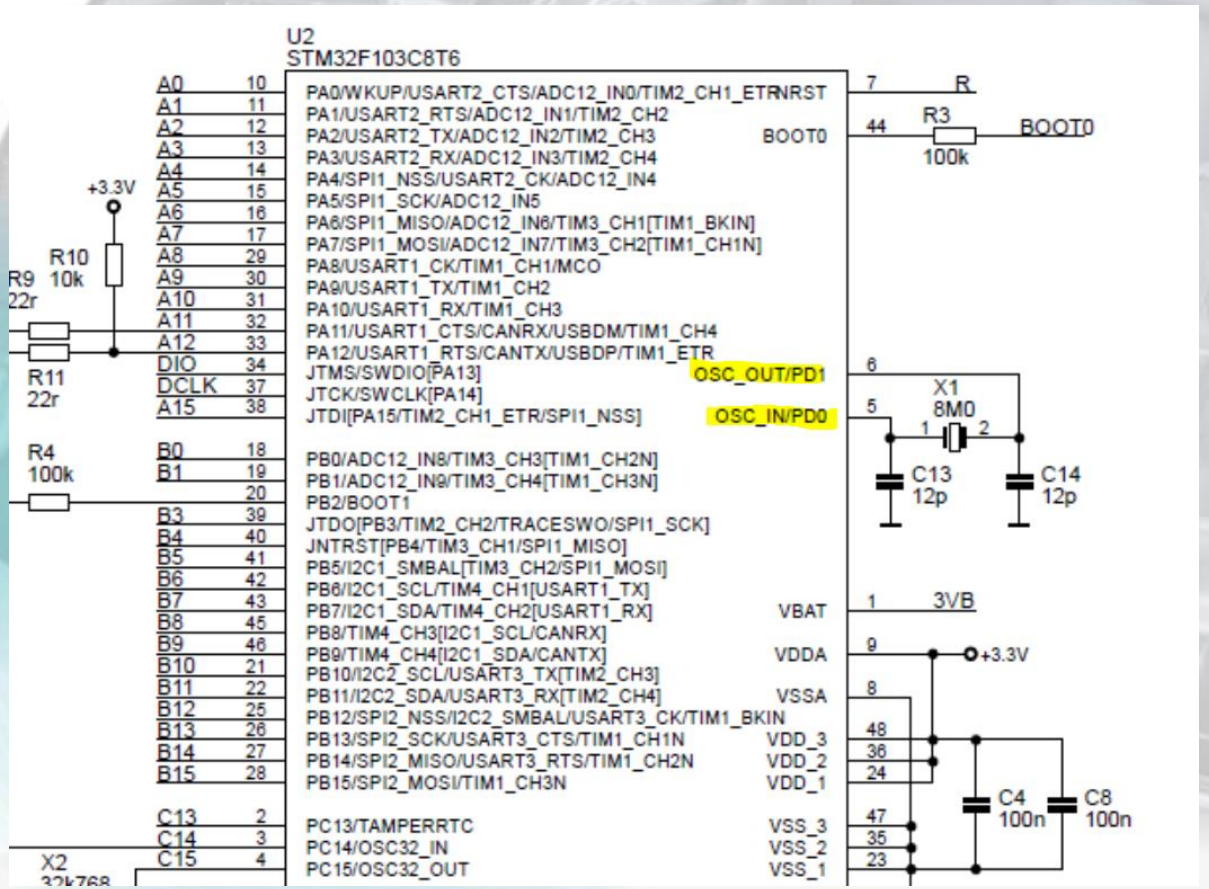
Figure 9. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

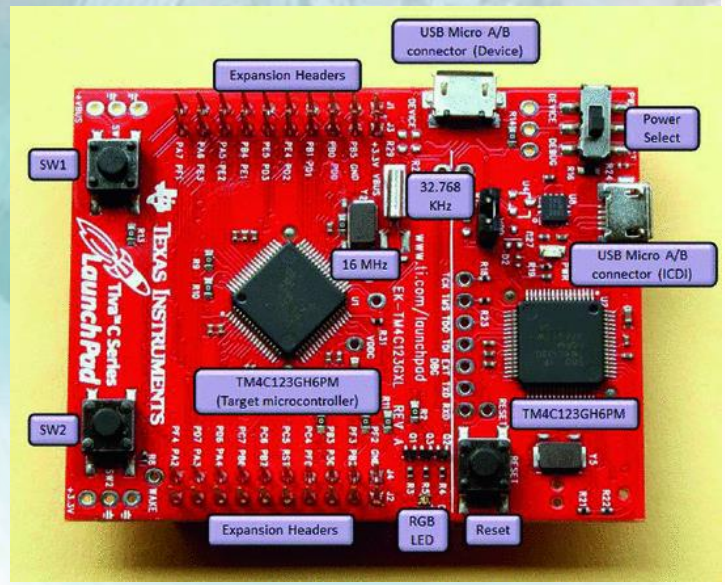


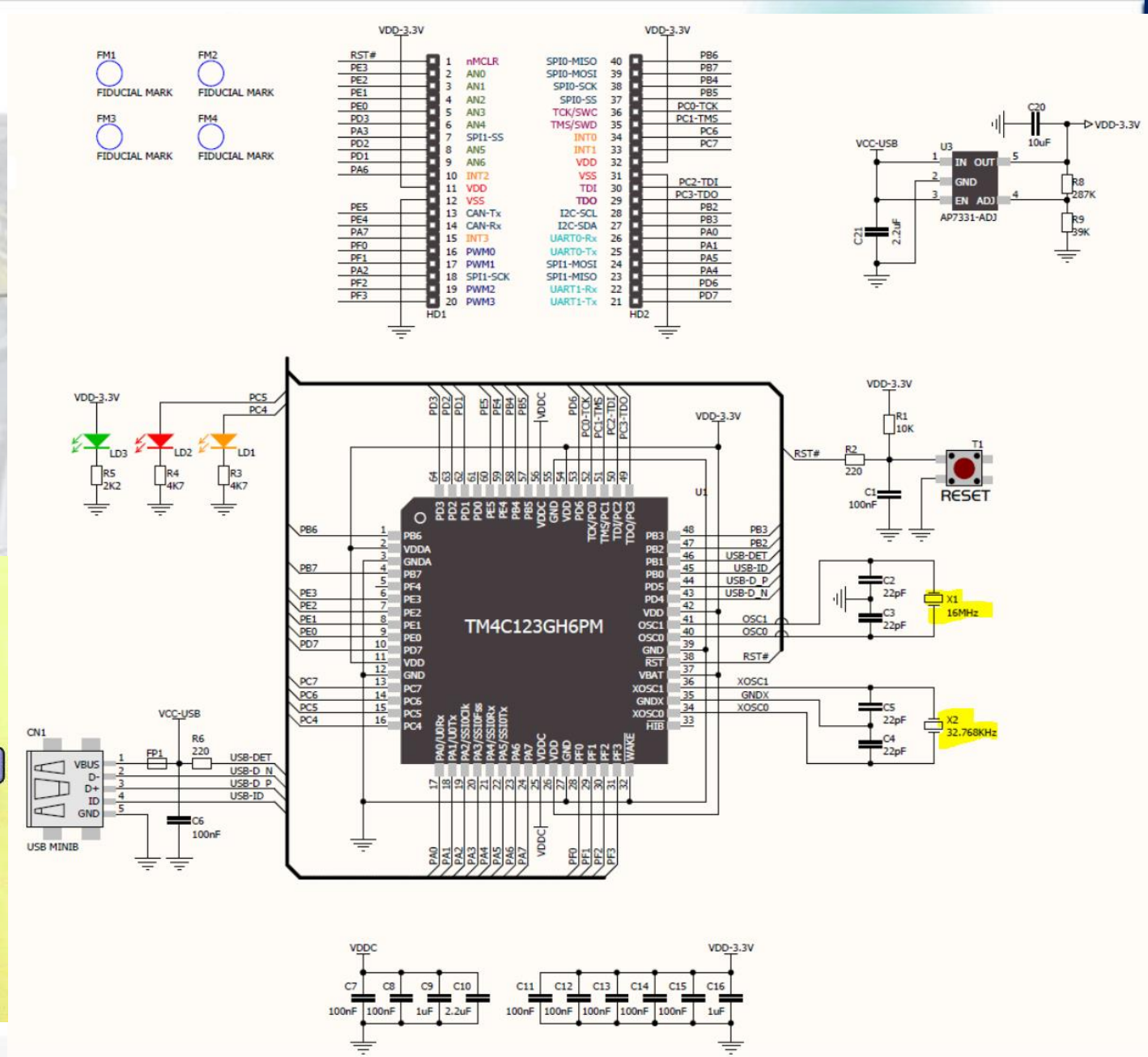
STM32F103C8T6 board, alias Bluepill

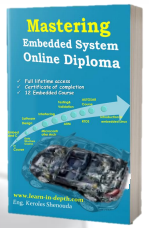


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

TIVAC Board Example

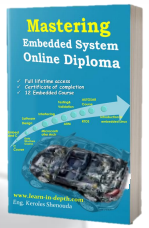






#LEARN_IN DEPTH

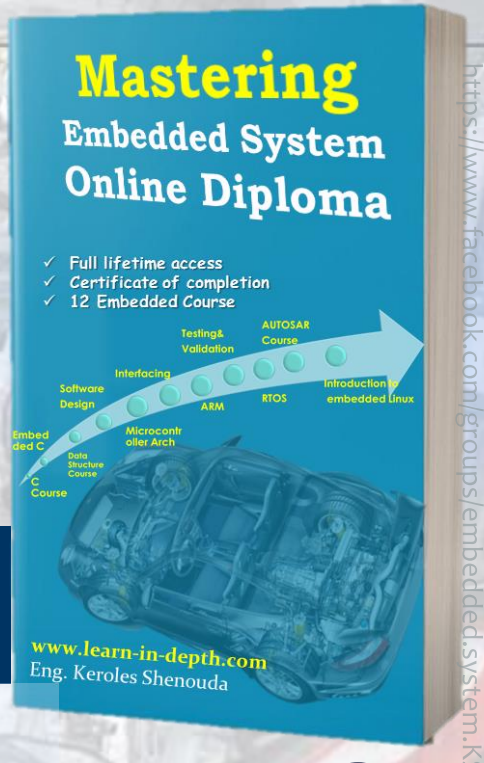
#Be_professional_in_embedded_system



#LEARN_IN DEPTH

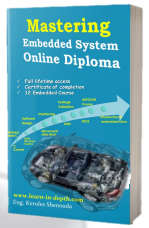
#Be_professional_in_embedded_system

High SPEED INTERNAL CLOCK SIGNAL (HIS)



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

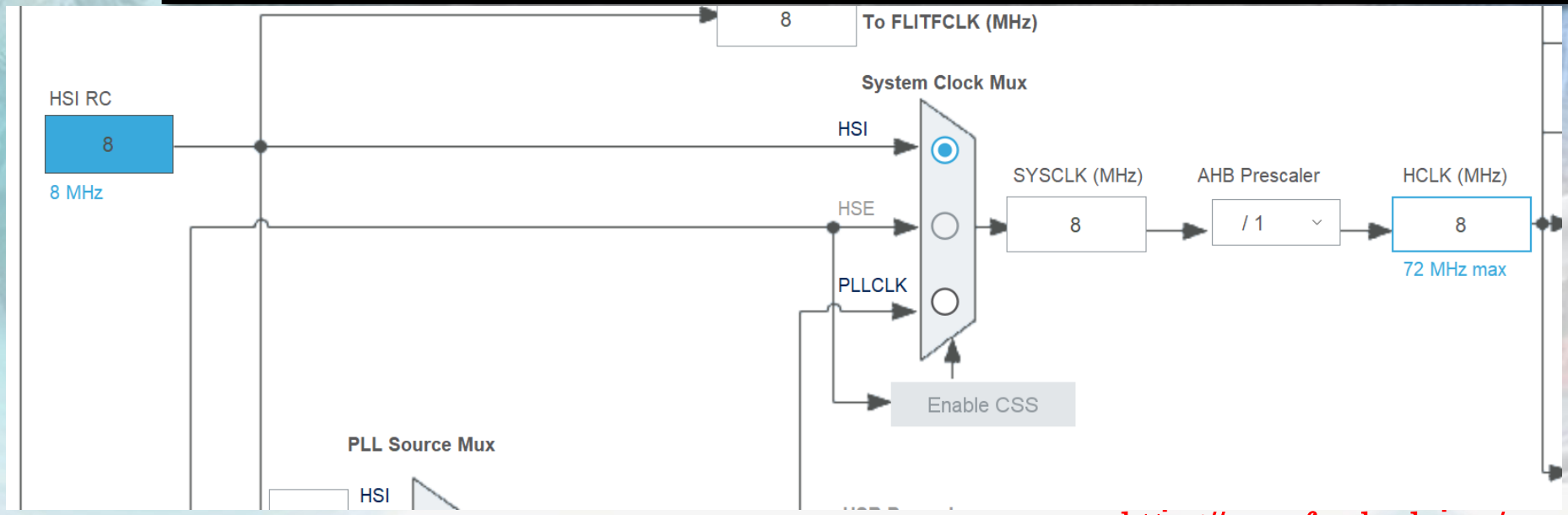


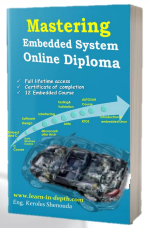
HSI

7.2.2 HSI clock

The HSI clock signal is generated from an internal 8 MHz RC Oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.

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.

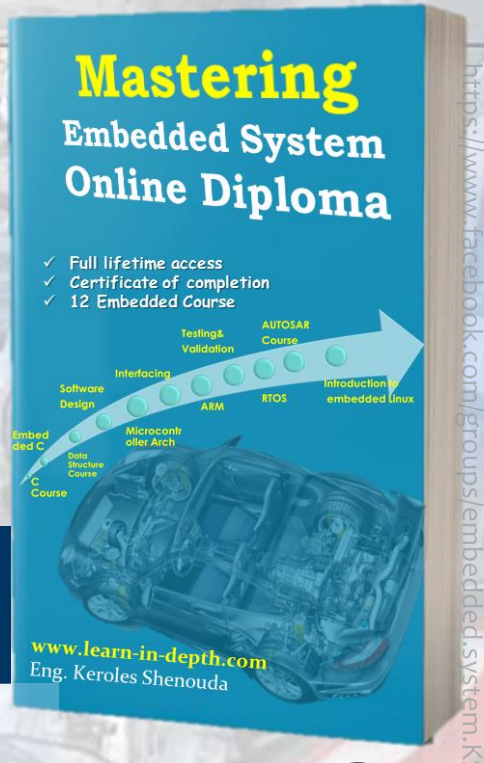




#LEARN_IN DEPTH

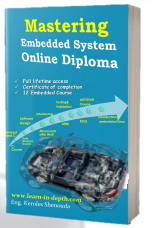
#Be_professional_in_embedded_system

(Clock Design) failures example

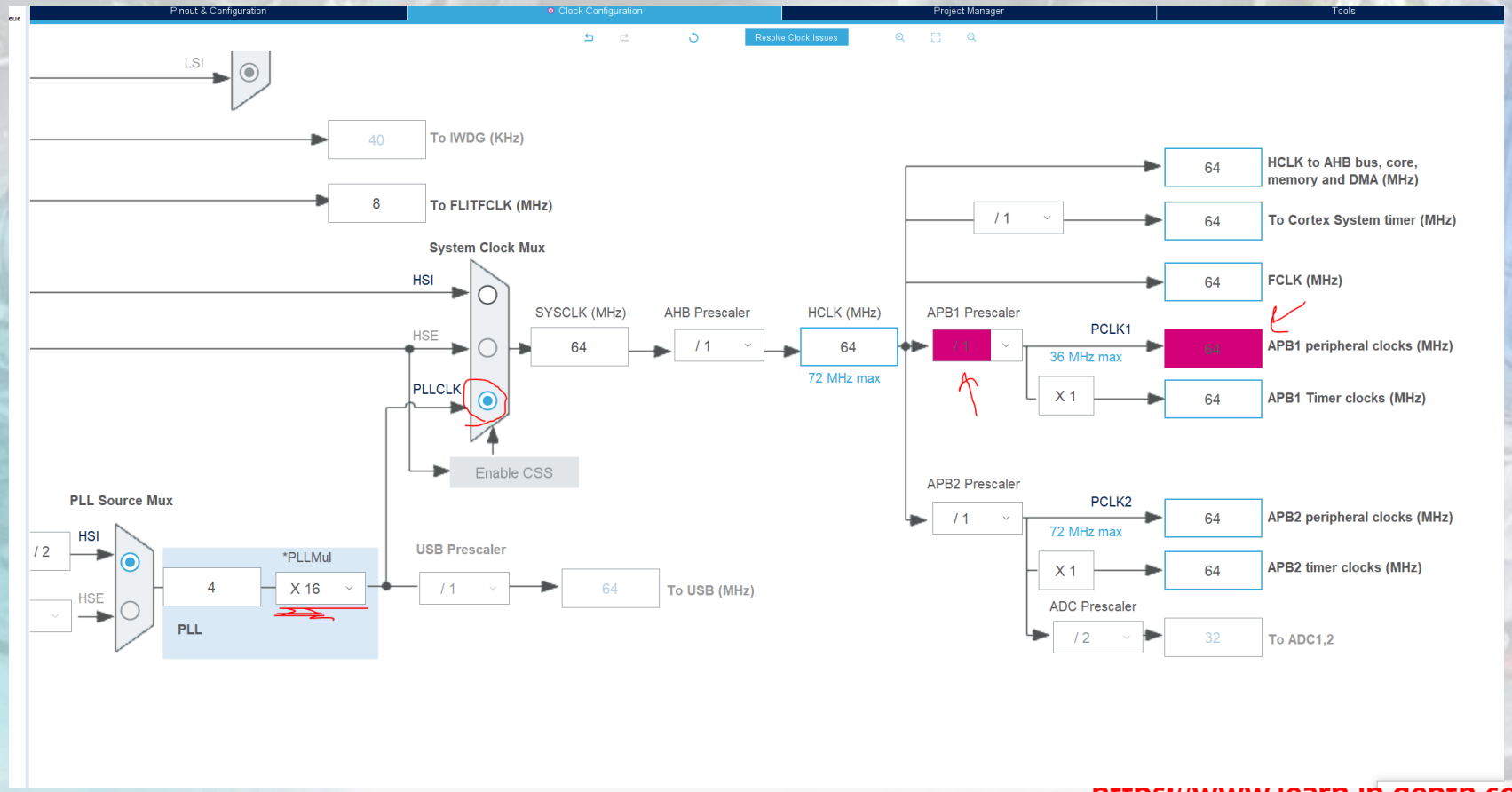


LEARN-IN-DEPTH
Be professional in embedded system

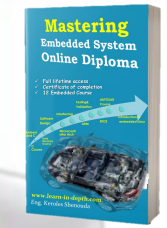
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Think in depth (what is the issue here) ?



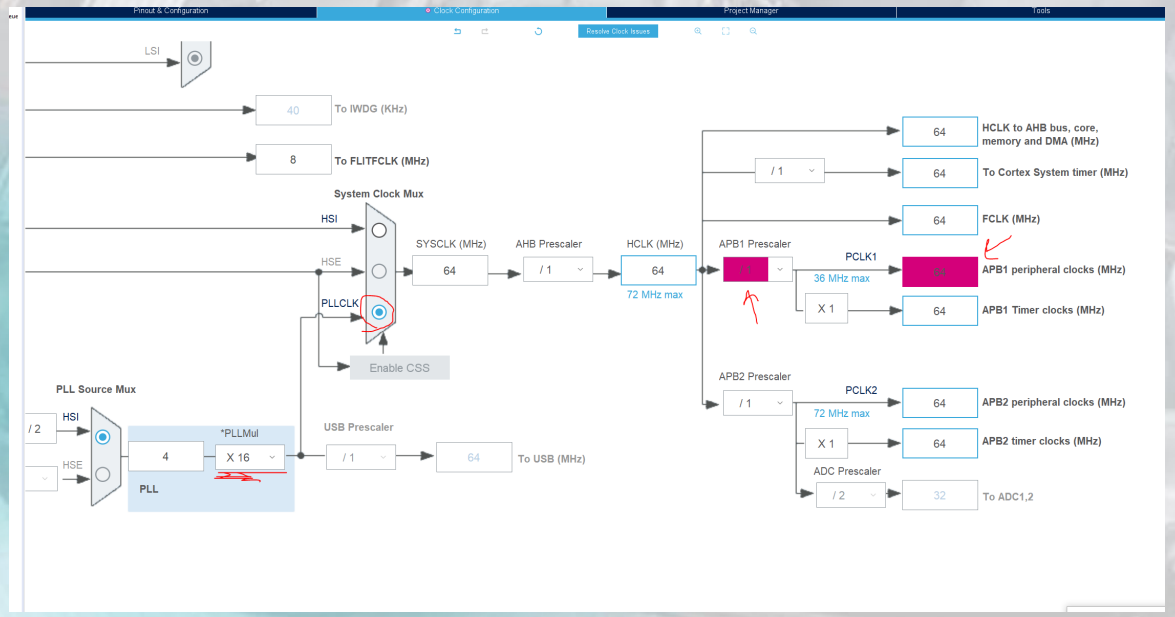
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

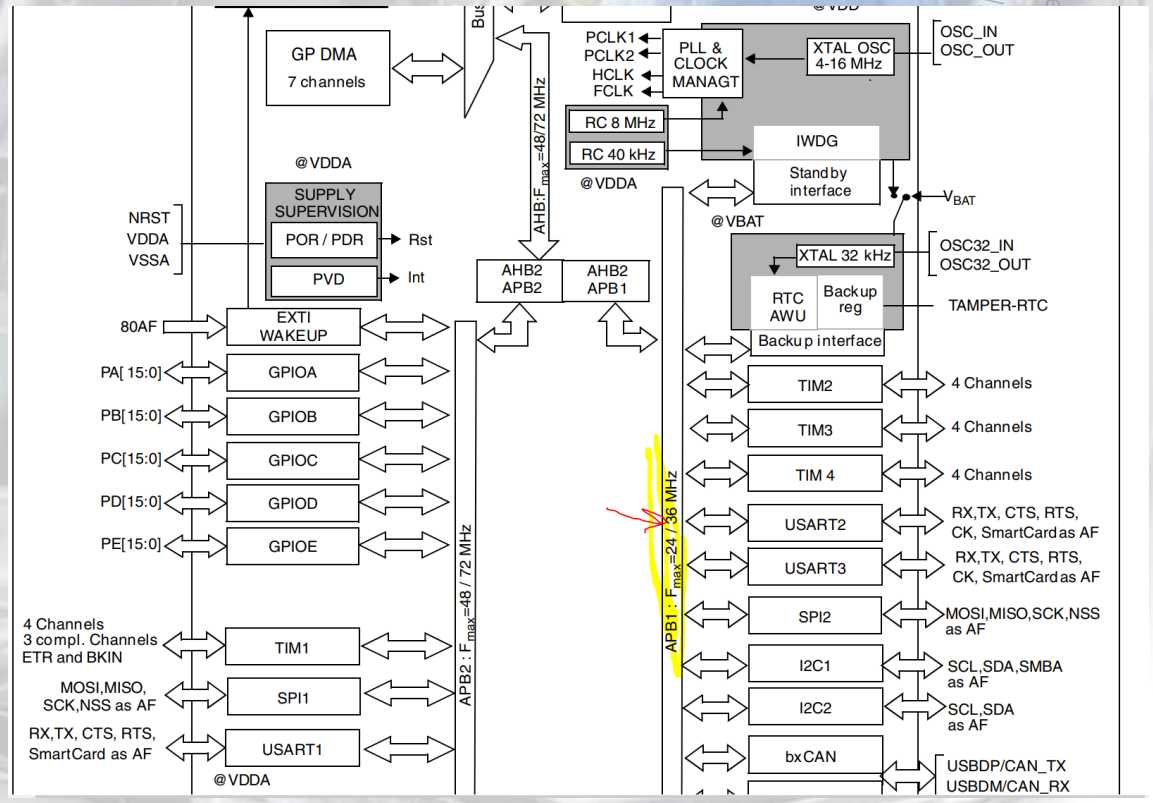


#LEARN_IN DEPTH

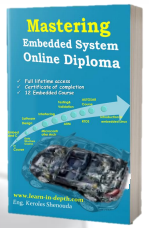
#Be_professional_in_embedded_system

Think in depth (what is the issue here) ?

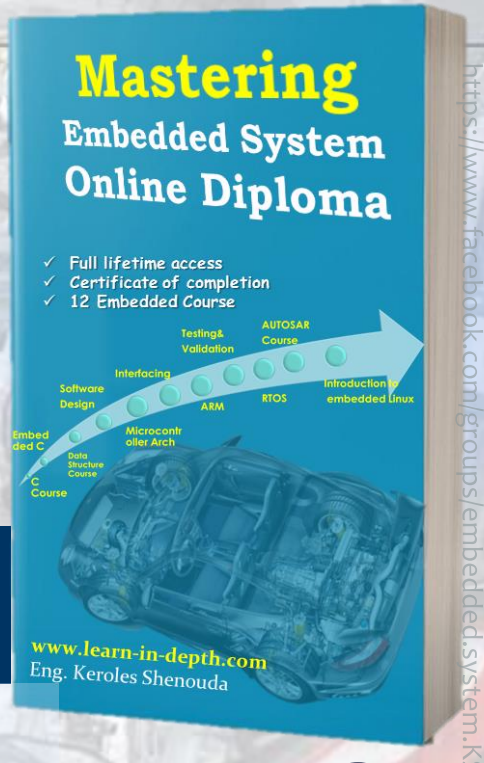




https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

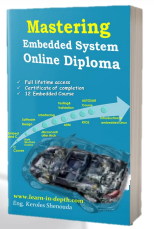


RCC registers



LEARN-IN-DEPTH
Be professional in embedded system

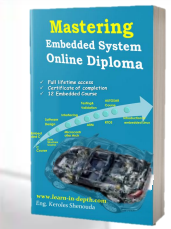
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



RCC registers

- 7.3 RCC registers
  - 7.3.1 Clock control register (RCC_CR)
  - 7.3.2 Clock configuration register (RCC_CFGR)
  - 7.3.3 Clock interrupt register (RCC_CIR)
  - 7.3.4 APB2 peripheral reset register (RCC_APB2RSTR)
  - 7.3.5 APB1 peripheral reset register (RCC_APB1RSTR)
  - 7.3.6 AHB peripheral clock enable register (RCC_AHBENR)
  - 7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)
  - 7.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)
  - 7.3.9 Backup domain control register (RCC_BDCR)
  - 7.3.10 Control/status register (RCC_CSR)
  - 7.3.11 RCC register map

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



7.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

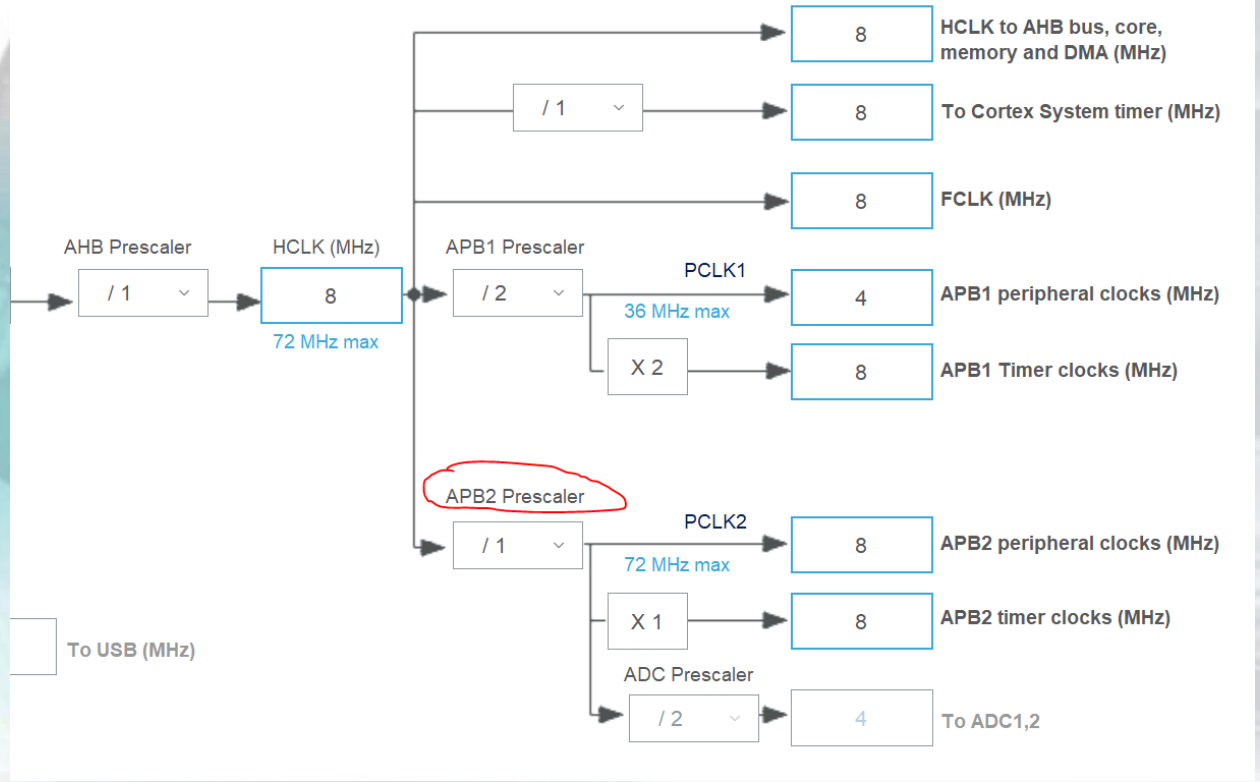
Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

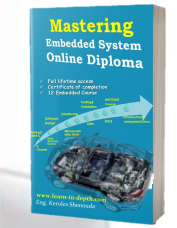


Bits 10:8 **PPRE1:** APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).
Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

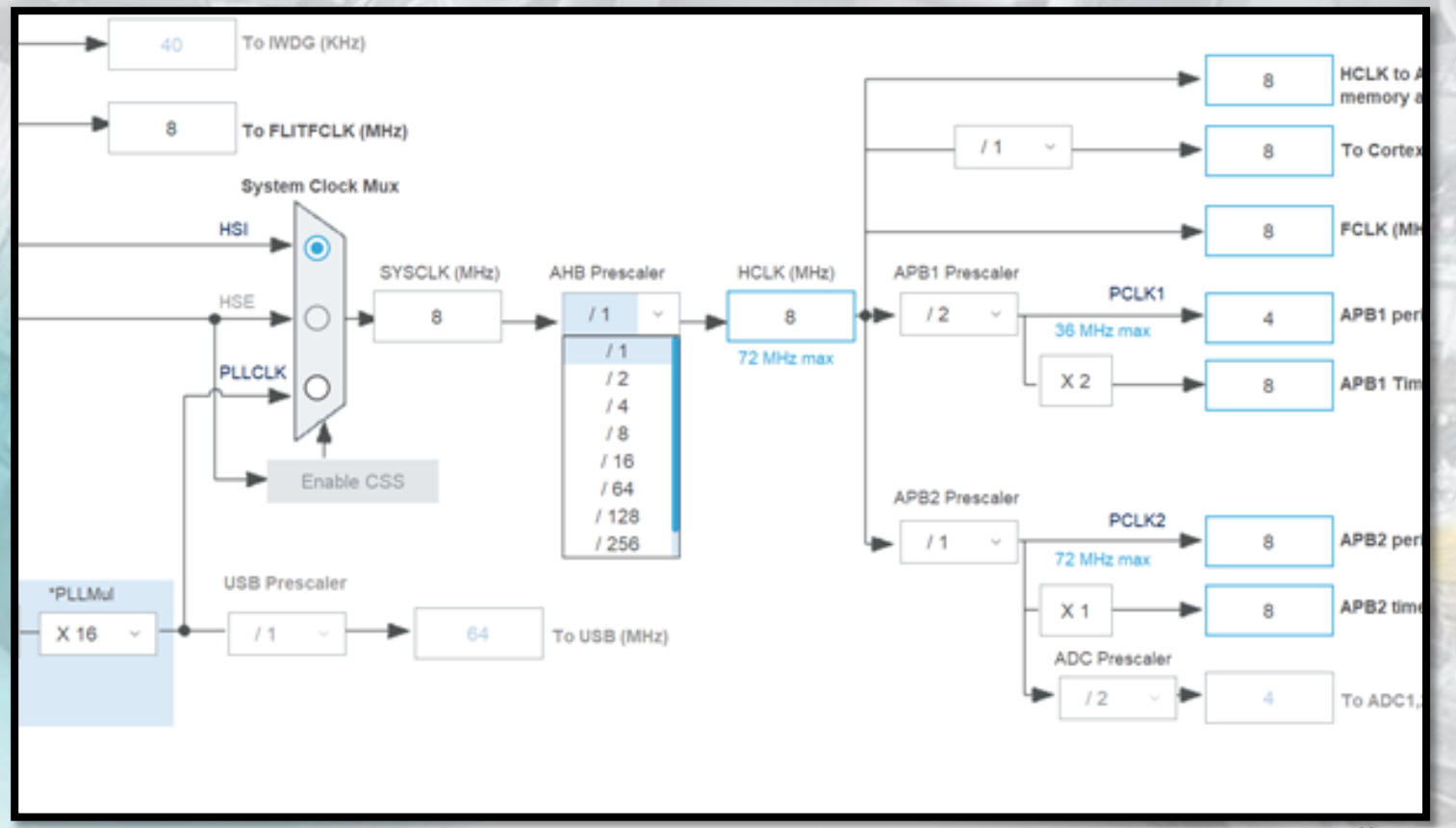
Bits 13:11 **PPRE2:** APB high-speed prescaler (APB2)

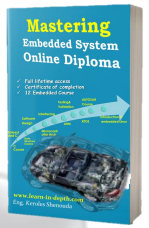
Set and cleared by software to control the division factor of the APB high-speed clock (PCLK2).
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16



Bits 7:4 **HPRE:** AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.
0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

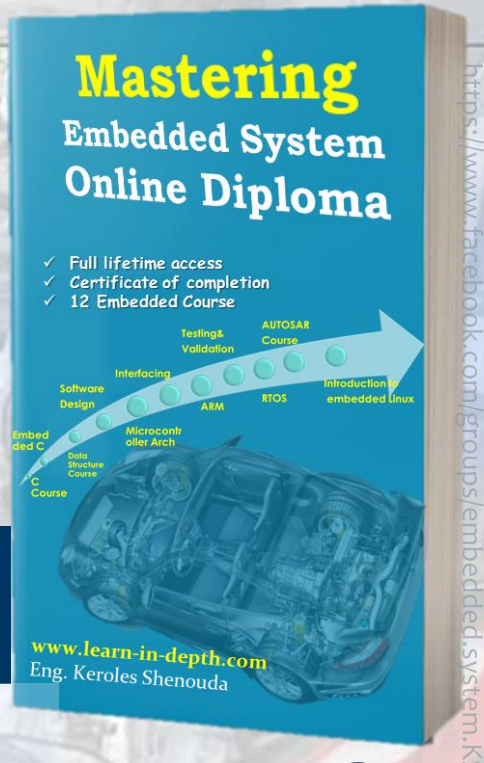




#LEARN_IN DEPTH

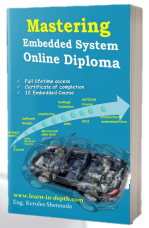
#Be_professional_in_embedded_system

Peripherals Clocks



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

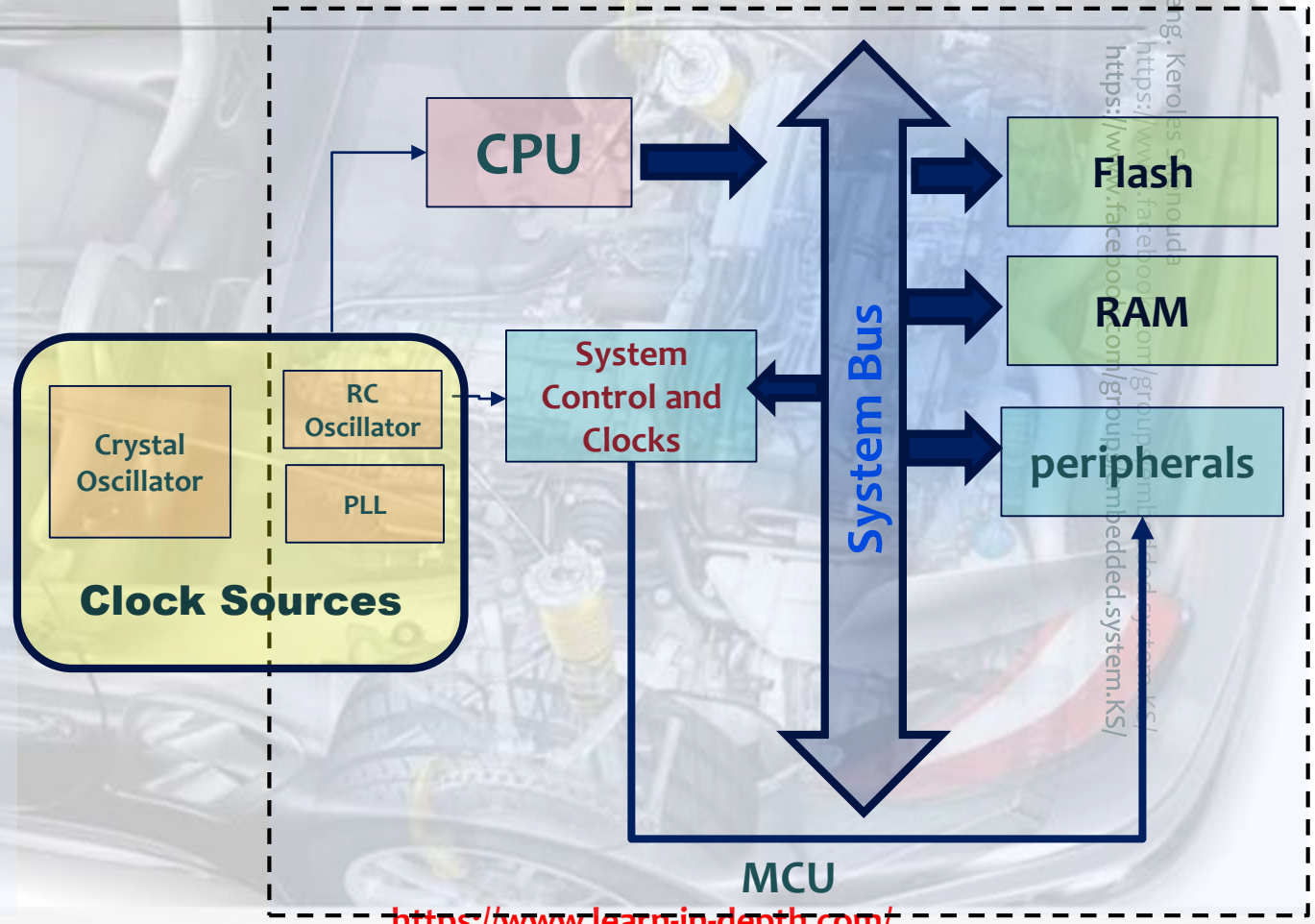


#LEARN_IN DEPTH

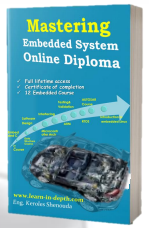
#Be_professional_in_embedded_system

Peripherals Clock Configuration

- By default, Peripherals Clocks of almost Peripherals will be disabled to save Power.

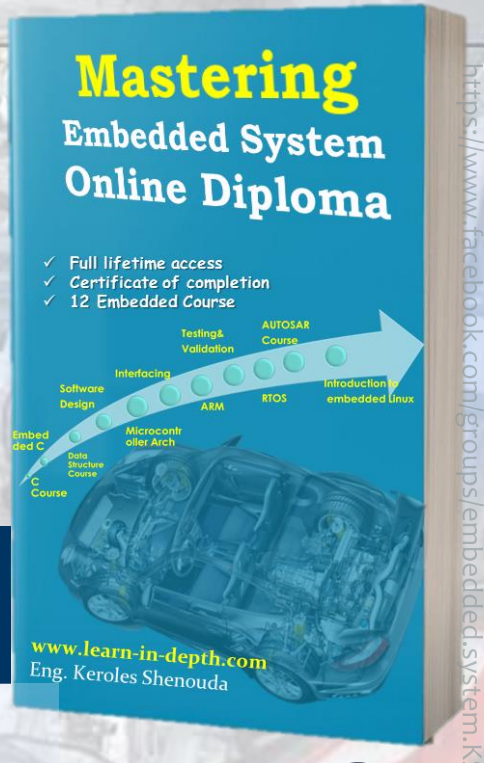


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system



LAB1: Practical lab on STM32F103XX

TOGGLE LED

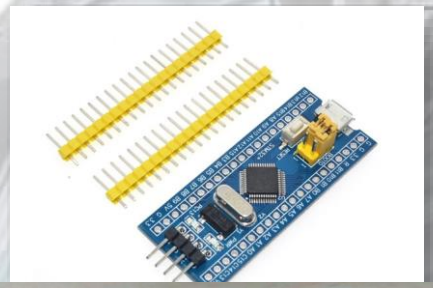
LEARN-IN-DEPTH
Be professional in embedded system

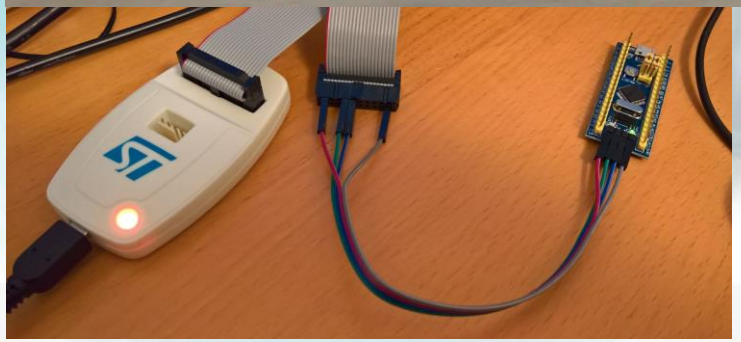
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

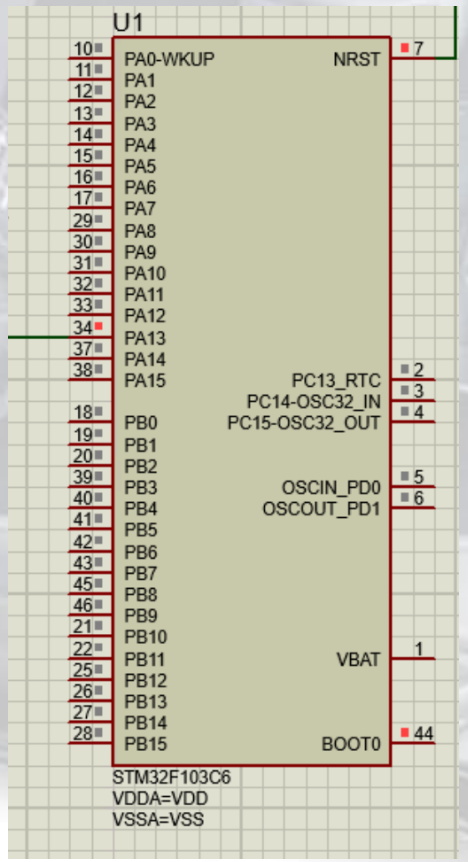
#LEARN_IN DEPTH

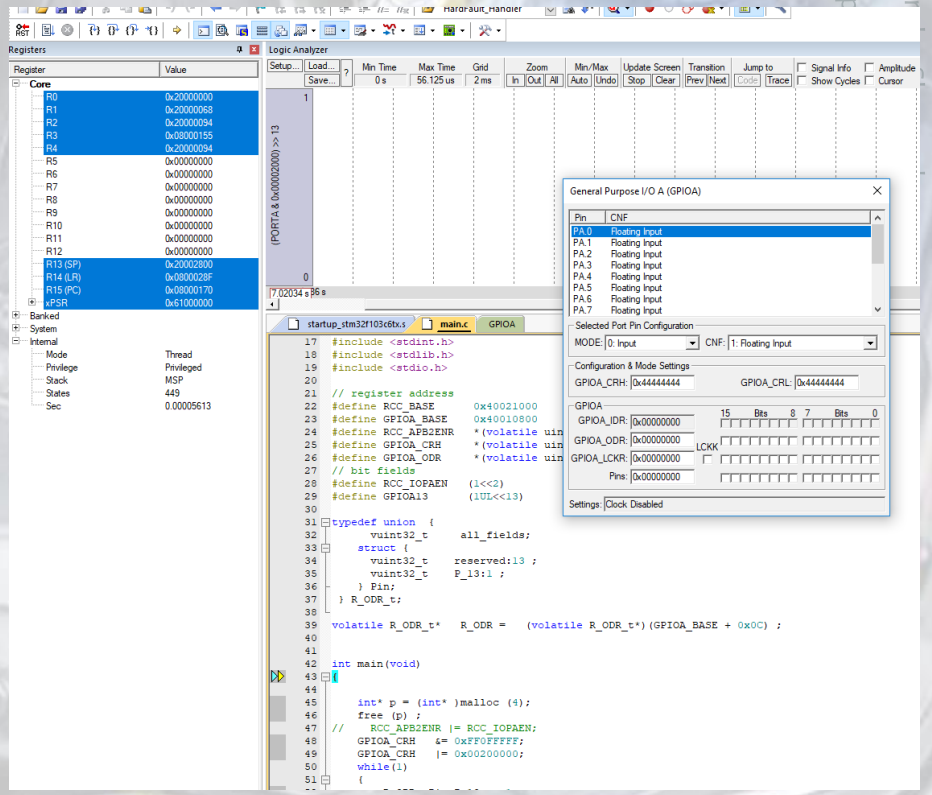
#Be_professional_in_embedded_system

ARM STM32 Minimum System Development Board "STM32F103C8T6"









https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

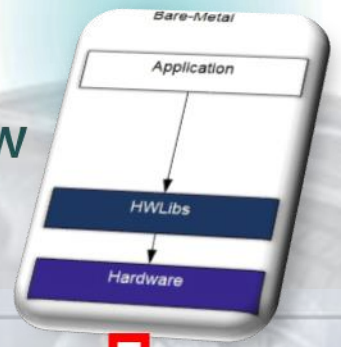
Toggle port A pin 13

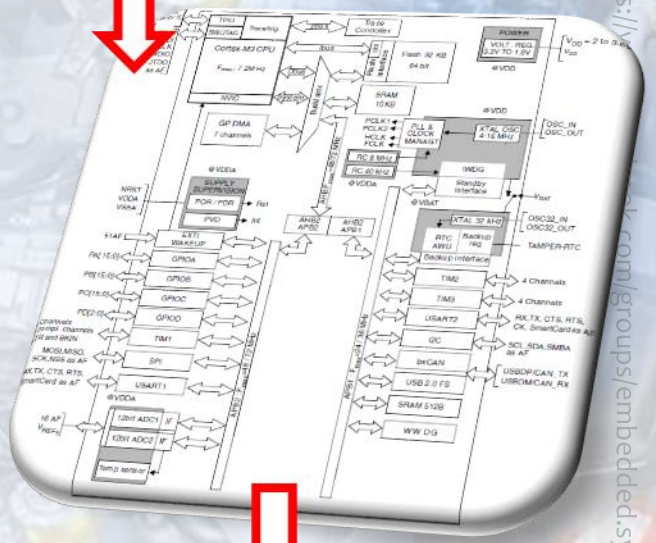
Run This Code …. ?

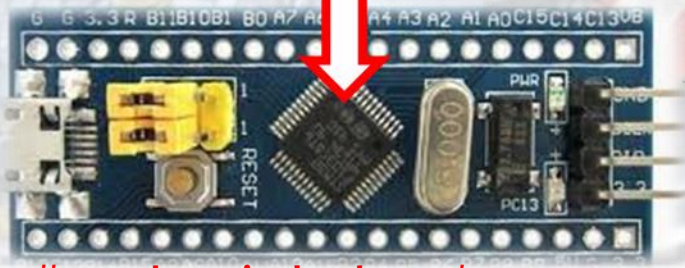
What is the expected … output ? ….why ?

```
//Learn-in-depth
//Keroles Shenouda
//Mastering_Embedded System online diploma
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

// register address
#define GPIOA_BASE    0x40010800
#define GPIOA_CRH     *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR     *(volatile uint32_t *)(GPIOA_BASE + 0x0C)

int main(void)
{
//Init GPIOA
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
    GPIOA_ODR |=  1<<13 ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}
```

Bare metal SW



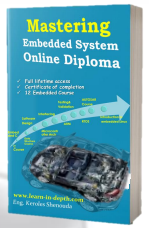




FOLLOW US

Press here

55



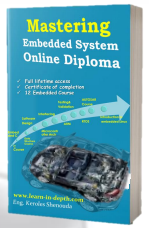
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.facebook.com/groups/embedded.system.KS/

Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



STM32103XX

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

MCU/MPU Filters

Part Number: STM32F103C6

Core
Series
Line
Package
Other
Peripheral

Features | Block Diagram | Docs & Resources | Datasheet | Buy

STM32F1 Series

STM32F103C6 LQFP48

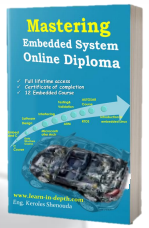
STM32F103C6Tx

MCUs/MPUs List: 2 items

Display similar items | Export

*	Part No	Reference	Marketing ...	Unit Price f...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F103...	STM32F103...	NA	NA		LQFP48	32 kBytes	10 kBytes	37	72 MHz
☆		STM32F103...	NA	NA		UFQFPN48	32 kBytes	10 kBytes	37	72 MHz

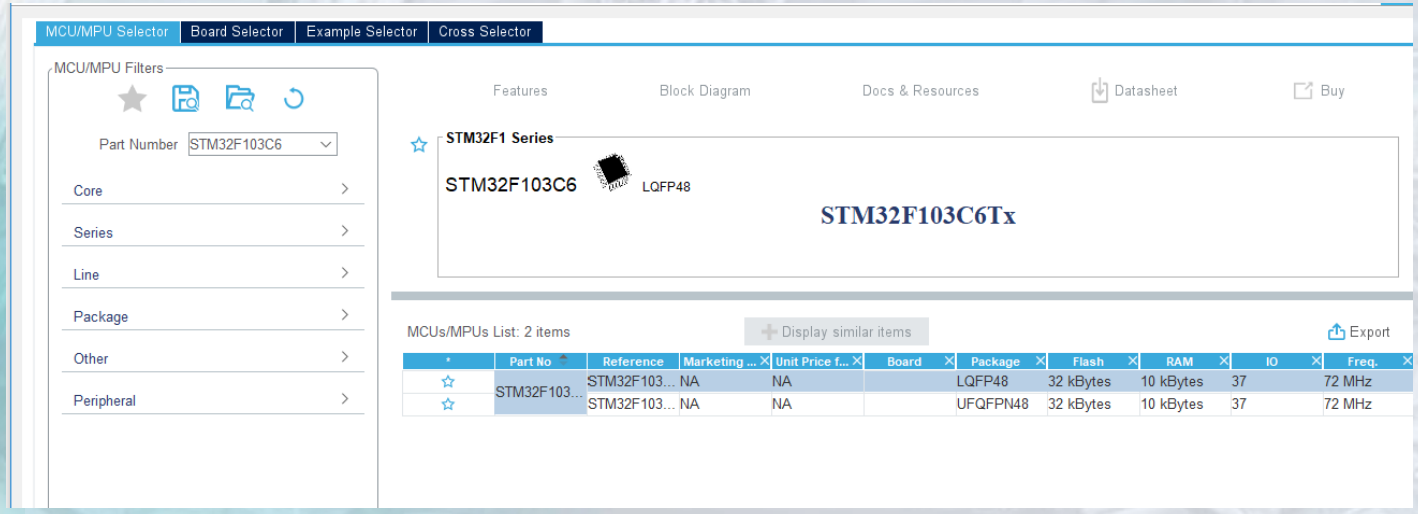
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

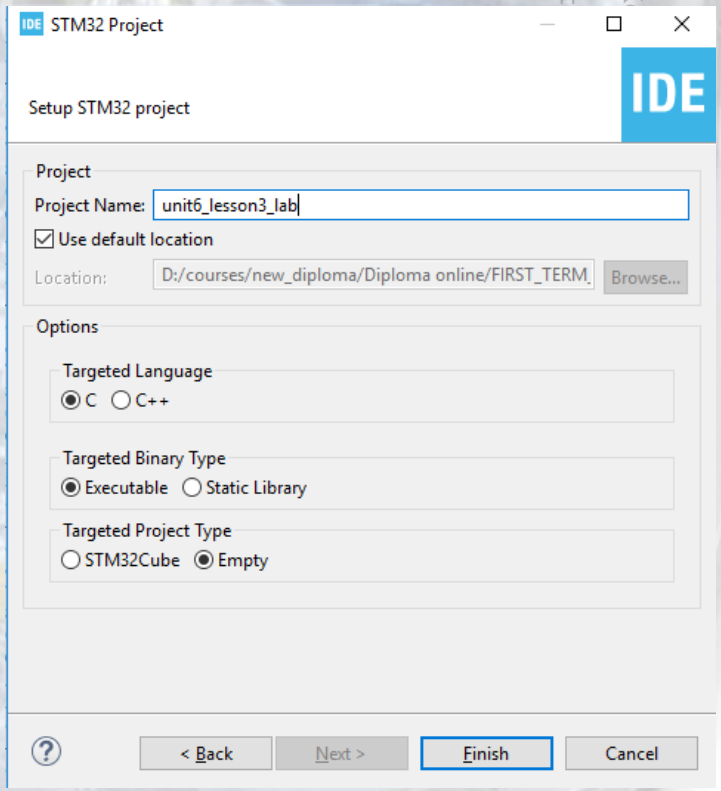


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

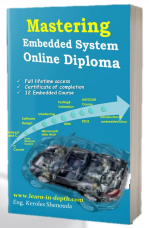
STM32103XX





https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

STM32103XX

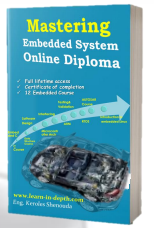


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

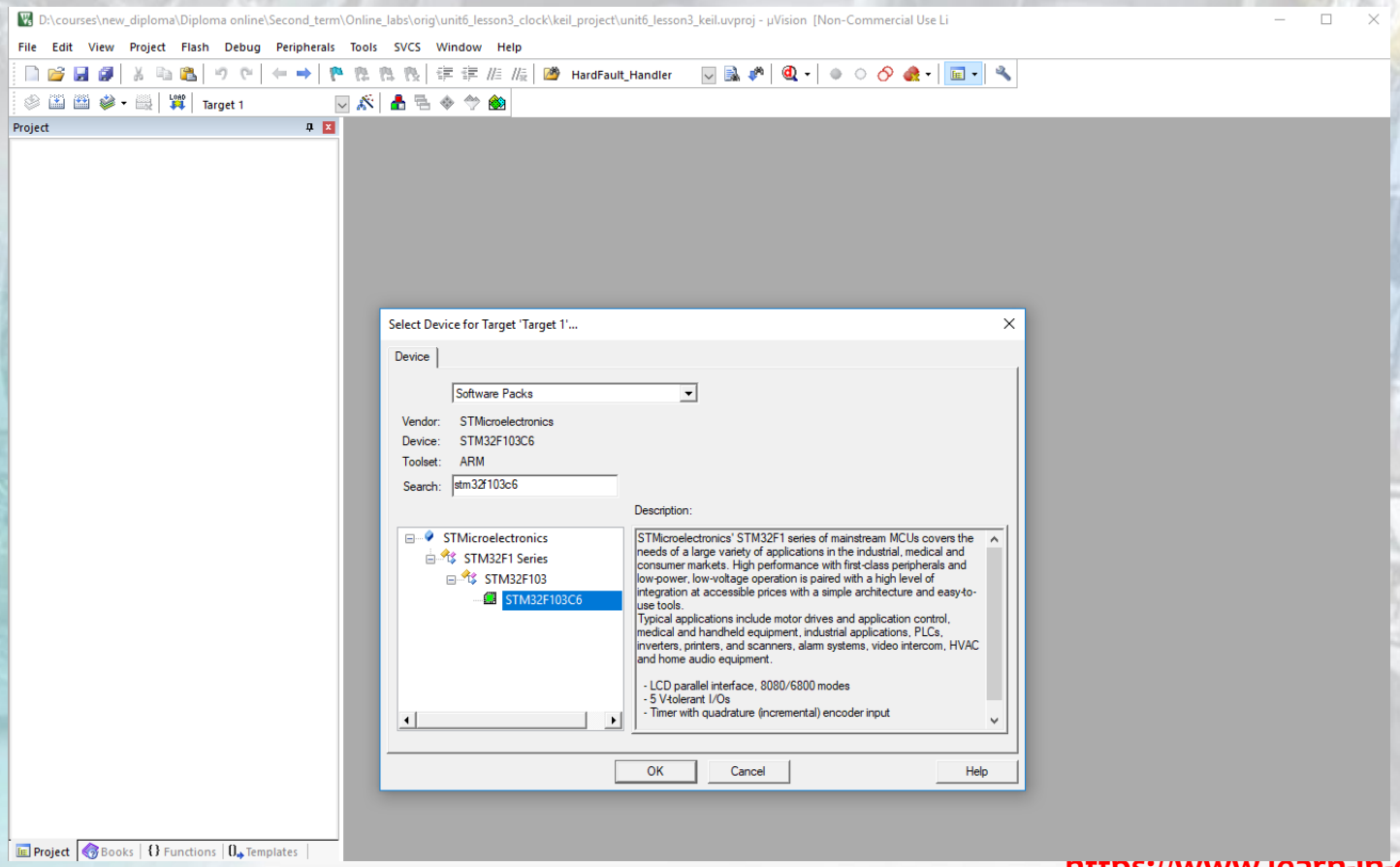
```
11
12  //Learn-in-depth
13  //Keroles Shenouda
14  //Mastering_Embedded System online diploma
15  typedef volatile unsigned int vuint32_t ;
16  #include <stdint.h>
17  #include <stdlib.h>
18  #include <stdio.h>
19
20  // register address
21  #define GPIOA_BASE    0x40010800
22  #define GPIOA_CRH     *(volatile uint32_t *)(GPIOA_BASE + 0x04)
23  #define GPIOA_ODR     *(volatile uint32_t *)(GPIOA_BASE + 0x0C)
24
25  int main(void)
26  {
27      //Init GPIOA
28      GPIOA_CRH   &= 0xFF0FFFFF;
29      GPIOA_CRH   |= 0x00200000;
30      while(1)
31      {
32          GPIOA_ODR |=  1<<13 ;
33          for (int i = 0; i < 5000; i++); // arbitrary delay
34          GPIOA_ODR  &=  ~(1<<13) ;
35          for (int i = 0; i < 5000; i++); // arbitrary delay
36      }
37  }
38
```


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



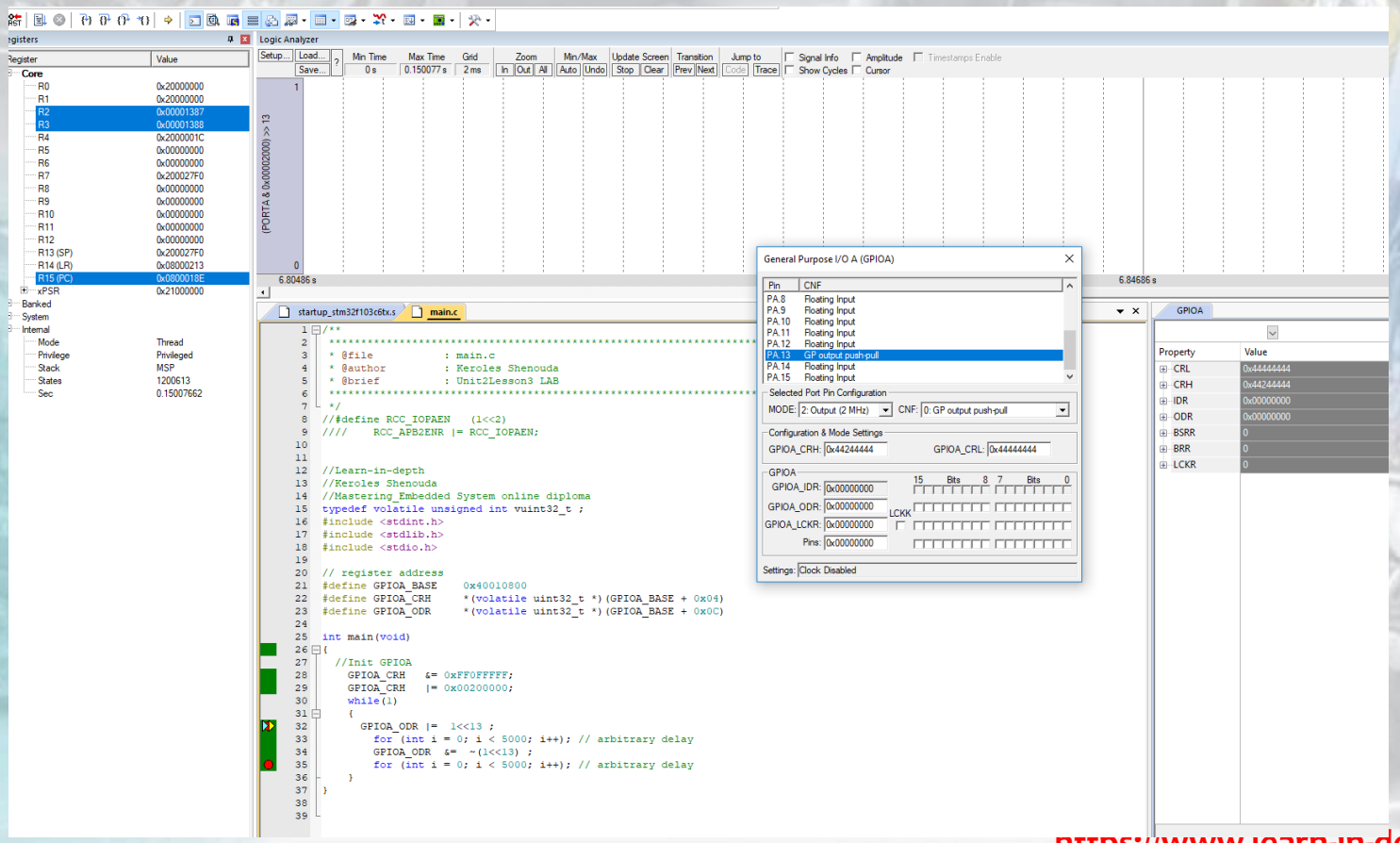
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

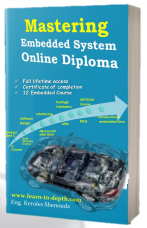


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

No Output as No Clock for the gpioa Module



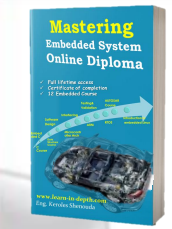

```c
/**
 ******************************************************************************
 * @file           : main.c
 * @author         : Keroles Shenouda
 * @brief          : Unit2Lesson3 LAB
 ******************************************************************************
 */
//#define RCC_IOPAEN    (1<<2)
////     RCC_APB2ENR |= RCC_IOPAEN;


//Learn-in-depth
//Keroles Shenouda
//Mastering_Embedded System online diploma
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

// register address
#define GPIOA_BASE     0x40010800
#define GPIOA_CRH      *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR      *(volatile uint32_t *)(GPIOA_BASE + 0x0C)

int main(void)
{
  //Init GPIOA
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
      GPIOA_ODR |=  1<<13 ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}
```

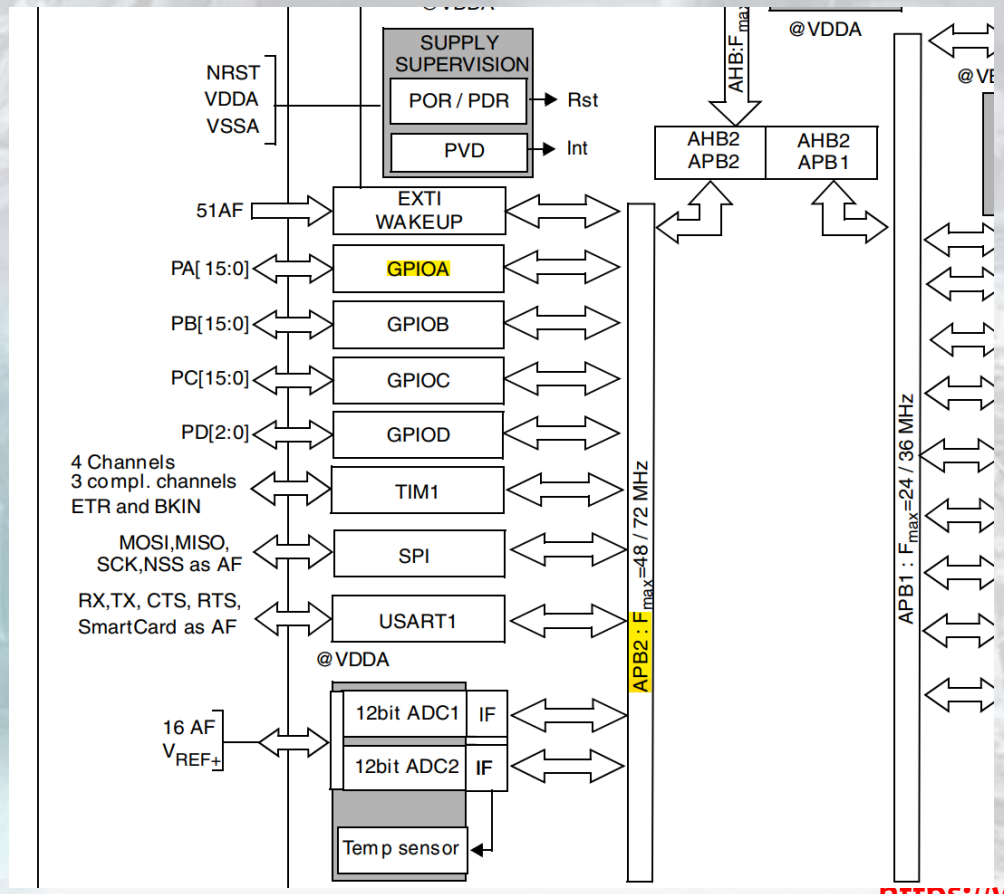
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

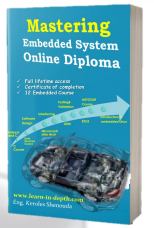
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



So we need to enable Clock



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



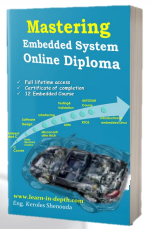
So we need to enable Clock

```
24
25 #define RCC_BASE     0x40021000
26
```

Table 3. Register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC	AHB	*Section 21.6.9 on page 564*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*Section 28.16.6 on page 913*
0x4003 0000 - 0x4FFF FFFF	Reserved		-
0x4002 8000 - 0x4002 9FFF	Ethernet		*Section 29.8.5 on page 1069*
0x4002 3400 - 0x4002 7FFF	Reserved		-
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4 on page 65*
0x4002 2000 - 0x4002 23FF	Flash memory interface		-
0x4002 1400 - 0x4002 1FFF	Reserved		-
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		*Section 7.3.11 on page 121*
0x4002 0800 - 0x4002 0FFF	Reserved		-

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



So we need to enable Clock

```
25 #define RCC_BASE     0x40021000
26 #define RCC_APB2ENR      *(volatile uint32_t *)(GPIOA_BASE + 0x18)
27
```

7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

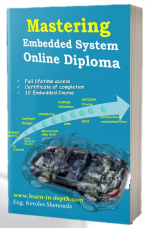
Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: *When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.*



So we need to enable Clock

Bit 2 **IOPAEN:** IO port A clock enable

Set and cleared by software.

0: IO port A clock disabled

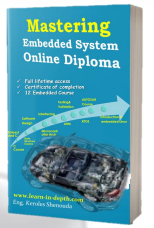
1: IO port A clock enabled

```
29  {
30      //Enable IO port A clock
31      RCC_APB2ENR |= 1<<2 ;
32      //Init GPIOA
```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

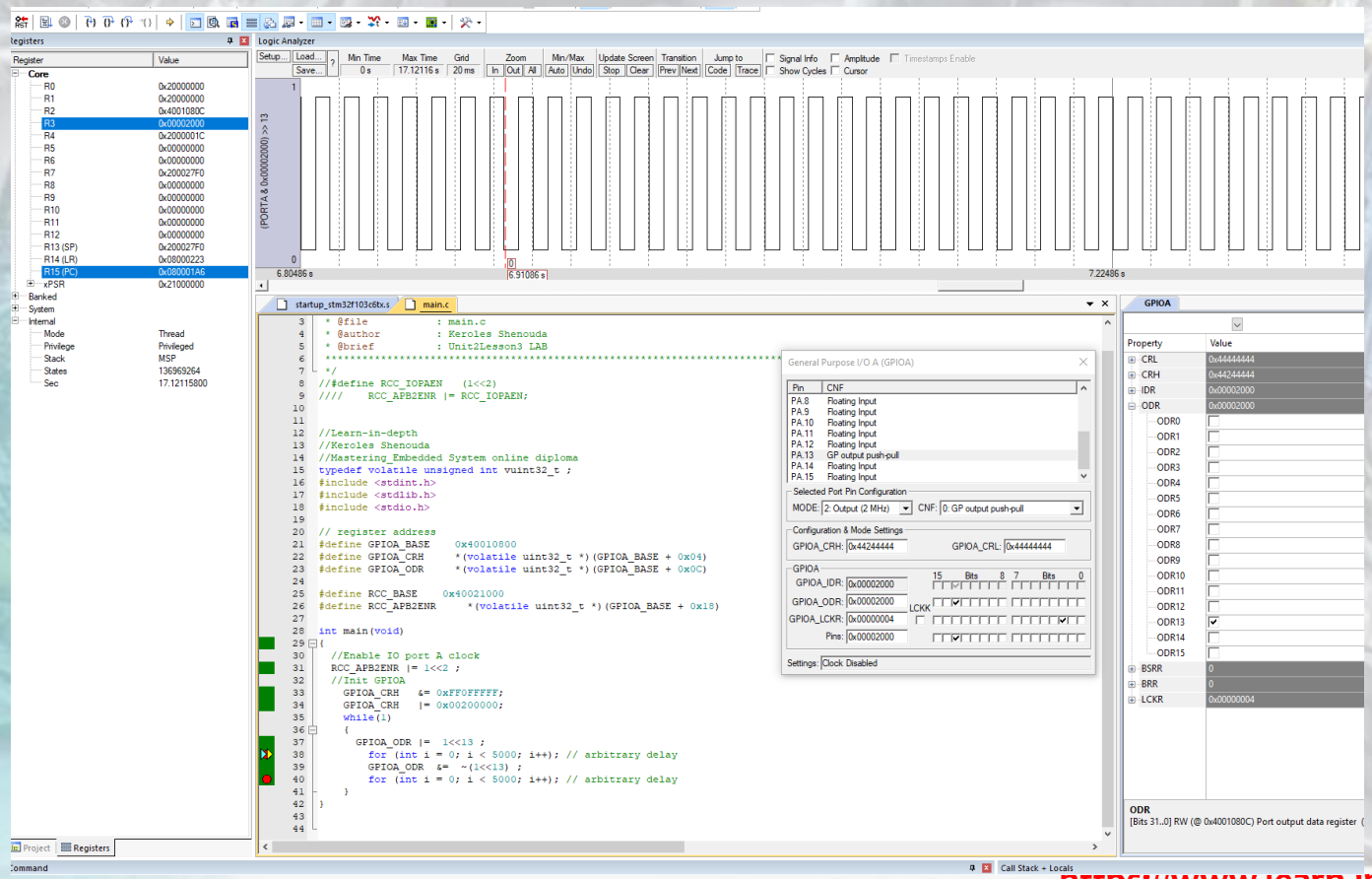


So we need to enable Clock

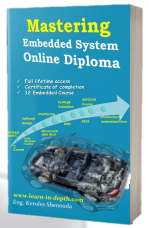
```c
//Learn-in-depth
//Keroles Shenouda
//Mastering_Embedded System online diploma
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

// register address
#define GPIOA_BASE    0x40010800
#define GPIOA_CRH     *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR     *(volatile uint32_t *)(GPIOA_BASE + 0x0C)

#define RCC_BASE    0x40021000
#define RCC_APB2ENR     *(volatile uint32_t *)(GPIOA_BASE + 0x18)

int main(void)
{
    //Enable IO port A clock
    RCC_APB2ENR |= 1<<2 ;
    //Init GPIOA
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
        GPIOA_ODR |=  1<<13 ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}
```


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

We Can See the output now



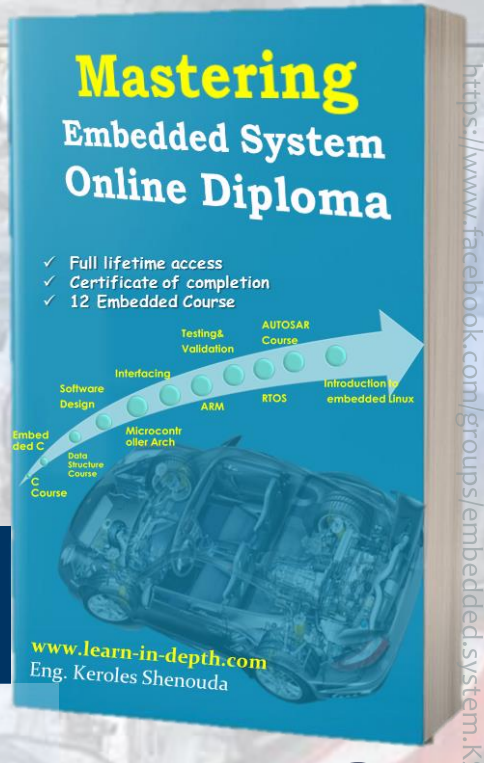
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

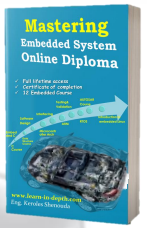
#Be_professional_in_embedded_system

Lab2: Change SYSCLK, HCLK, PCLK1 and PCLK2 wit different frequencies



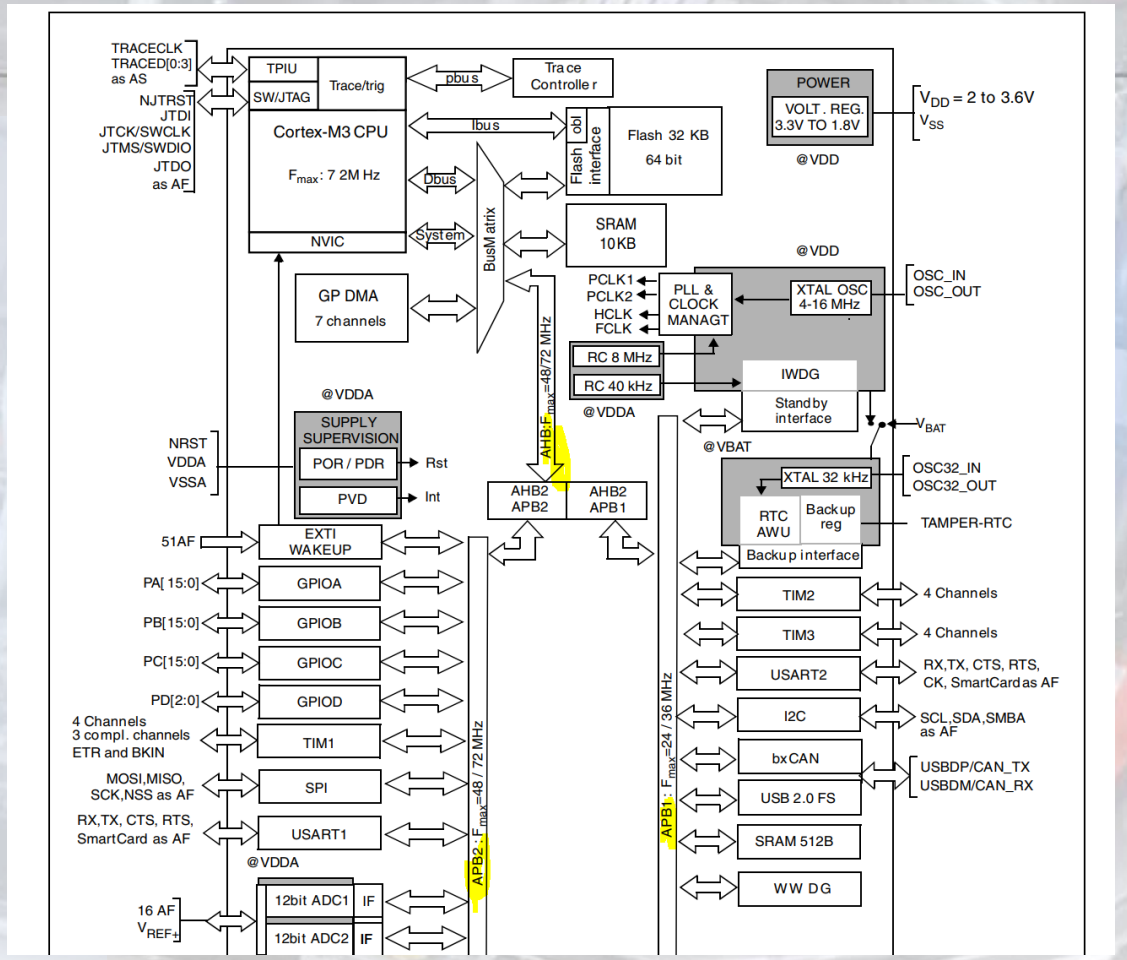
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



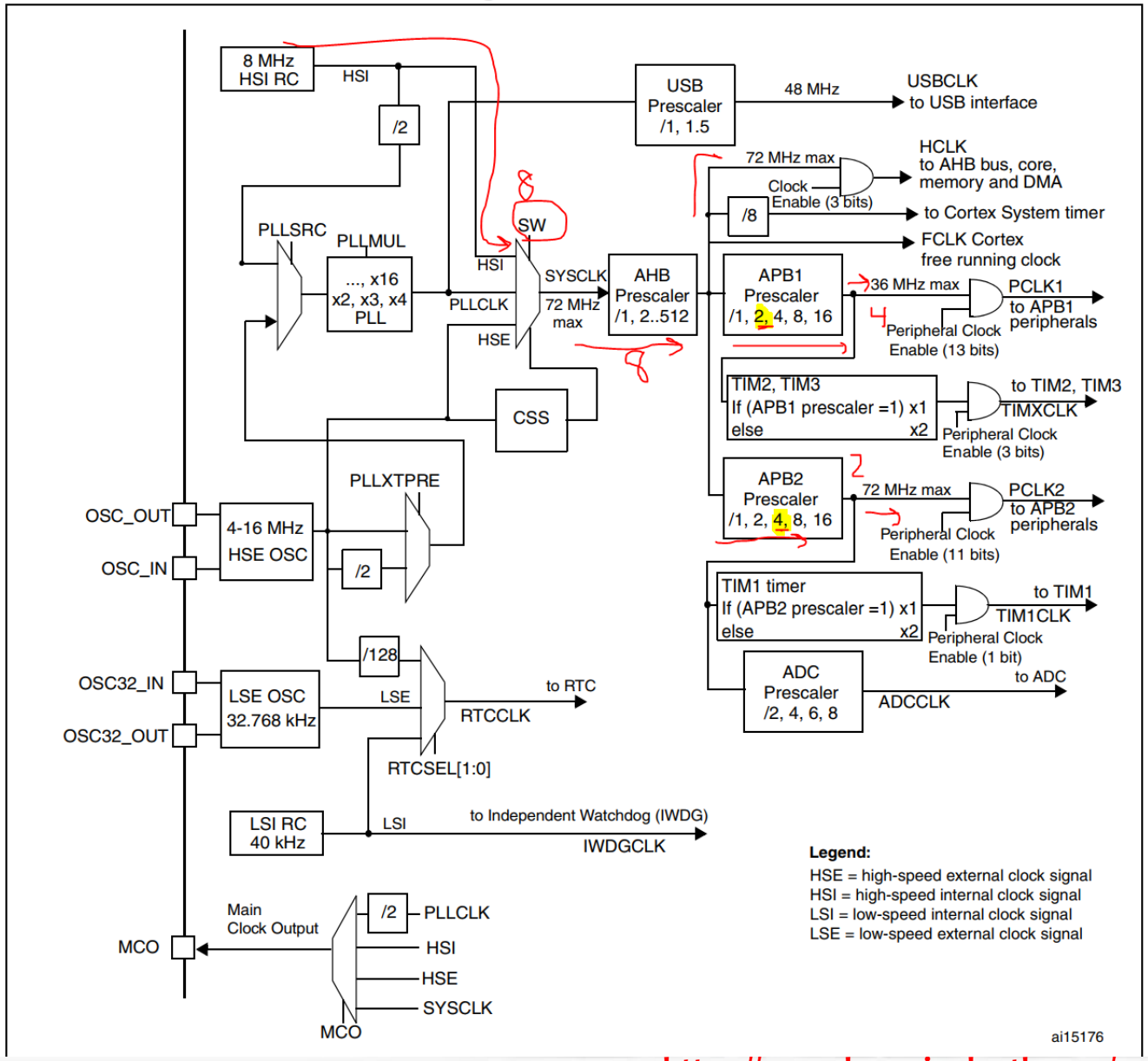
LAB2

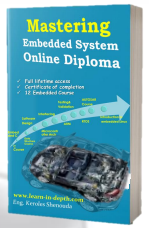
- Configure Board to run with the Following rates:
  - APB1 Bus frequency 4MHZ
  - APB2 Bus frequency 2MHZ
  - AHB frequency 8 MHZ
  - SysClk 8 MHZ
  - Use only internal HSI_RC



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Lab2 Clock Tree Design

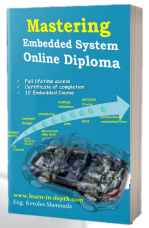




#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



7.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

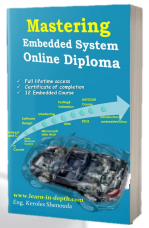
Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw



Bits 13:11 **PPRE2:** APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB high-speed clock (PCLK2).

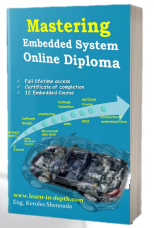
0xx: HCLK not divided

100: HCLK divided by 2

101: HCLK divided by 4

110: HCLK divided by 8

111: HCLK divided by 16



Bits 10:8 **PPRE1:** APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).
Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 7:4 **HPRE:** AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.
0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

Note: The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock. Refer to Reading the Flash memory section for more details.

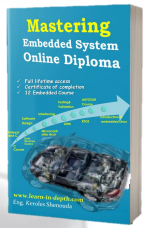
Bits 3:2 **SWS:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

Bits 1:0 **SW:** System clock switch

Set and cleared by software to select SYSCLK source.
Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).
00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

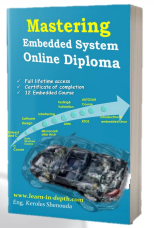
```c
 ******************************************************************************
 */
//#define RCC_IOPAEN   (1<<2)
////    RCC_APB2ENR |= RCC_IOPAEN;


//Learn-in-depth
//Keroles Shenouda
//Mastering_Embedded System online diploma
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

#define RCC_BASE    0x40021000
#define RCC_CFGR     *(volatile uint32_t *)(RCC_BASE + 0x04)

void clock_init()
{


//  Bits 10:8 PPRE1: APB low-speed prescaler (APB1)
//  Set and cleared by software to control the division factor of the APB low-speed clock
//  (PCLK1).
//  Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.
//  0xx: HCLK not divided
//  100: HCLK divided by 2
//  101: HCLK divided by 4
//  110: HCLK divided by 8
//  111: HCLK divided by 16
    RCC_CFGR |= (0b100 << 8);

//  Bits 13:11 PPRE2: APB high-speed prescaler (APB2)
//  Set and cleared by software to control the division factor of the APB high-speed clock
//  (PCLK2).
//  0xx: HCLK not divided
//  100: HCLK divided by 2
//  101: HCLK divided by 4
//  110: HCLK divided by 8
//  111: HCLK divided by 16

    RCC_CFGR |= (0b101 << 11);

}

int main(void)
{
    clock_init();
    while(1)
    {

    }
}
```



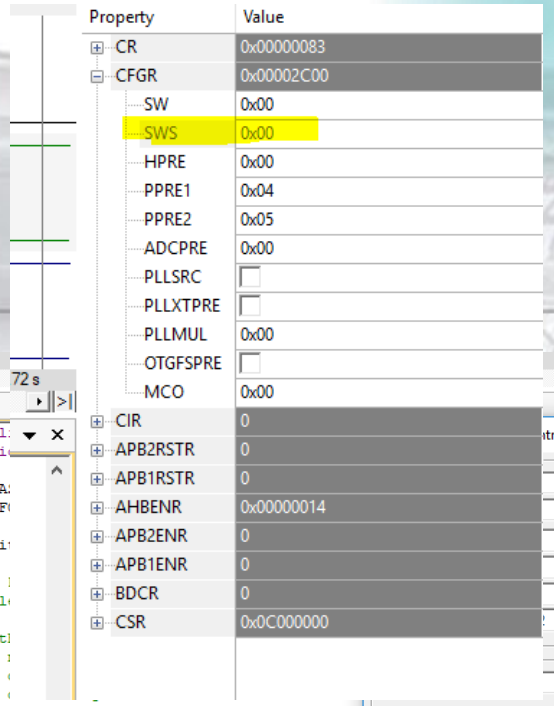
#LEARN_IN DEPTH

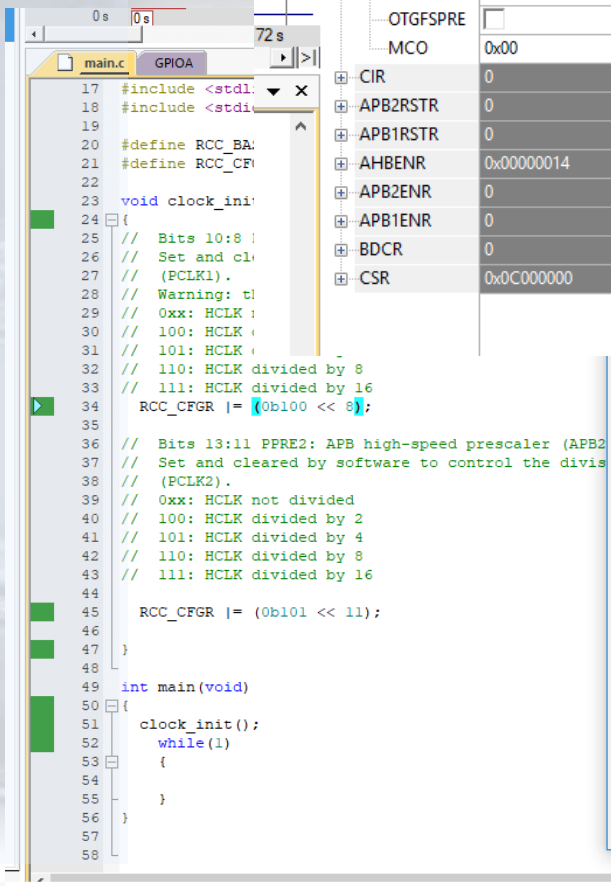
#Be_professional_in_embedded_system

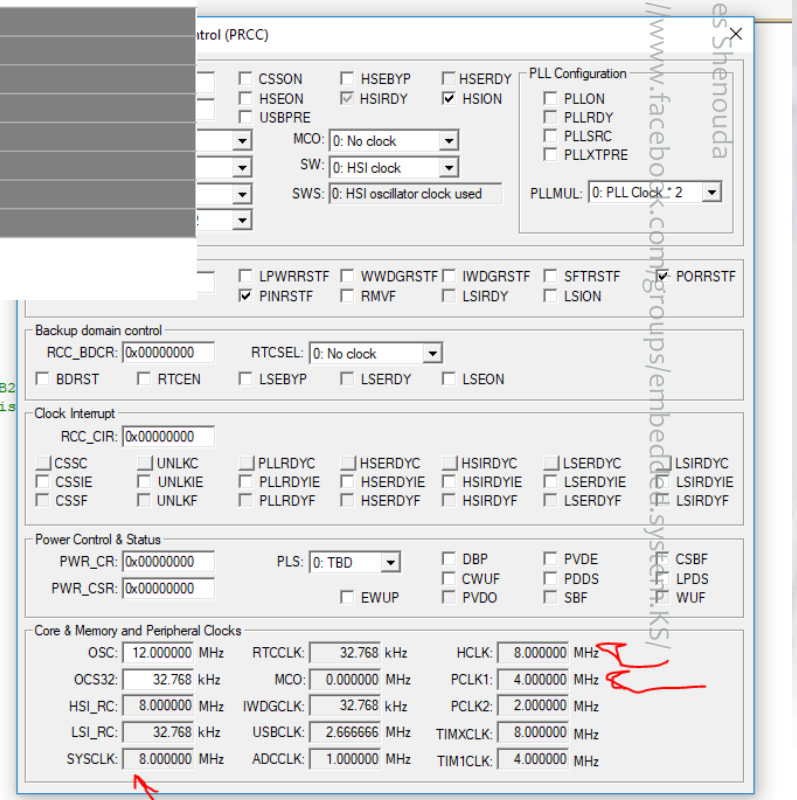
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



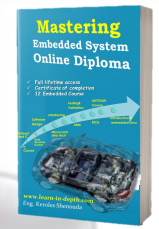
- Configure Board to run with the Following rates:
  - APB1 Bus frequency 4MHZ
  - APB2 Bus frequency 2MHZ
  - AHB frequency 8 MHZ
  - SysClk 8 MHZ
  - Use only internal HSI_RC



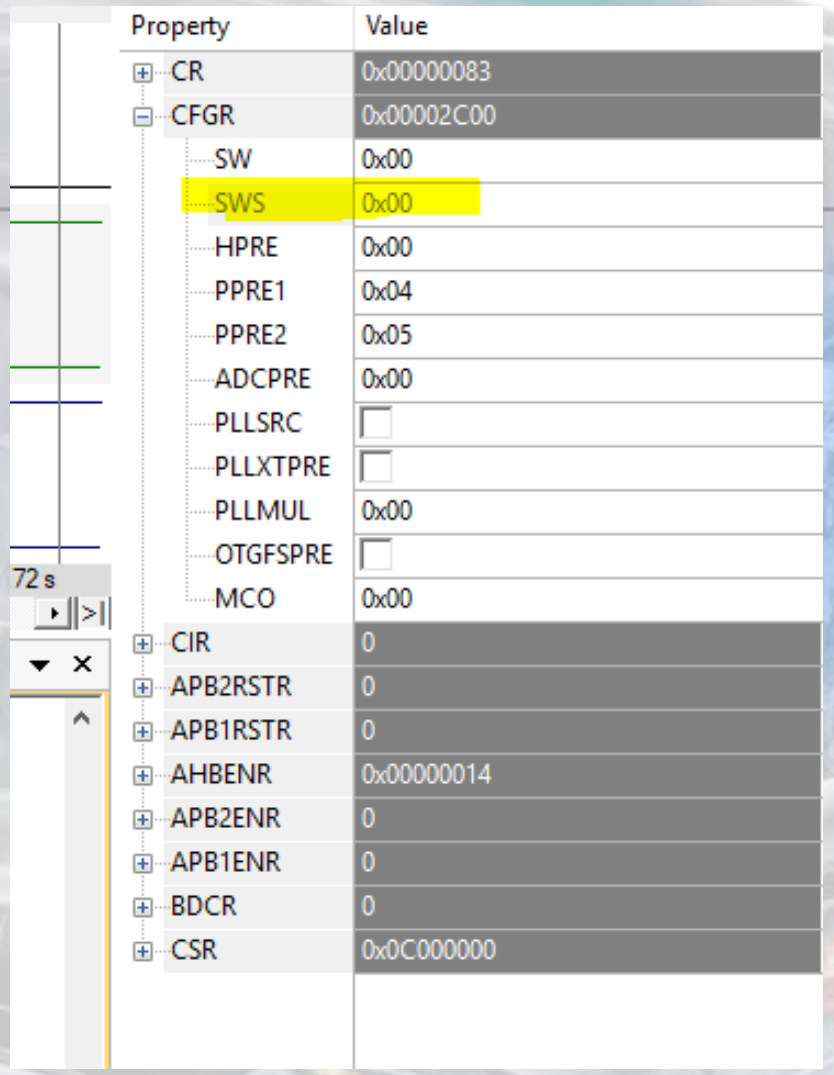




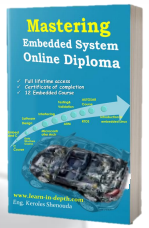
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



- Configure Board to run with the Following rates:
  - APB1 Bus frequency 4MHZ
  - APB2 Bus frequency 2MHZ
  - AHB frequency 8 MHZ
  - SysClk 8 MHZ
  - Use only internal HSI_RC

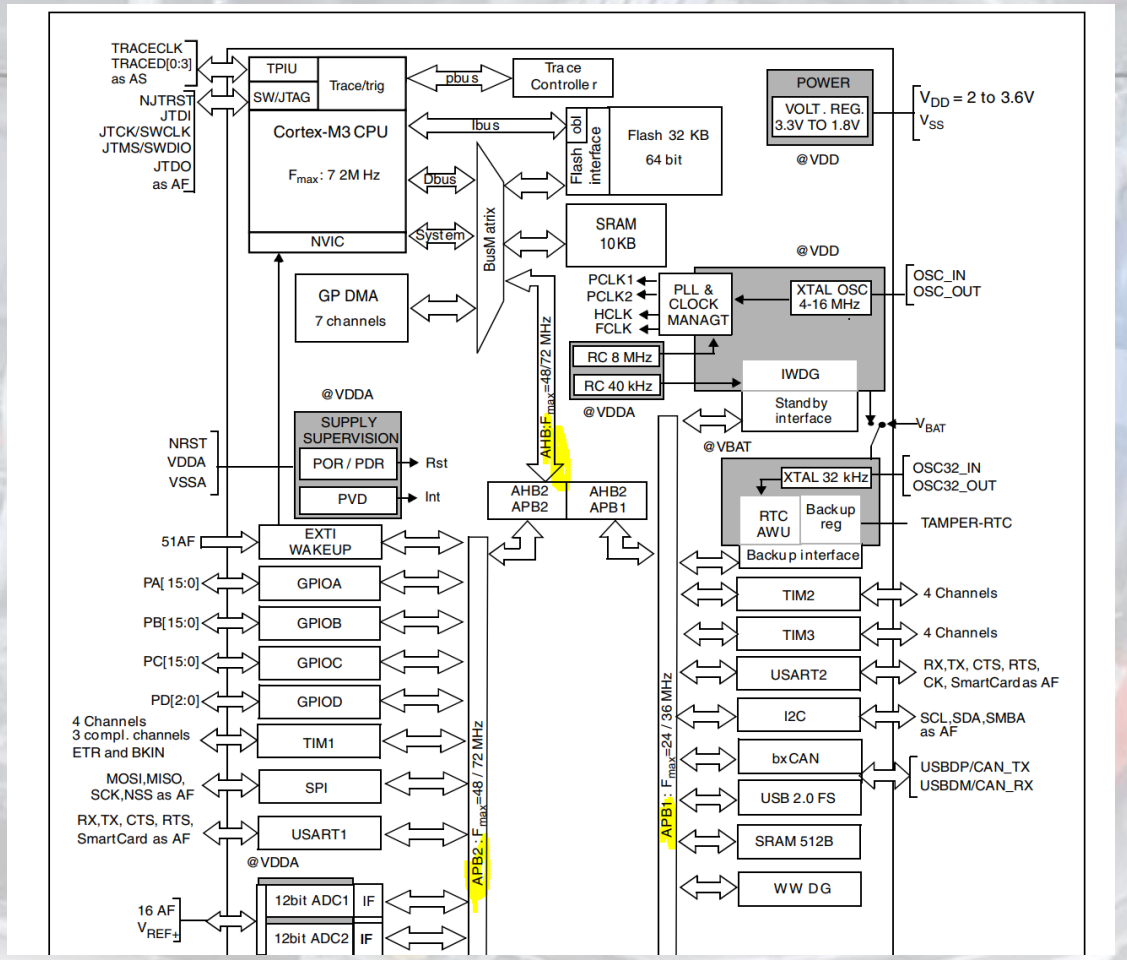


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



LAB2

- Configure Board to run with the Following rates:
  - APB1 Bus frequency 16MHZ
  - APB2 Bus frequency 8MHZ
  - AHB frequency 32 MHZ
  - SysClk 32 MHZ
  - Use only internal HSI_RC



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

LAB2

```
//  Bits 21:18 PLLMUL: PLL multiplication factor
//  These bits are written by software to define the PLL multiplication factor. These bits can be
//  written only when PLL is disabled.
//  Caution: The PLL output frequency must not exceed 72 MHz.
//  0000: PLL input clock x 2
//  0001: PLL input clock x 3
//  0010: PLL input clock x 4
//  0011: PLL input clock x 5
//  0100: PLL input clock x 6
//  0101: PLL input clock x 7
//  0110: PLL input clock x 8
    RCC_CFGR |= (0b0110 << 18);
```

Bits 21:18 **PLLMUL:** PLL multiplication factor

These bits are written by software to define the PLL multiplication factor. These bits can be written only when PLL is disabled.

Caution: The PLL output frequency must not exceed 72 MHz.

0000: PLL input clock x 2
0001: PLL input clock x 3
0010: PLL input clock x 4
0011: PLL input clock x 5
0100: PLL input clock x 6
0101: PLL input clock x 7
0110: PLL input clock x 8
0111: PLL input clock x 9
1000: PLL input clock x 10
1001: PLL input clock x 11
1010: PLL input clock x 12
1011: PLL input clock x 13
1100: PLL input clock x 14
1101: PLL input clock x 15
1110: PLL input clock x 16
1111: PLL input clock x 16

Bit 17 **PLLXTPRE:** HSE divider for PLL entry

Set and cleared by software to divide HSE before PLL entry. This bit can be written only when PLL is disabled.

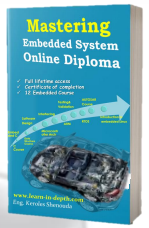
0: HSE clock not divided
1: HSE clock divided by 2

Bit 16 **PLLSRC:** PLL entry clock source

Set and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled.

0: HSI oscillator clock / 2 selected as PLL input clock
1: HSE oscillator clock selected as PLL input clock

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Lab2

```
//Bits 1:0 SW: System clock switch
//Set and cleared by software to select SYSCLK source.
//Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of
//failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security
//System is enabled).
//00: HSI selected as system clock
//01: HSE selected as system clock
//10: PLL selected as system clock
//11: not allowed
RCC_CFGR |= (0b10 << 0);
```

Bits 1:0 **SW:** System clock switch

Set and cleared by software to select SYSCLK source.

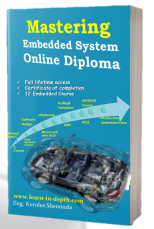
Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).

00: HSI selected as system clock

01: HSE selected as system clock

10: PLL selected as system clock

11: not allowed



LAB2 Status Should be 10 at run time (is written by HW)

Bits 3:2 **SWS:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

```
26  //  Bits 3:2 SWS: System clock switch status
27  //  Set and cleared by hardware to indicate which clock source is used as system clock.
28  //  00: HSI oscillator used as system clock
29  //  01: HSE oscillator used as system clock
30  //  10: PLL used as system clock
31  //  11: not applicable
32      RCC_CFGR |= (0b10 << 2);
```


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

LAB2

```
19
20 #define RCC_BASE    0x40021000
21 #define RCC_CFGR     *(volatile uint32_t *)(RCC_BASE + 0x04)
22 #define RCC_CR      *(volatile uint32_t *)(RCC_BASE + 0x0)
23
24 void clock_init()
25 {
26 //  Bit 24 PLLON: PLL enable
27 //  Set and cleared by software to enable PLL.
28 //  Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the
29 //  PLL clock is used as system clock or is selected to become the system clock.
30 //  0: PLL OFF
31 //  1: PLL ON
32     RCC_CR |= (1 << 24);
33
```

7.3.1 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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						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

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** PLL clock ready flag
Set by hardware to indicate that the PLL is locked.
0: PLL unlocked
1: PLL locked

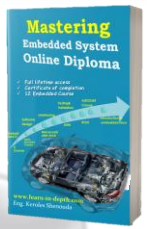
Bit 24 **PLLON:** PLL enable
Set and cleared by software to enable PLL.
Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock.
0: PLL OFF
1: PLL ON

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Lab2

```
24  void clock_init()
25  {
26
27
28
29    //Bits 1:0 SW: System clock switch
30    //Set and cleared by software to select SYSCLK source.
31    //Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of
32    //failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security
33    //System is enabled).
34    //00: HSI selected as system clock
35    //01: HSE selected as system clock
36    //10: PLL selected as system clock
37    //11: not allowed
38    RCC_CFGR |= (0b10 << 0);
39
40    //  Bits 21:18 PLLMUL: PLL multiplication factor
41    //  These bits are written by software to define the PLL multiplication factor. These bits can be
42    //  written only when PLL is disabled.
43    //  Caution: The PLL output frequency must not exceed 72 MHz.
44    //  0000: PLL input clock x 2
45    //  0001: PLL input clock x 3
46    //  0010: PLL input clock x 4
47    //  0011: PLL input clock x 5
48    //  0100: PLL input clock x 6
49    //  0101: PLL input clock x 7
50    //  0110: PLL input clock x 8
51    RCC_CFGR |= (0b0110 << 18);
52
53    //  Bits 10:8 PPRE1: APB low-speed prescaler (APB1)
54    //  Set and cleared by software to control the division factor of the APB low-speed clock
55    //  (PCLK1).
56    //  Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.
57    //  0xx: HCLK not divided
58    //  100: HCLK divided by 2
59    //  101: HCLK divided by 4
60    //  110: HCLK divided by 8
61    //  111: HCLK divided by 16
62    RCC_CFGR |= (0b100 << 8);
63
64    //  Bits 13:11 PPRE2: APB high-speed prescaler (APB2)
65    //  Set and cleared by software to control the division factor of the APB high-speed clock
66    //  (PCLK2).
67    //  0xx: HCLK not divided
68    //  100: HCLK divided by 2
69    //  101: HCLK divided by 4
70    //  110: HCLK divided by 8
71    //  111: HCLK divided by 16
72
73    RCC_CFGR |= (0b101 << 11);
74
75
76    //  Bit 24 PLLON: PLL enable
77    //  Set and cleared by software to enable PLL.
78    //  Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the
79    //  PLL clock is used as system clock or is selected to become the system clock.
80    //  0: PLL OFF
81    //  1: PLL ON
82    RCC_CR |= (1 << 24);
83  }
```


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Simulation

- Configure Board to run with the Following rates:
  - APB1 Bus frequency 16MHZ
  - APB2 Bus frequency 16MHZ
  - AHB frequency 32 MHZ
  - SysClk 32 MHZ
  - Use only internal HSI_RC

main.c | startup_stm32f103c6tx.s | GPIOA

```
53  //  Bits 10:8 PPRE1: APB low-speed prescaler
54  //  Set and cleared by software to control t
55  //  (PCLK1).
56  //  Warning: the software has to set correct
57  //  0xx: HCLK not divided
58  //  100: HCLK divided by 2
59  //  101: HCLK divided by 4
60  //  110: HCLK divided by 8
61  //  111: HCLK divided by 16
62  RCC_CFGR |= (0b100 << 8);
63
64  //  Bits 13:11 PPRE2: APB high-speed prescal
65  //  Set and cleared by software to control t
66  //  (PCLK2).
67  //  0xx: HCLK not divided
68  //  100: HCLK divided by 2
69  //  101: HCLK divided by 4
70  //  110: HCLK divided by 8
71  //  111: HCLK divided by 16
72
73  RCC_CFGR |= (0b101 << 11);
74
75
76  //  Bit 24 PLLON: PLL enable
77  //  Set and cleared by software to enable PL
78  //  Cleared by hardware when entering Stop o
79  //  PLL clock is used as system clock or is
80  //  0: PLL OFF
81  //  1: PLL ON
82  RCC_CR |= (1 << 24);
83 }
84
85 int main(void)
86 {
87  clock_init();
88  while(1)
89  {
90
91  }
92 }
93
94
```

Power, Reset and Clock Control (PRCC)

Clock Control & Configuration
RCC_CR: 0x03000083 — CSSON, HSEBYP, HSERDY, HSEON, HSIRDY, HSION, USBPRE
RCC_CFGR: 0x00182C0A
PPRE1: 4: HCLK / 2 — MCO: 0: No clock
PPRE2: 5: HCLK / 4 — SW: 2: PLL clock
HPRE: 0: SYSCLK — SWS: 2: PLL clock used
ADCPRE: 0: PCLK2 / 2
PLL Configuration: PLLON, PLLRDY, PLLSRC, PLLXTPRE, PLLMUL: 6: PLL Clock * 8

Control/Status
RCC_CSR: 0x0C000000 — LPWRRSTF, WWDGRSTF, IWDGRSTF, SFTRSTF, PORRSTF, PINRSTF, RMVF, LSIRDY, LSION

Backup domain control
RCC_BDCR: 0x00000000 — RTCSEL: 0: No clock
BDRST, RTCEN, LSEBYP, LSERDY, LSEON

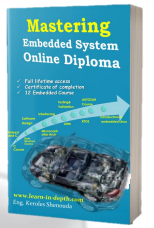
Clock Interrupt
RCC_CIR: 0x00000000
CSSC, UNLKC, PLLRDYC, HSERDYC, HSIRDYC, LSERDYC, LSIRDYC
CSSIE, UNLKIE, PLLRDYIE, HSERDYIE, HSIRDYIE, LSERDYIE, LSIRDYIE
CSSF, UNLKF, PLLRDYF, HSERDYF, HSIRDYF, LSERDYF, LSIRDYF

Power Control & Status
PWR_CR: 0x00000000 — PLS: 0: TBD
PWR_CSR: 0x00000000
DBP, PVDE, CSBF, CWUF, PDDS, LPDS, EWUP, PVDO, SBF, WUF

Core & Memory and Peripheral Clocks

OSC:	12.000000 MHz	RTCCLK:	32.768 kHz	HCLK:	32.000000 MHz
OCS32:	32.768 kHz	MCO:	0.000000 MHz	PCLK1:	16.000000 MHz
HSI_RC:	8.000000 MHz	IWDGCLK:	32.768 kHz	PCLK2:	8.000000 MHz
LSI_RC:	32.768 kHz	USBCLK:	21.333333 MHz	TIMXCLK:	32.000000 MHz
SYSCLK:	32.000000 MHz	ADCCLK:	4.000000 MHz	TIM1CLK:	16.000000 MHz

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



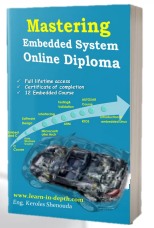
#LEARN_IN DEPTH

#Be_professional_in embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/