An Introduction to NEW* STM32/ARM On Arduino IDE

Focusing on Blue Pill F103C8

Using: STM32F1xx and STM32 Cores by ST-Microelectronics - New July 2017
(integrates CMSIS/STM32 HAL and STM32 Register definitions into Arduino IDE)

** Surgeon General Warning: May be hazardous to your health due to bugs*

STM32/ARM Programming Complexity Levels

- Entry Level STM32/ARM Programming (today's presentation)

  Arduino IDE using Sketch examples for STM32/ARM processors
- Intermediate Level STM32/ARM Programming (today's presentation)

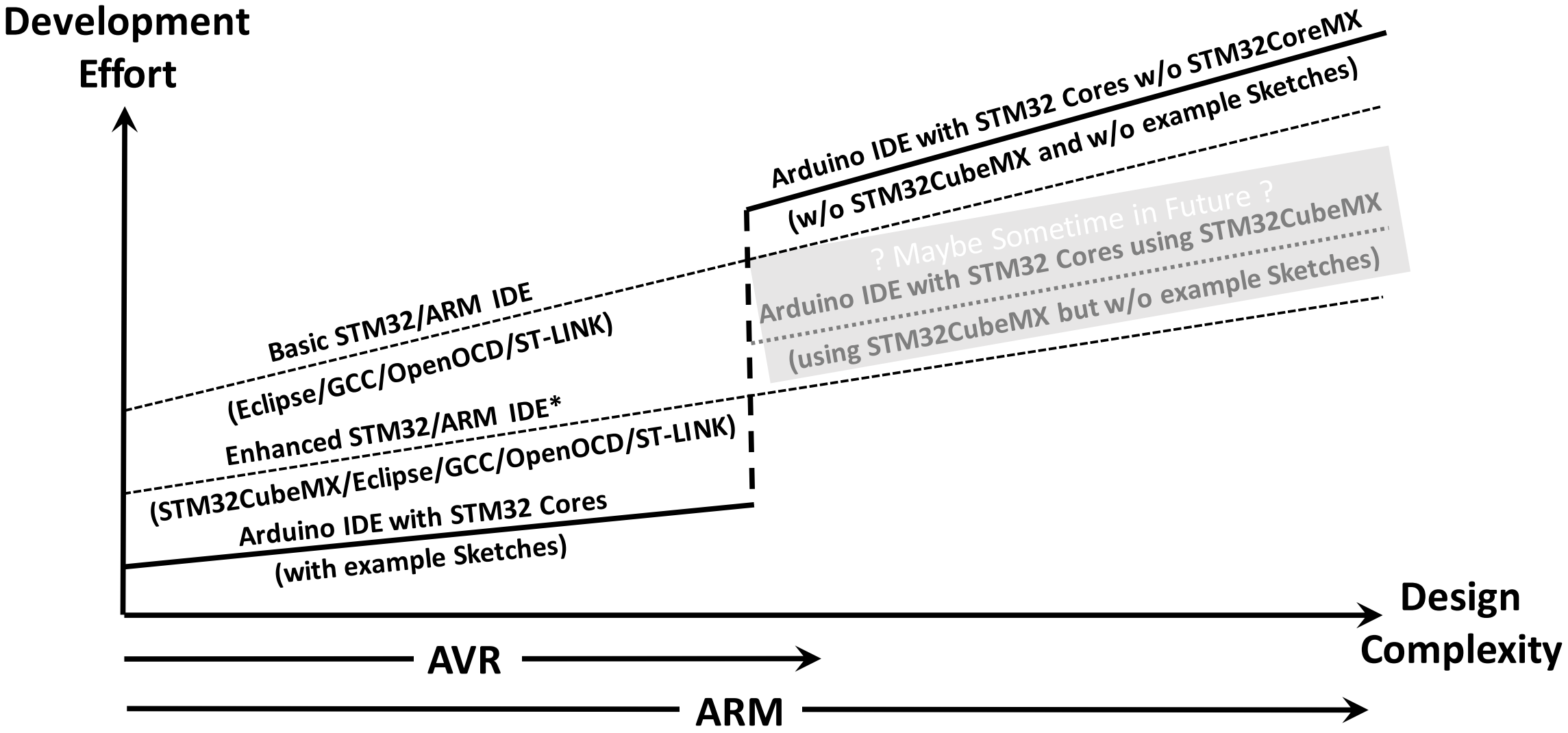
  Arduino IDE beyond Sketch examples for STM32/ARM processors
- Advanced Level STM32/ARM Programming (potential future presentation)

  STM32CubeMX/Eclipse/GCC/ST-LINK programming STM32/ARM processors
- Expert Level STM32/ARM Programming (beyond my paygrade)

  Adding/Integrating new STM32/ARM processor/board to Arduino IDE

  (reference: Add a new variant (board) · stm32duino/wiki Wiki …)

Arduino IDE vs STM32/ARM IDEs



*e.g. Atollic TrueSTUDIO

Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Arduino/STM32 History

- ST-Microelectronic's STM32 ARM Device Firmware Upgrade (DFU) ~ 2008
- Leaflabs' Maple STM32F103 board using DFU and IDE ~ 2009
- Leaflabs' Maple Mini STM32F103 board ~ 2011
- Chinese Blue/Red (STM32F103C8) Pill ~ 2013
- Roger Clark "Arduino IDE with an STM32F103 board" YouTube – 2014
  (https://www.youtube.com/watch?v=-zwGnytGT8M)
- STM32duino wiki ~ 2015
  (http://wiki.stm32duino.com)
- STM32F1xx/STM32 Boards by ST-Microelectronics for Arduino IDE – ***New July 2017***
  (integrates CMSIS*/STM32 HAL and STM32 Register definitions into Arduino IDE)
  (https://community.st.com/community/stm32-community/blog/2017/07/13/stm32-cores-enabled-in-arduino-ide)
- *stm32duino/Arduino_Core_STM32:* https://github.com/stm32duino/Arduino_Core_STM32

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

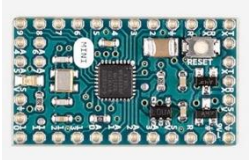
- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Arduino AVR and STM32/ARM Entry Level Devices

AVR

ATmega328

Arduino Pro Mini



Program using USB to Serial

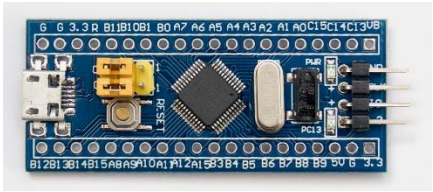
Arduino Uno R3

Program using USB Bootloader

STM32/ARM

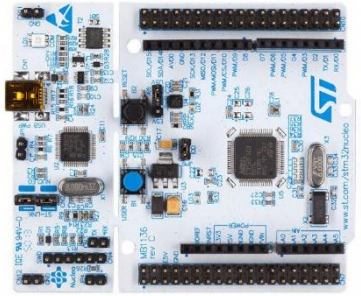
STM32F103

Blue Pill F103C8



Program using USB to Serial,
USB Bootloader, or ST-LINK

Nucleo-F103RB



Program using built-in ST-LINK

Arduino AVR and STM32/ARM Entry Level Devices

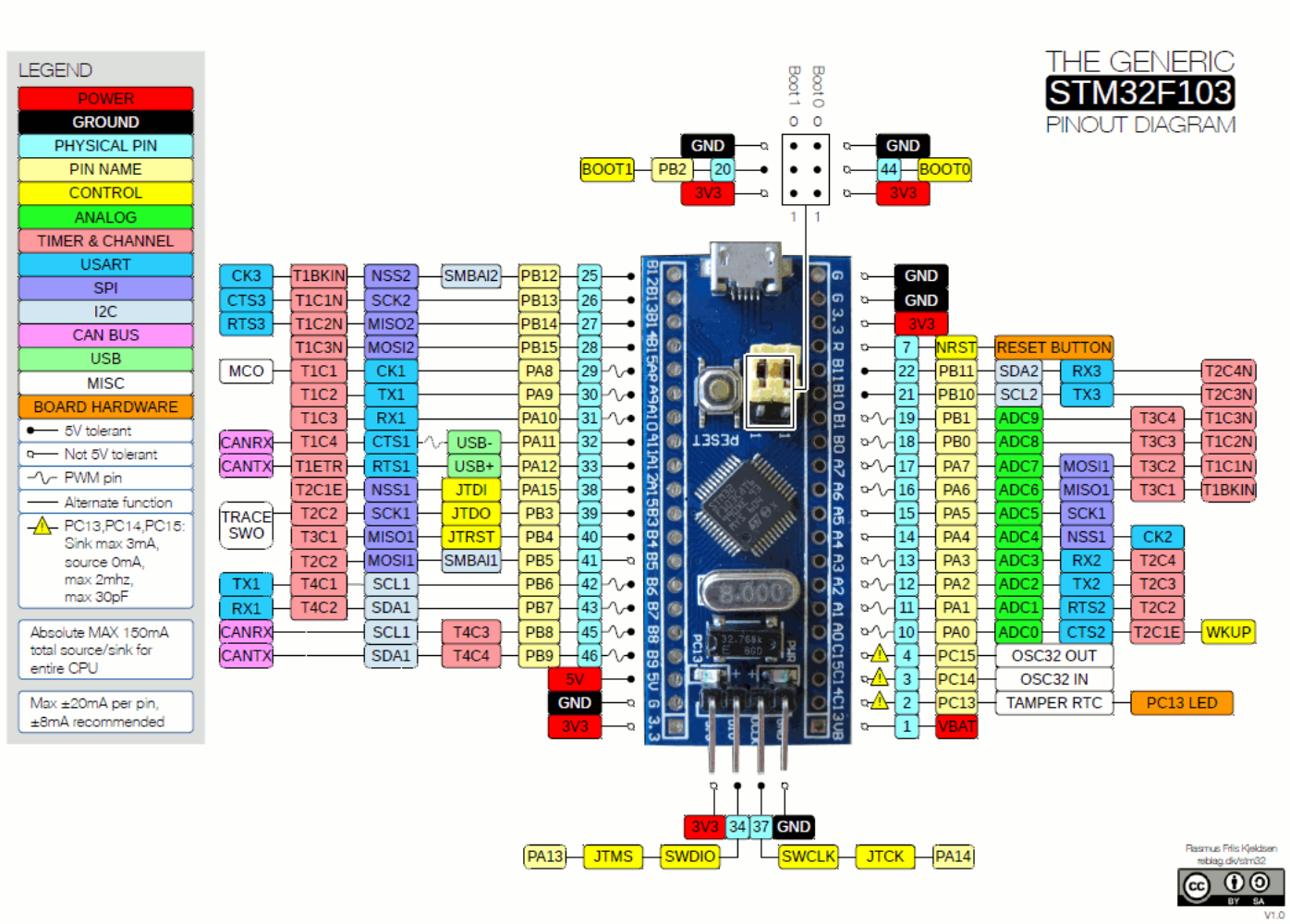
Characristics	Arduino Pro Mini	Arduino Uno	Blue Pill F103C8	Nucleo F103RB
Processor	ATmega328	ATmega328	STM32F103C8	STM32F103RB
Cost	$8 [$4] (Microcenter)	$10 [$8] (Microcenter)	$1.68 (China)@	$10+S&H (Mouser)
Clock	16 MHz	16 MHz	72 MHz	72 MHz
Data	8 bit	8 bit	32 bit	32 bit
Flash	16 KB	16 KB	64 KB	128 KB
RAM	2 KB	2 KB	20 KB	20 KB
EEPROM	1 KB	1 KB	No #	No #
GPIO	14	20	34+1	51
PWM	6	6	15	16
Quadrature Decoder	None	None	4	4
ADC	6 (10 bits @ 0.2 MHz)	6 (10 bits @ 0.2 MHz)	10 (Dual 12 bits @ 1 MHz)	13(Dual 12 bits @ 1 MHz)
DMA	None	None	7 channels	7 channels
Interrupt Controller	No	No	NVIC	NVIC
Vcc	5 V	5 V	3.3 V*	3.3 V*

@ https://www.aliexpress.com/item/mini-Stm32f103c8t6-system-board-stm32-learning-development-board/1568685935.html

[$] Sale Price # Emulate EEPROM with Flash * Most digital pins 5 V tolerant

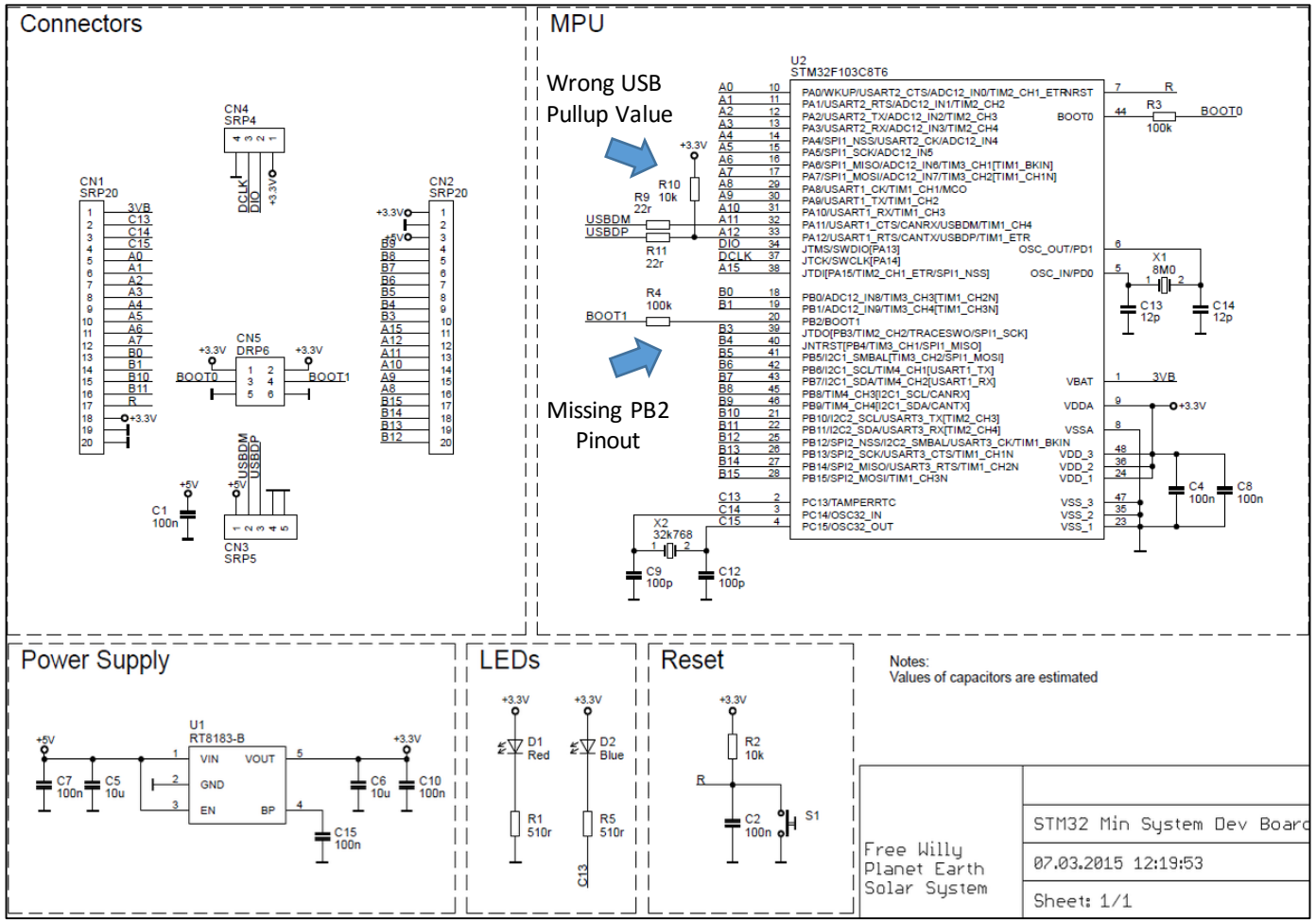
STM32/ARM Entry Level - Blue Pill F103C8 Pinout

34 GPIO Pins



STM32/ARM Entry Level - Blue Pill F103C8 Schematic

With Two Minor Fixable Shortcomings

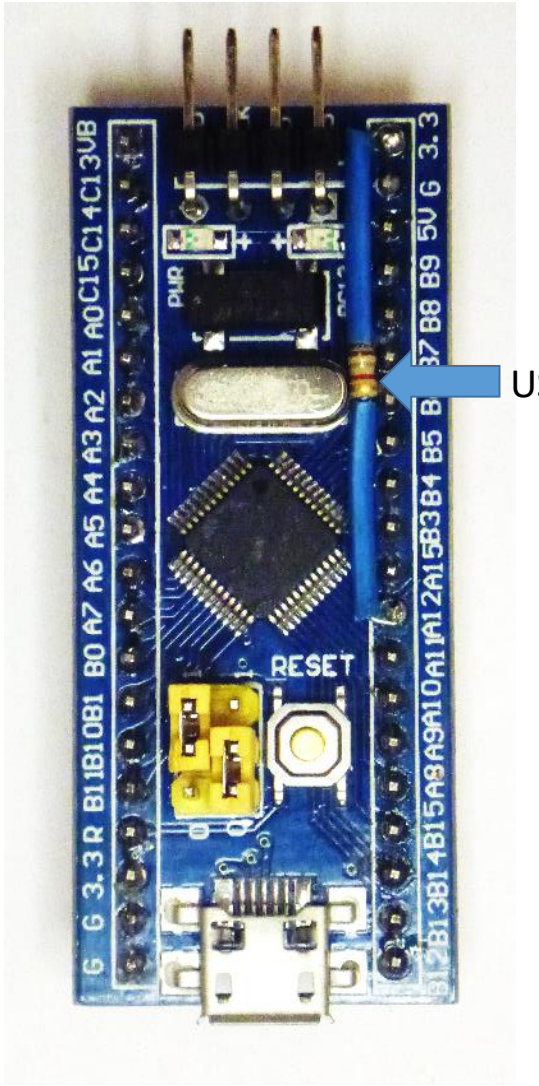


Fixing STM32/ARM Entry Level - Blue Pill F103C8 Wrong USB Pullup Value

USB Spec's 1.5K Pull-Up Resistor

"Blue Pill" has 10K Pull-Up Resistor

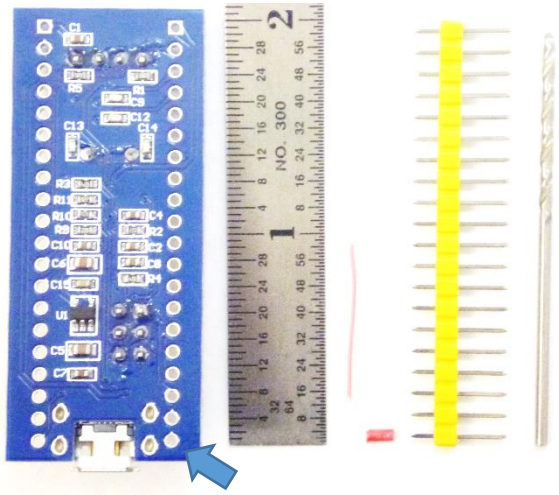
"Blue Pill" needs 1.8K Shunt Resistor between 3.3V and pin PA12



Fixing STM32/ARM Entry Level - Blue Pill F103C8 Lack of PB2 Pinout

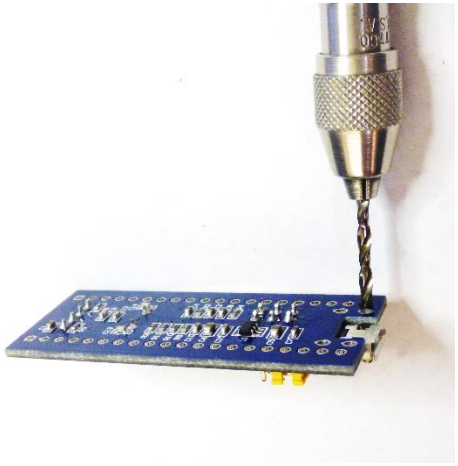
- Board
- Scale
- Jumper Wire
- Pin Insulator
- Terminal Strip
- 5/64” Drill

Start

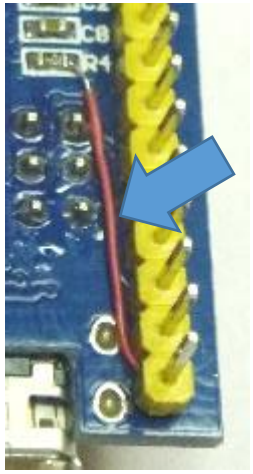


Extra Gnd pad

Drilling out extra Gnd pad



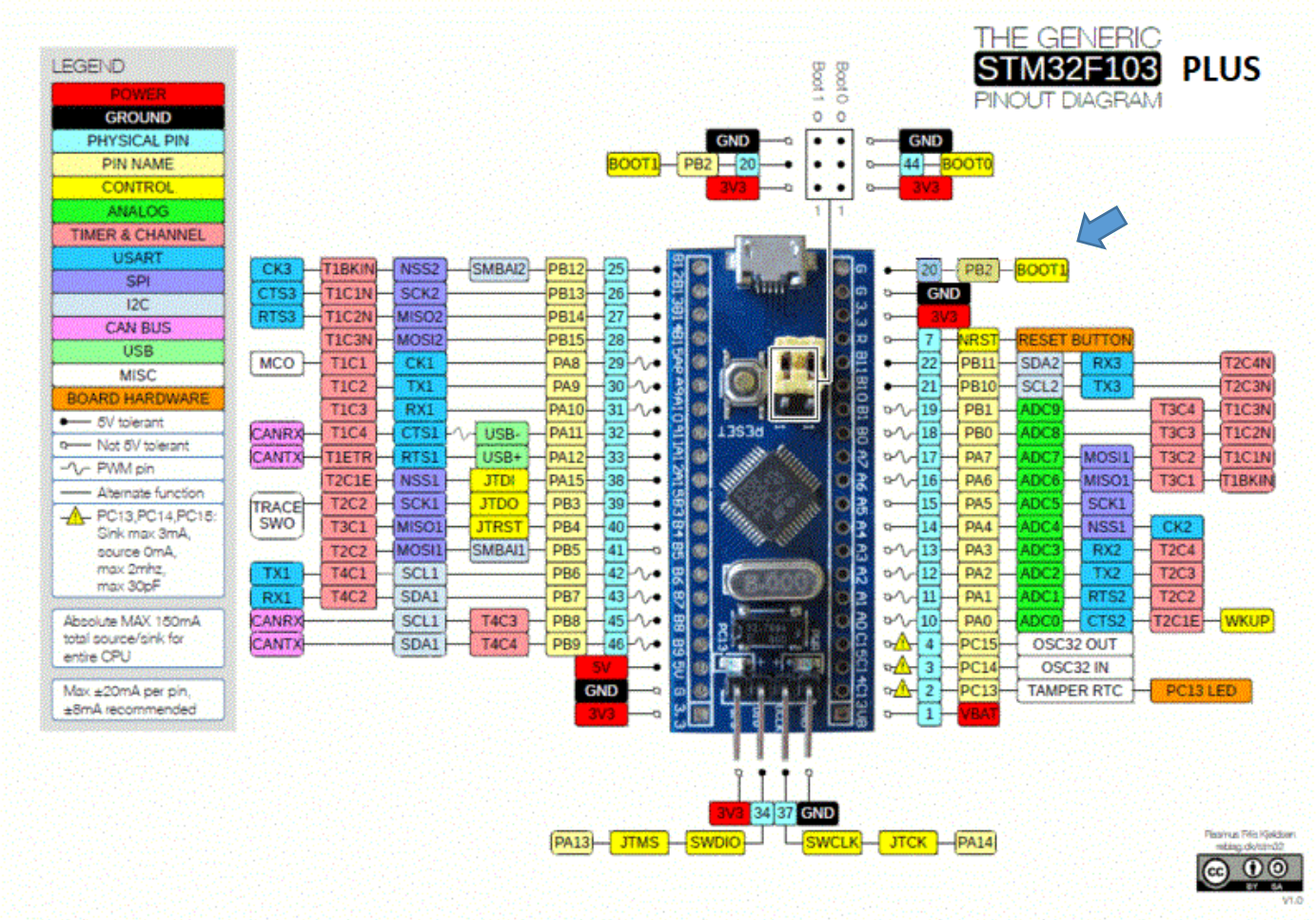
Add
PB2 Jumper Wire
Connected to R4



New PB2 Pin with Insulator

STM32/ARM Entry Level - Blue Pill F103C8 ***Plus*** Pinout

35 GPIO Pins
(with new PB2 pinout)



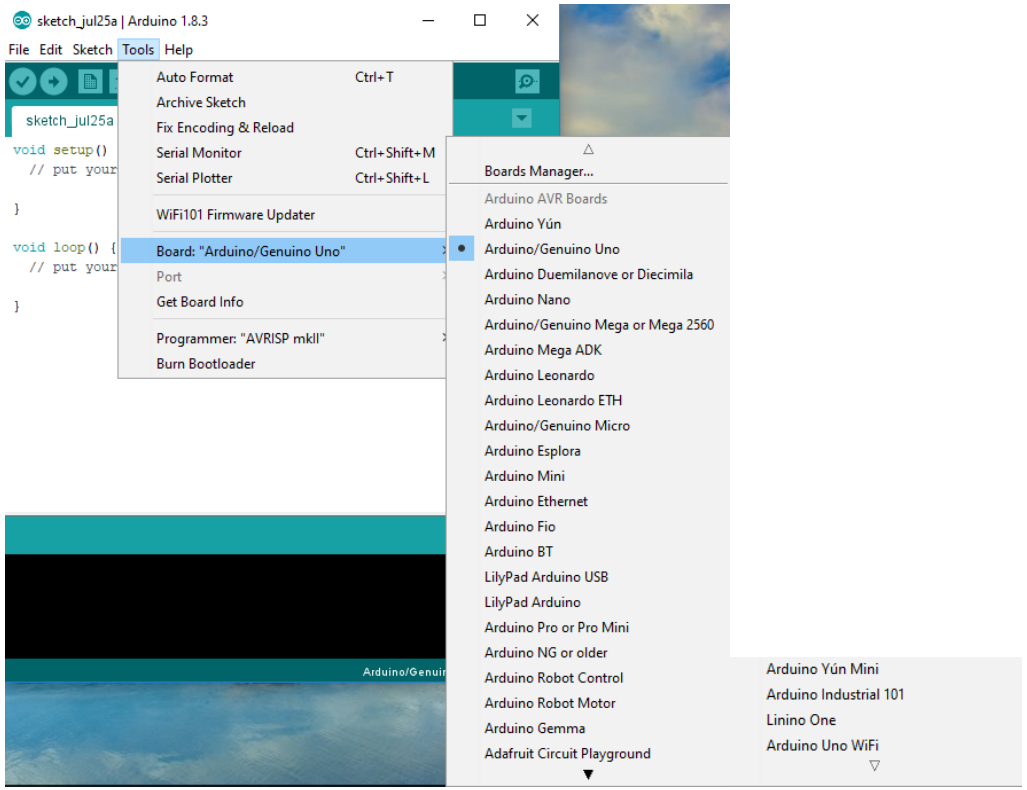
Entry Level STM32/ARM Programming

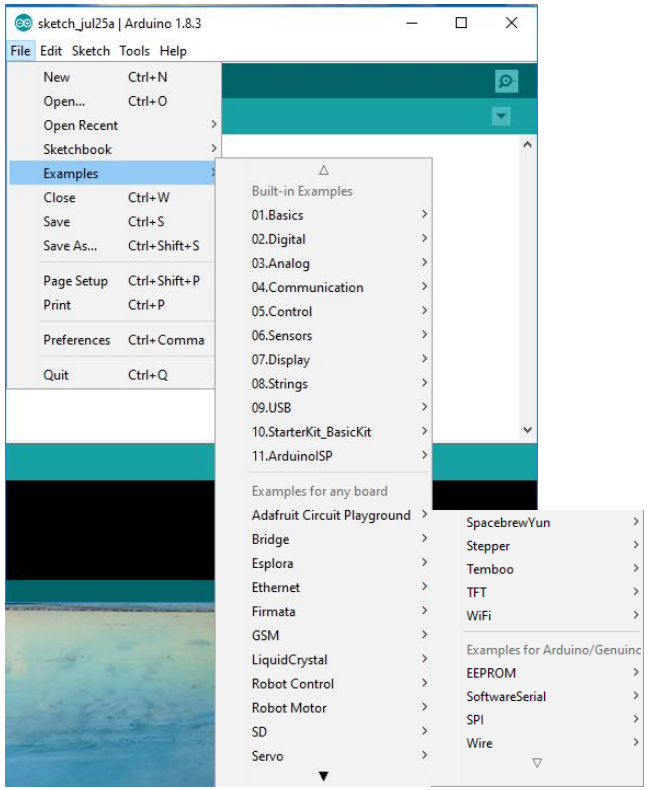
Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Arduino IDE ***without*** STM32F1xx/STM32 Cores by ST-Microelectronics

(First pick the board and then the example)





Arduino IDE ***with*** STM32F1xx/STM32 Cores* by ST-Microelectronics

STM32F1xx Boards

- **Nucleo-F103RB** **128KB 20KB 72MHz**
- **STM32VL Discovery STM32F100RB** **128KB 8KB 24MHz**
- **Blue Pill STM32F103C8** **64KB 20KB 72MHz**
- **MapleMini STM32F103CB** **128KB 20KB 72MHz**

AVR Boards

- Pro Mini 16KB 2KB 16MHz
- Uno R3 16KB 2KB 16MHz

Entry Level
Show-and-Tell

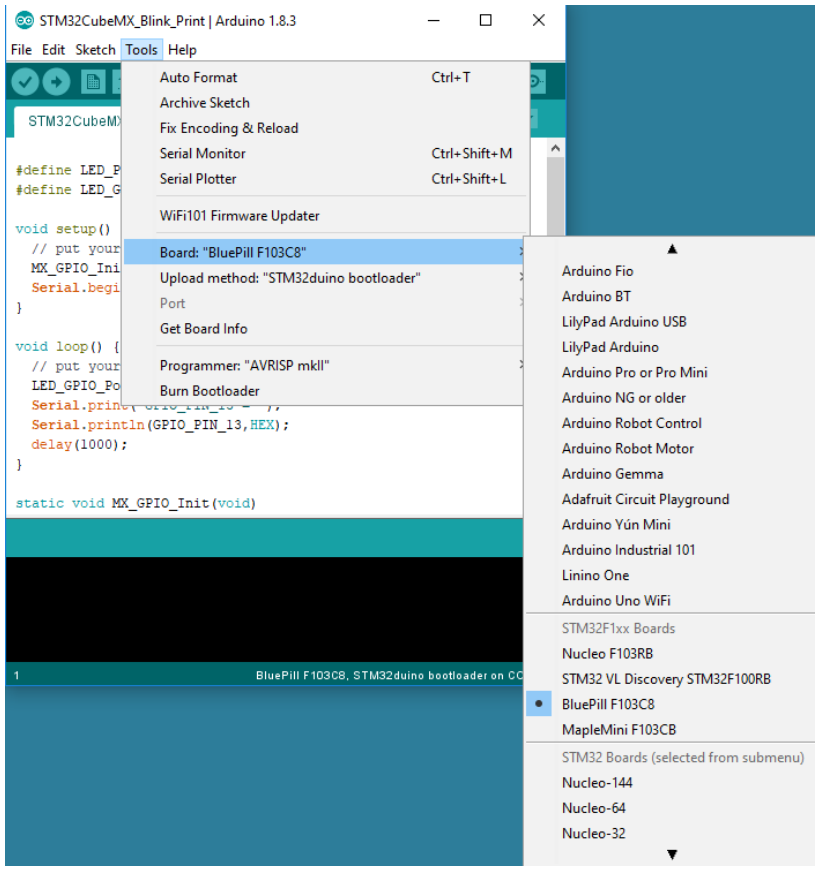
STM32 Boards (with ST-Link)

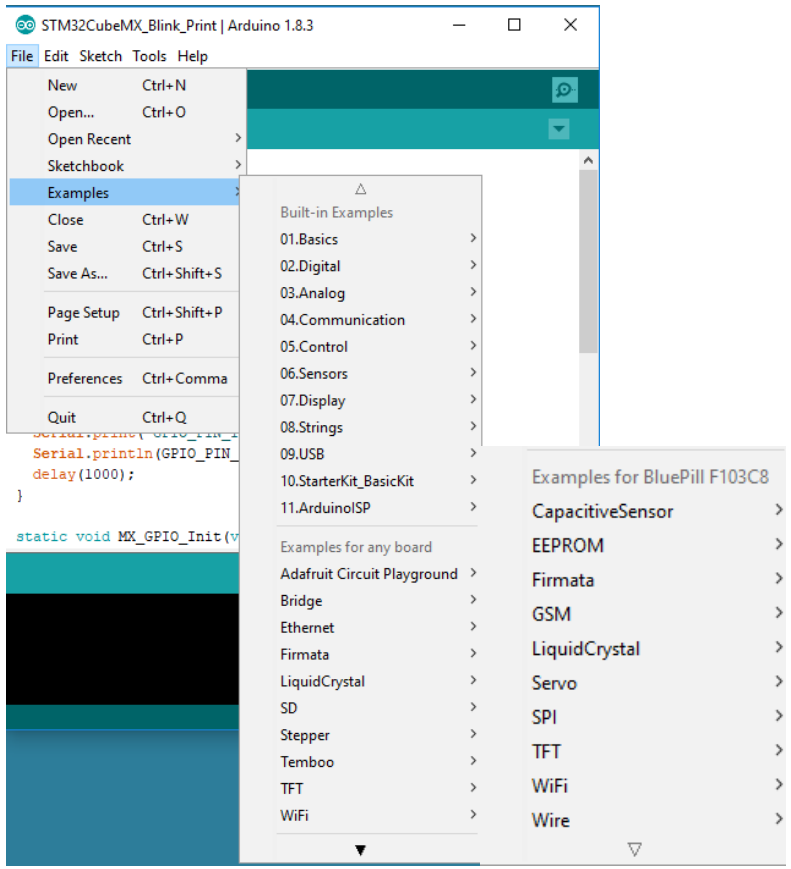
- Nucleo 144
  - Nucleo-F207ZG 1024KB 128KB 120MHz
  - **Nucleo-F429ZI** **2048KB 256KB 180MHz**
- Nucleo 64
  - Nucleo-F030R8 64KB 8KB 48MHz
  - Nucleo-F091RC 256KB 32KB 48MHz
  - **Nucleo-F103RB** **128KB 20KB 72MHz**
  - Nucleo-F303RB 128KB 40KB 72MHz
  - **Nucleo-F401RE** **512KB 96KB 84MHz**
  - **Nucleo-F411RE (DPRG alt CPU)** **512KB 128KB 100MHz**
  - Nucleo-L053R8 64KB 8KB 32MHz
  - Nucleo-L152RE 512KB 80KB 32MHz
  - **Nucleo-L476RG** **1024KB 128KB 80MHz**
- Nucleo 32
  - Nucleo-L432KC 256KB 64KB 80MHz
- Discovery
  - **STM32F100RB-DISCVL** **128KB 8KB 24MHz**
  - STM32F407G-DISC1 1024KB 192KB 168MHz
  - **STM32F746G-DISCOVERY** **1024KB 340KB 216MHz**

* ST-Microelectronics is in the process of integrating Blue Pill and MapleMini into single STM32 Boards package

Arduino IDE ***with*** STM32F1xx/STM32 Cores* by ST-Microelectronics

(First pick the board and then the example)



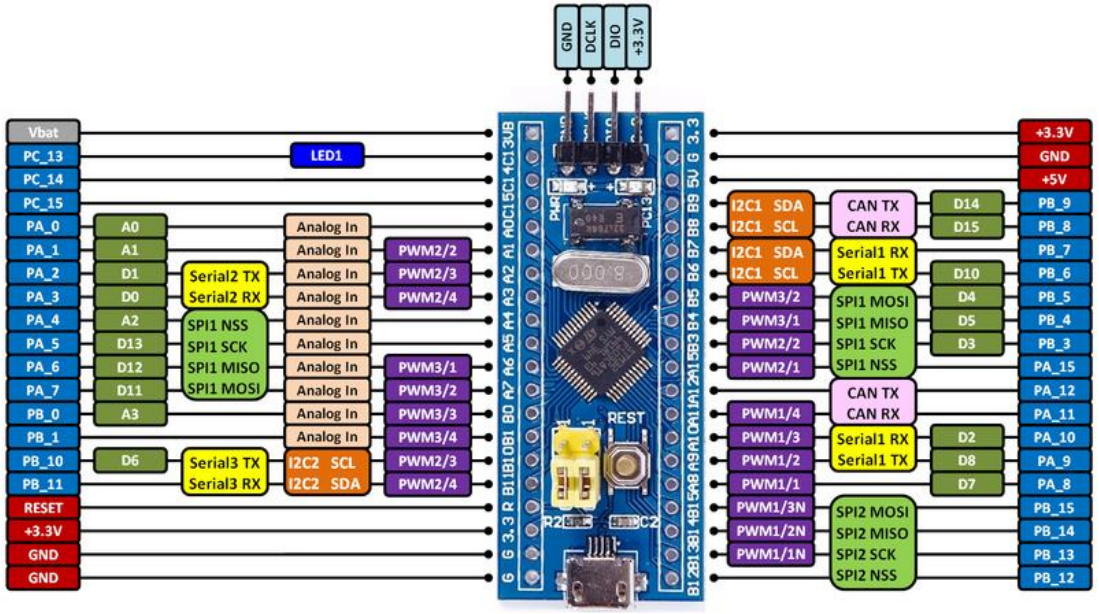


* ST-Microelectronics is in the process of integrating Blue Pill and MapleMini into single STM32 Boards package

Arduino IDE **with** STM32F1xx/STM32 Cores by ST-Microelectronics – Pin Mapping

Arduino			Roger Clark's Blue Pill	Nucleo 64	
				F030R8	
				F091RC	
				F103RB	
				F401RE	
				F411RE	
				L053R8	
				L476RG	
D0	0	RX	PA3	PA3	USART2_RX
D1	1	TX	PA2	PA2	USART2_TX
D2	2		PA10	PA10	
D3	3	PWM	PB3	PB3	TIM2_CH2
D4	4		PB5	PB5	
D5	5	PWM	PB4	PB4	TIM3_CH1
D6	6	PWM	PB10	PB10	TIM2_CH3
D7	7		PA8	PA8	
D8	8		PA9	PA9	
D9	9	PWM	n/a	PC7	TIM3_CH2
D10	10	PWM/CS	PB6	PB6	TIM4_CH1/SPI1_CS
D11	11	PWM/MOSI	PA7	PA7	TIM3_CH2/SPI1_MOSI
D12	12	MISO	PA6	PA6	SPI1_MISO
D13	13	SCK [User LED]	PA5	PA5	SPI1_SCK [User LED]
D14		SDA	PB9	PB9	I2C1_SDA
D15		SCL	PB8	PB8	I2C1_SCL
A0	14		PA0	PA0	ADC_0
A1	15		PA1	PA1	ADC_1
A2	16		PA4	PA4	ADC_4
A3	17		PB0	PB0	ADC_8
A4	18		n/a	PC1	ADC_11
A5	19		n/a	PC0	ADC_10
			PC13 User LED	PC13	User Button

Roger Clark's Blue Pill Pinout



Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE

Setup – Assumes Arduino IDE already installed

- In Arduino IDE, click File -> Preferences.
- Click on the edit button next to 'Additional Boards Manager URLs'.
- Add the URL of the STM32F1xx/STM32 Cores boards manager package json file*:

https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/d1c46284dec37305d563a87eecfb545e3c66a166/STM32/package_stm_index.json

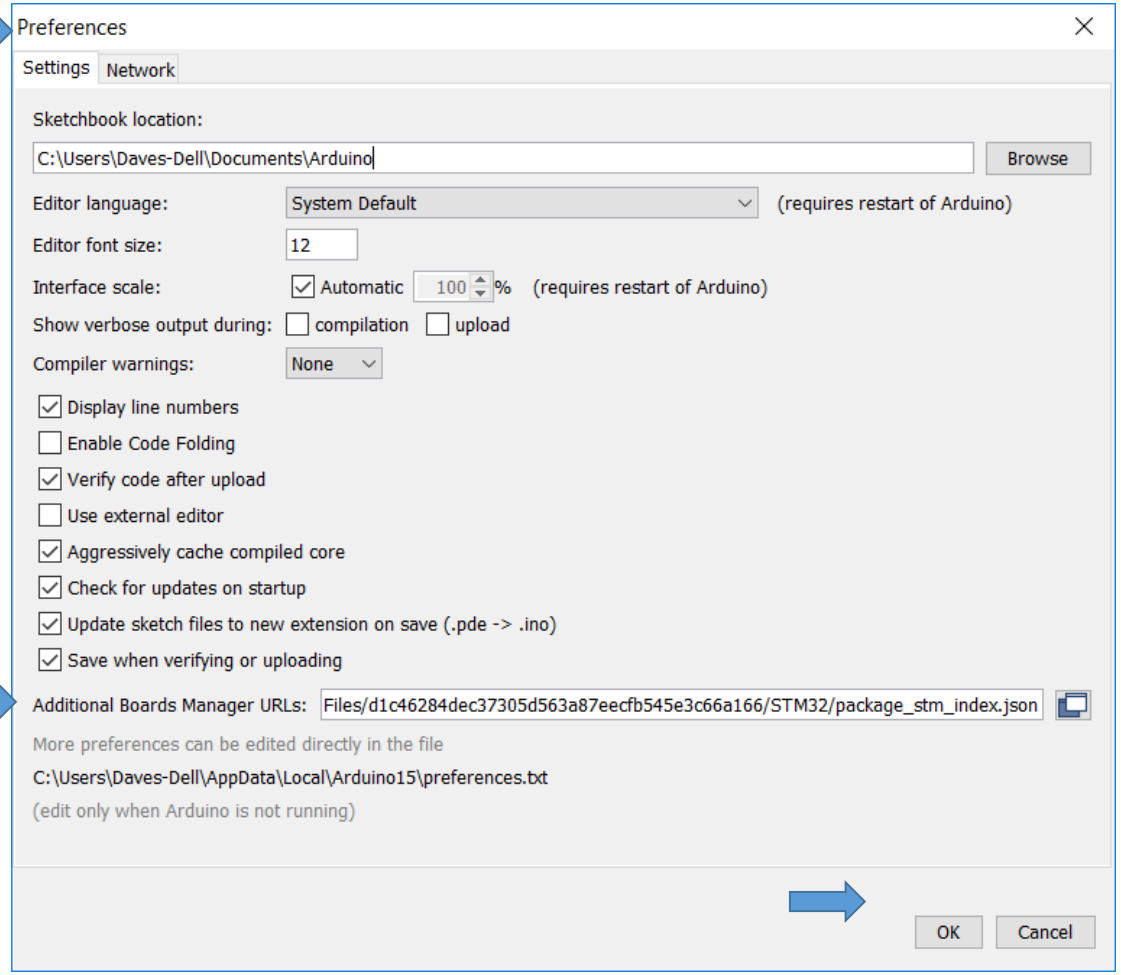
- Click OK
- Click Tools -> Board -> Board Manager… -> Type -> Contributed
- Select: STM32F1xx Cores by ST-Microelectronics -> Install
- Select: STM32 Cores by ST-Microelectronics -> Install

* WARNING: This URL is for the previous release which includes STM32F1 support which is missing from latest release.

Installing STM32F1xx/STM32 Cores* by ST-Microelectronics into Arduino IDE

(Arduino IDE: File -> Preferences)

File -> Preferences



https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/d1c46284dec37305d563a87eecfb545e3c66a166/STM32/package_stm_index.json

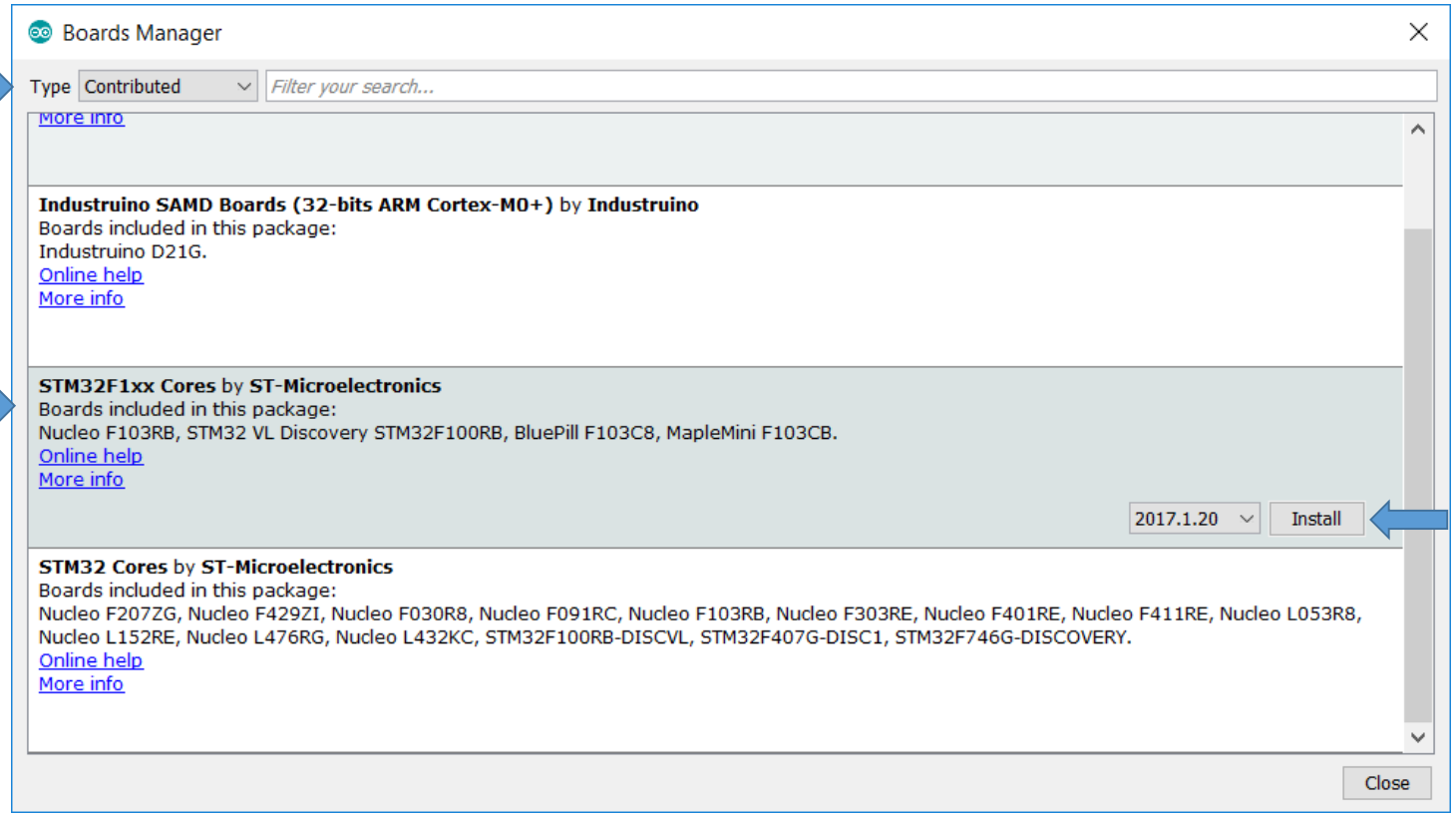
* WARNING: This URL is for the previous release which includes STM32F1 support which is missing from latest release.

Installing STM32F1xx Cores by ST-Microelectronics into Arduino IDE

(Arduino IDE: Tools -> Board -> Board Manager)

Contributed

Select



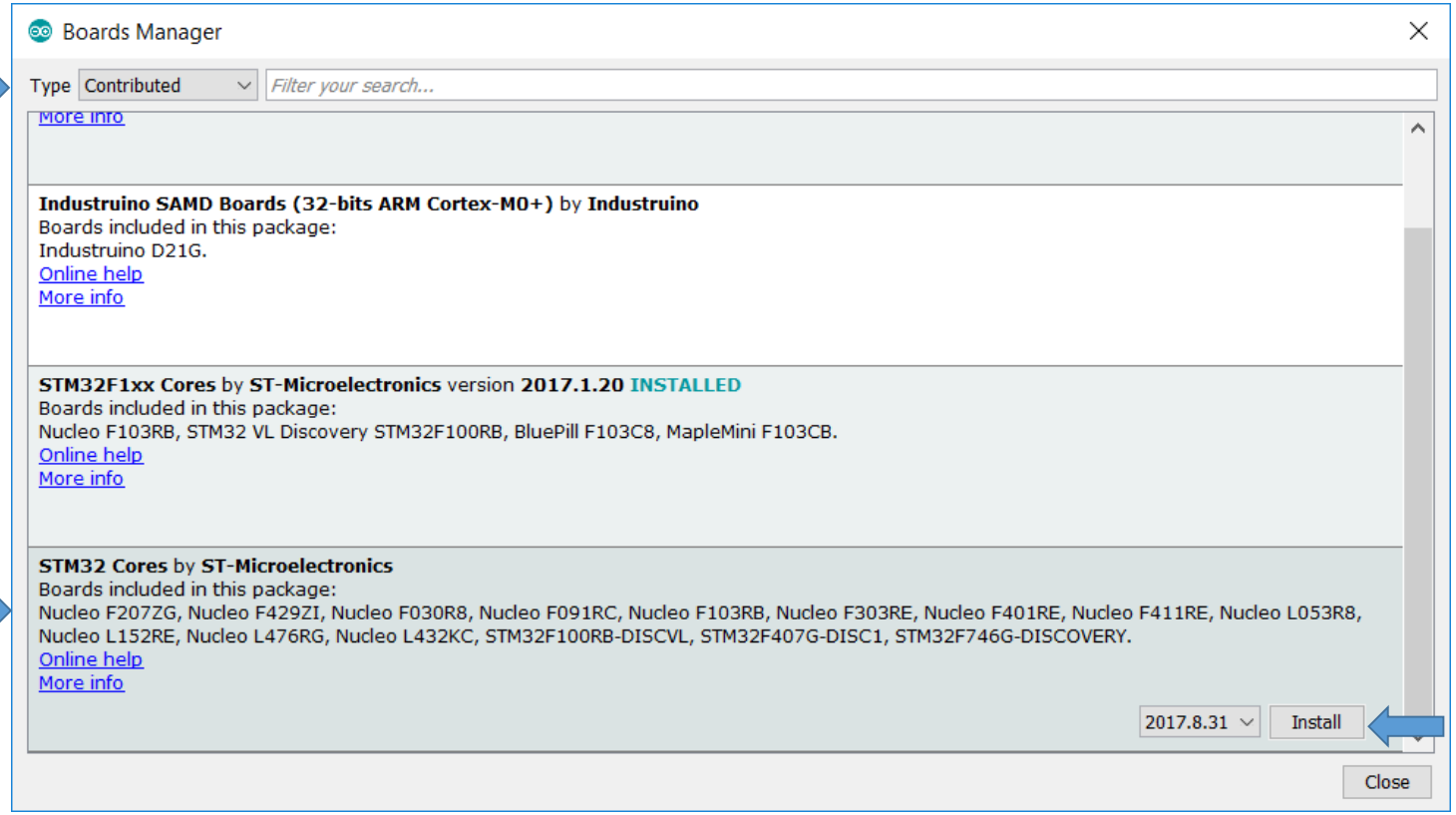
Install

Installing STM32 Cores by ST-Microelectronics into Arduino IDE

(Arduino IDE: Tools -> Board -> Board Manager)

Contributed

Select



Install

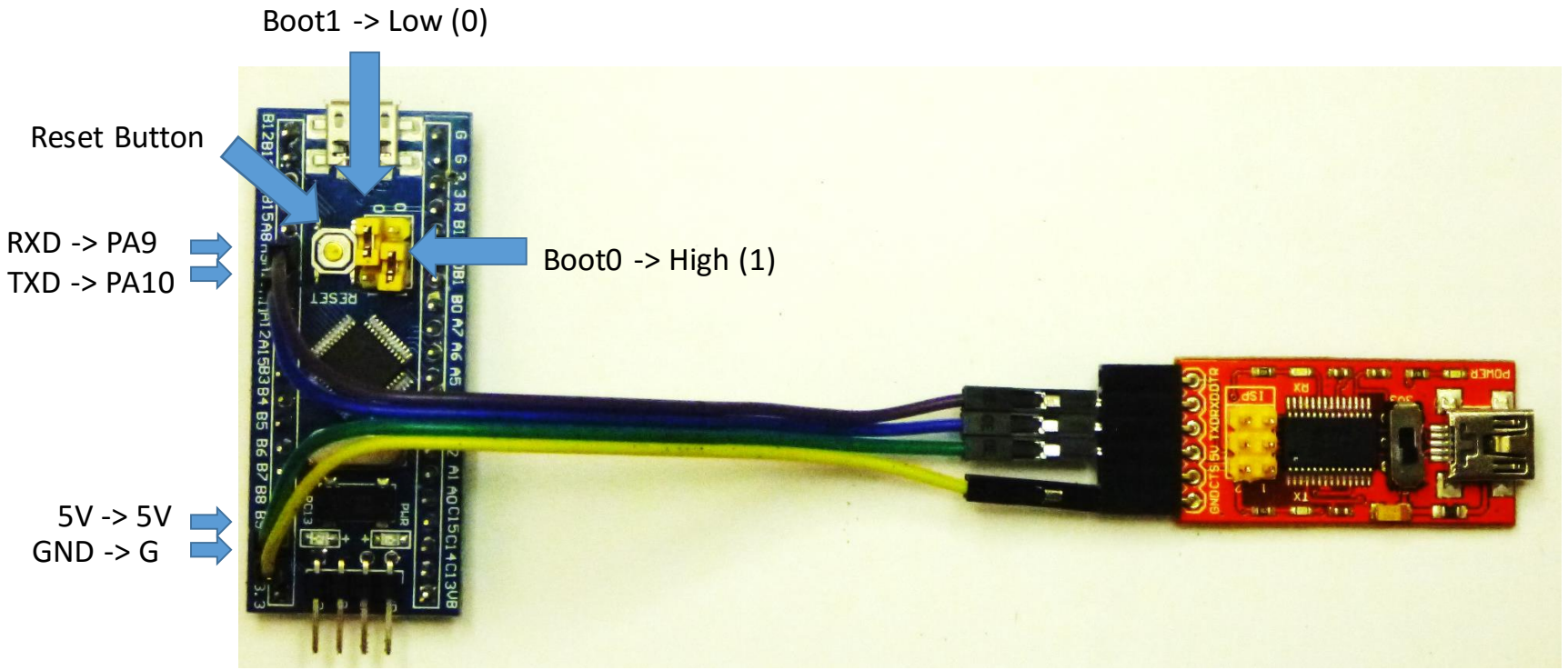
Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Downloading Arduino Blink Sketch to Blue-Pill F103C8 using USB-to-Serial

For serial downloading, set Boot0 pin high (1) and Boot1 pin low (0)



Note: STM32F103C8 UART1 pins (PA9/10) are 5V tolerant => 5V USB to RS232 Adapters are OK

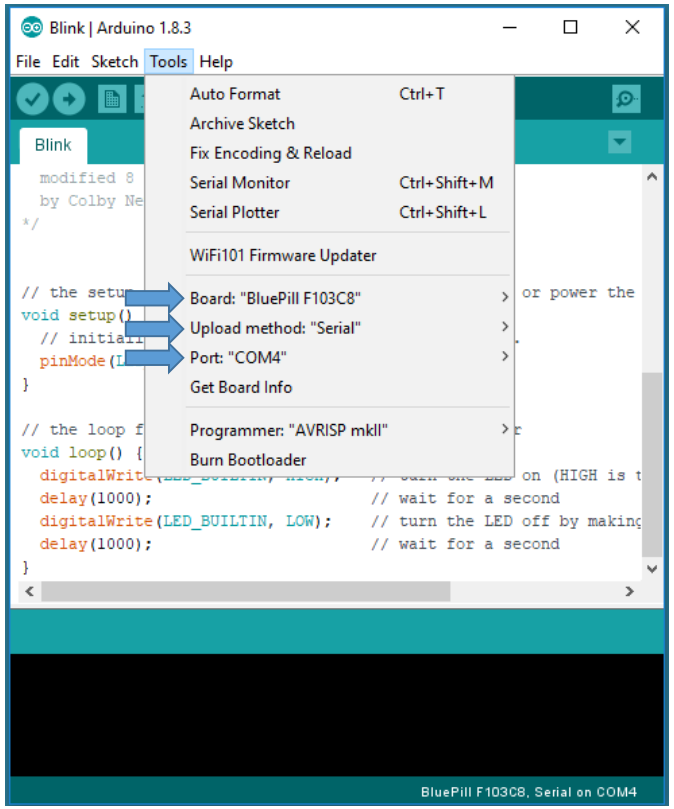
Downloading Arduino Blink Sketch to Blue-Pill F103C8 using USB-to-Serial

Setting Up Blue Pill F103C8

- Tools -> Board -> “Blue Pill F103C8”
- Tools -> Upload Method -> “Serial”
- Tools -> Port -> “COM4” [use Device Manager to check]

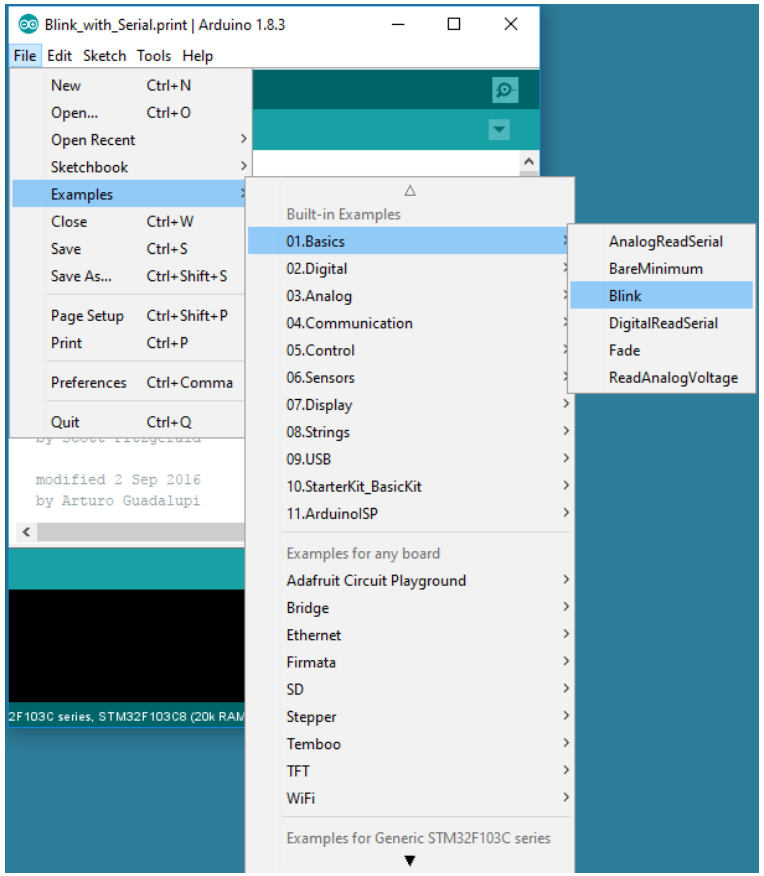
Downloading Arduino Blink Sketch to Blue-Pill F103C8 using USB-to-Serial

Setting Up Blue Pill F103C8



Downloading Arduino Blink Sketch to Blue-Pill F103C8 using USB-to-Serial

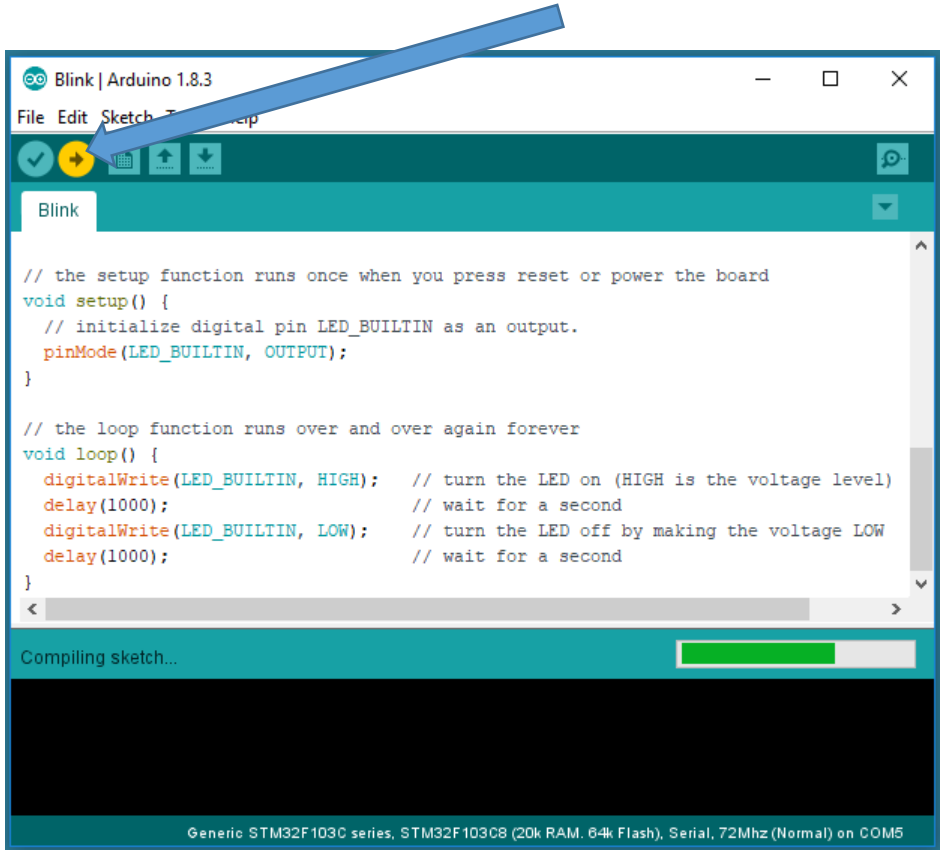
File -> Examples -> 01. Basic -> Blink



Downloading Arduino Blink Sketch to Blue-Pill F103C8 using USB-to-Serial

(Change delay values to confirm download)

First press the board reset and then Start Compile/Upload

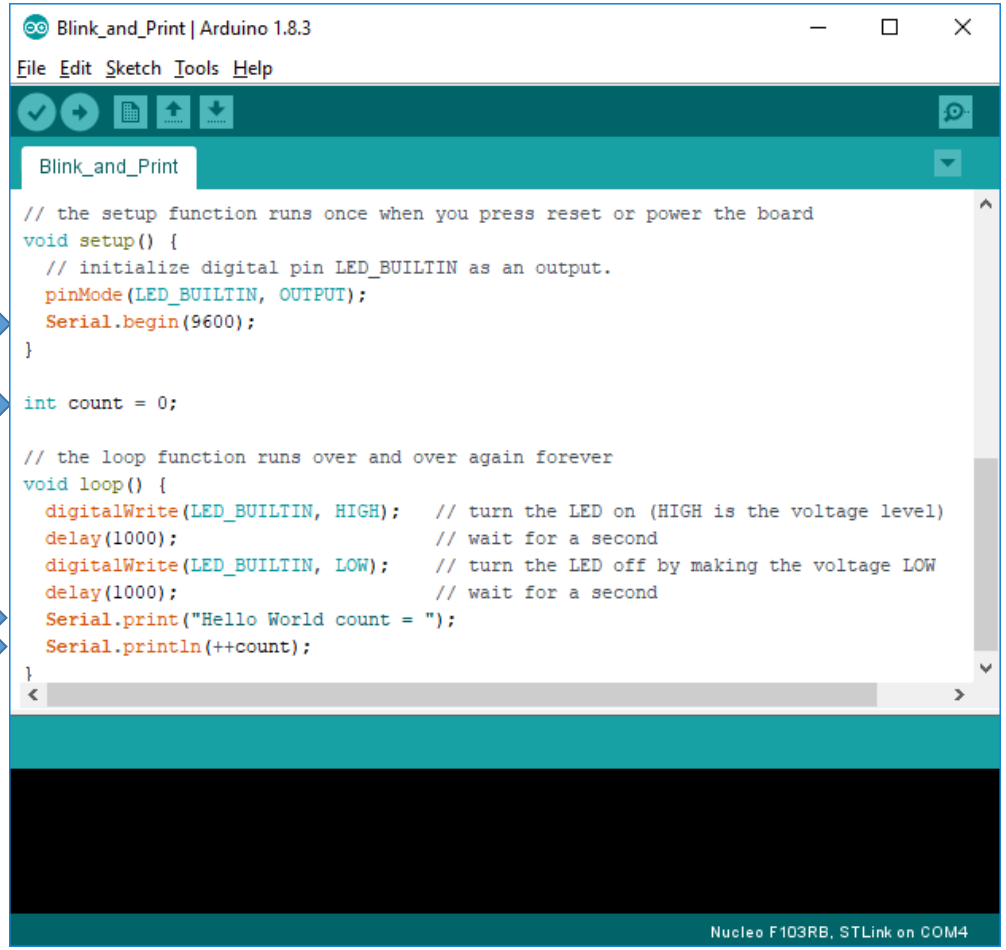


Enhancing Blink Sketch to include Serial Print and save-as Blink_and_Print

Initialize Serial Print

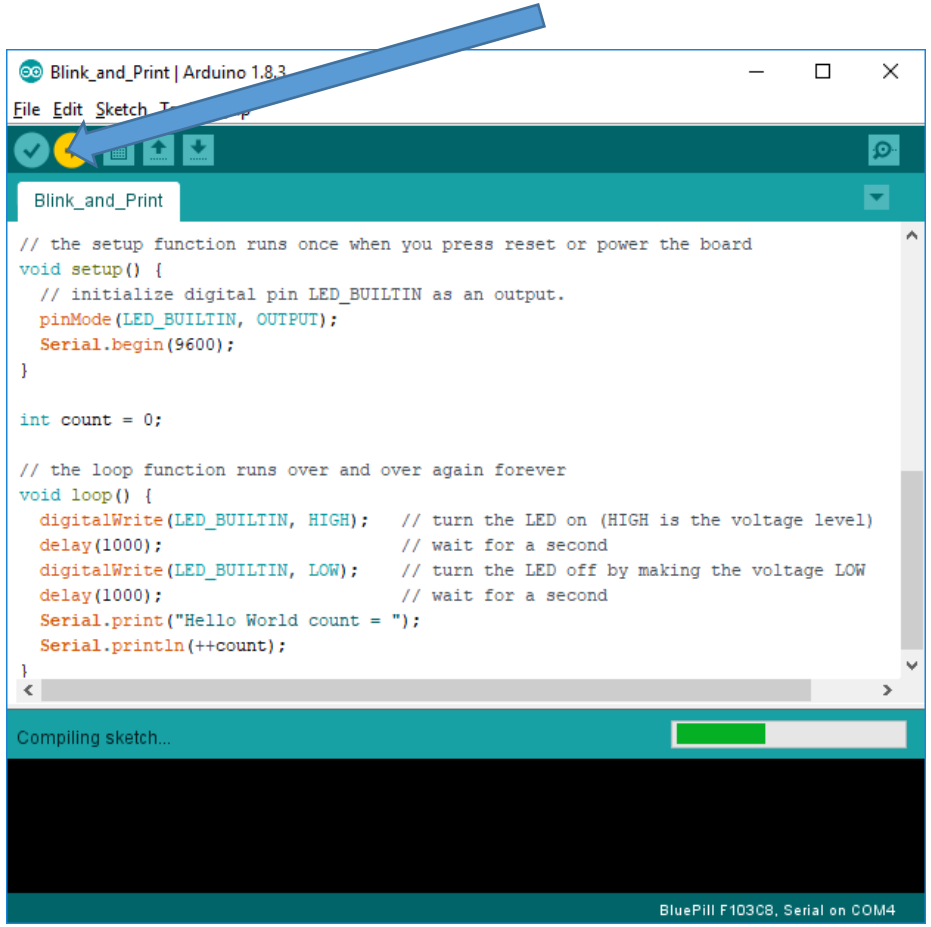
Define Print Variable

Print Text without Line End
Inc Variable & Print with Line End



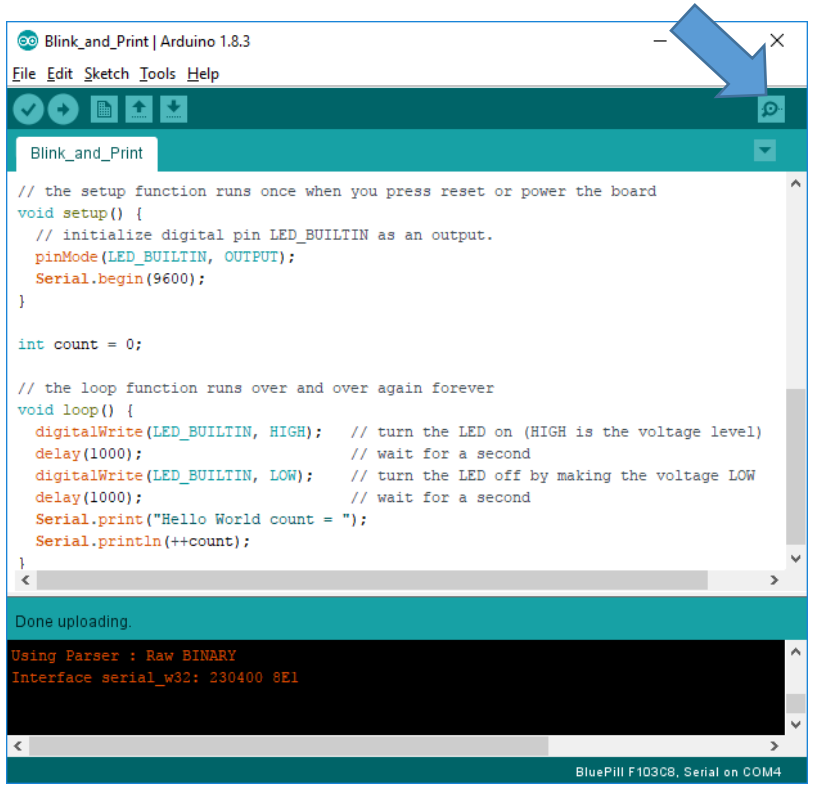
Downloading Arduino Blink_and_Print Sketch to Blue-Pill F103C8 using USB-to-Serial

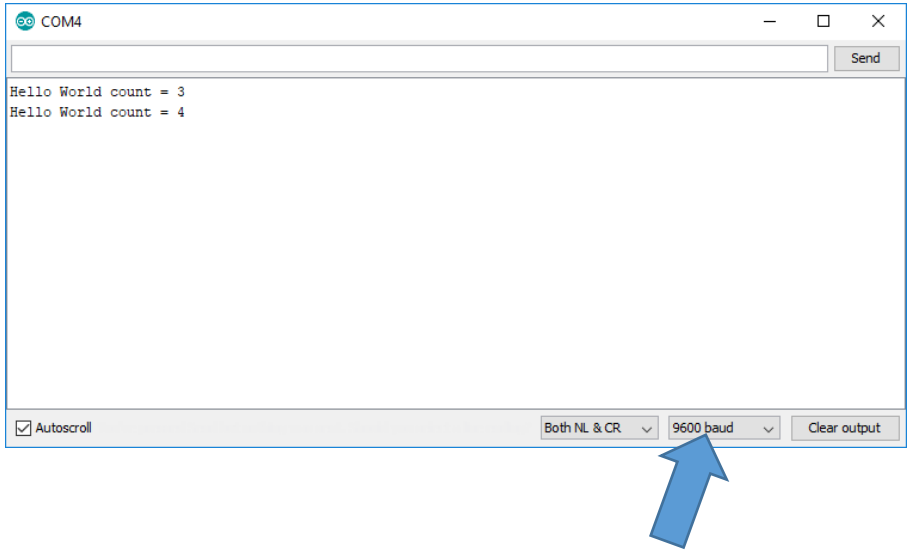
First press the board reset and then Start Compile/Upload



Running Arduino Blink_and_Print Sketch to Blue-Pill F103C8 using USB-to-Serial

Select Serial Monitor, set baud rate, and observe results





Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial

(Add 1.8K USB pull up shunt resistor to match USB's 1.5K requirement)

- **Download and Install** maple DFU and STM COM drivers* from: https://github.com/rogerclarkmelbourne/Arduino_STM32/archive/master.zip
  - **Unzip** into Documents\Arduino\hardware directory (may need to create hardware directory)
  - **Click** Documents\Arduino\hardware\Arduino_STM32-master\drivers\win**install_drivers.bat**
  - **Click** Documents\Arduino\hardware\Arduino_STM32-master\drivers\win**install_STM_COM_drivers.bat**
  - **Restart** computer to complete the driver installation process

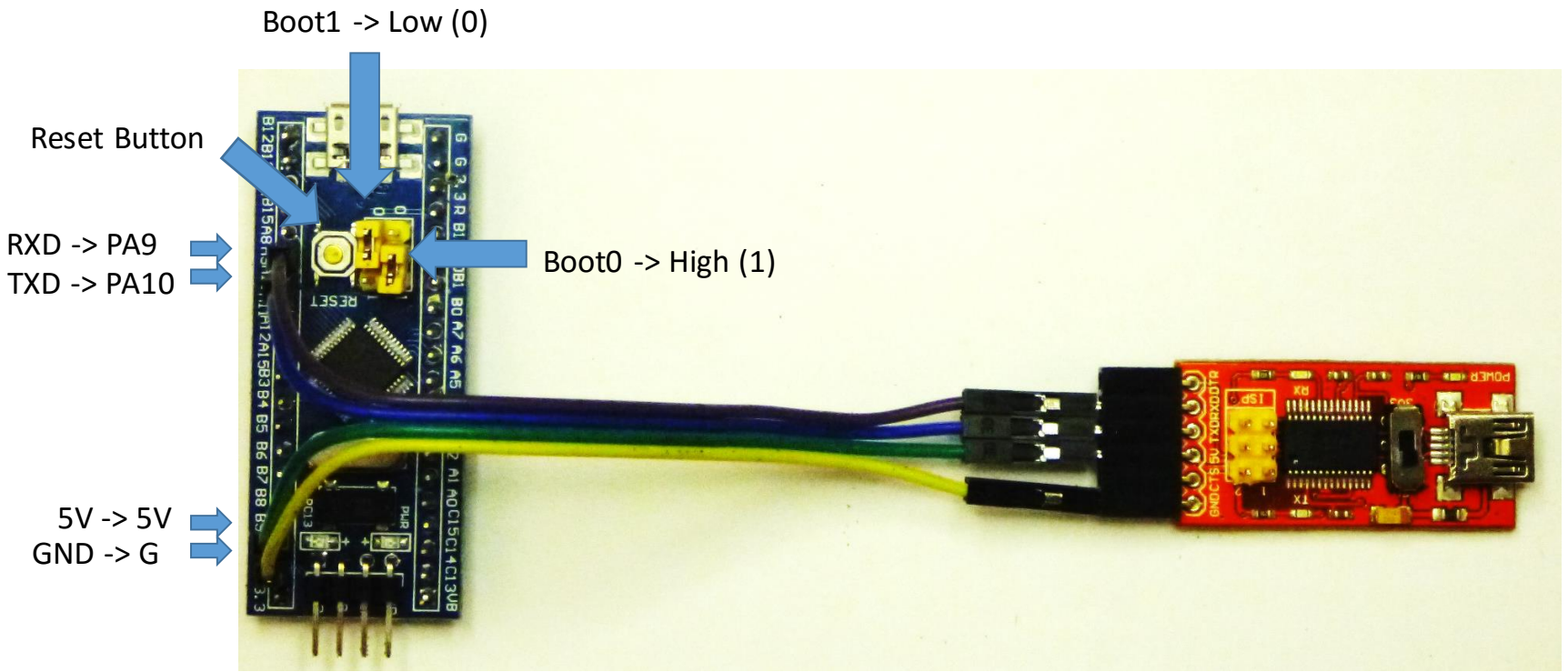
- **Download and install** ST-Microelectronics' FlashLoader ***Demonstrator*** v2.8.0 (en.flasher-stm32.zip) part number: "FLASHER-STM32" from: http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/flasher-stm32.html

- **Download bootloader binary** appropriate based on Blue Pill STM32F103C8's LED pin from: https://github.com/rogerclarkmelbourne/STM32duino-bootloader/tree/master/bootloader_only_binaries
  (Most Blue Pill boards have LED on pin PC13, thus use "generic_boot20_pc13.bin" bootloader)

***WARNING** – May need to reinstall drivers after any Window 10 updates

Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial with ST-Micro's Flash Loader Demonstrator

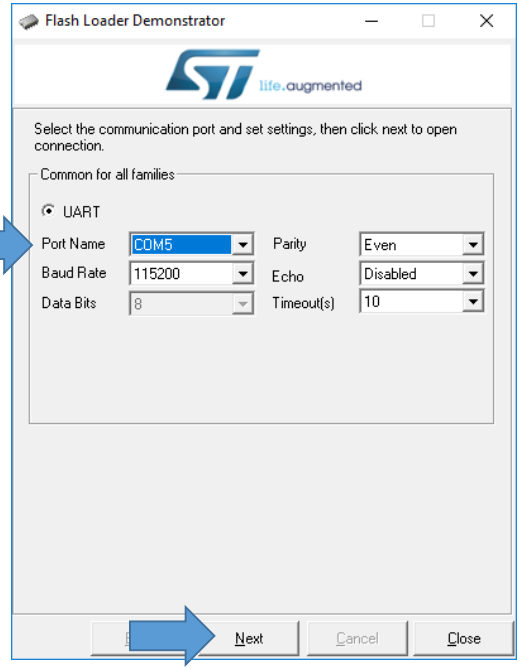
For serial download, set Boot0 pin high (1), Boot1 pin low (0), and Press Reset



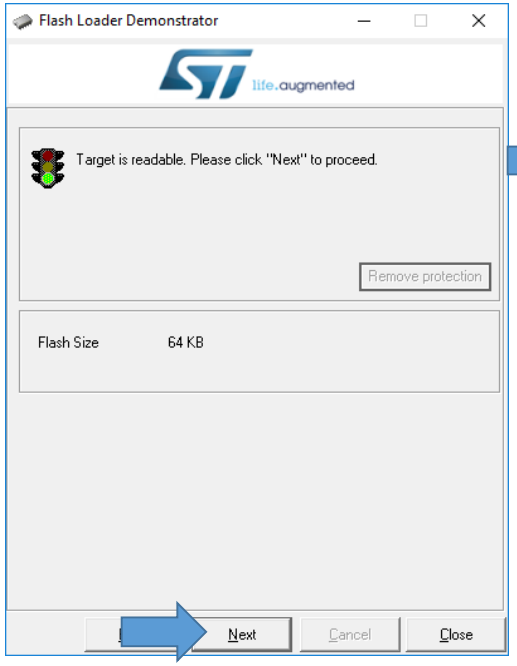
Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial with ST-Micro's Flash Loader Demonstrator

Open ST-Microelectronics' FlashLoader Demonstrator v2.8.0

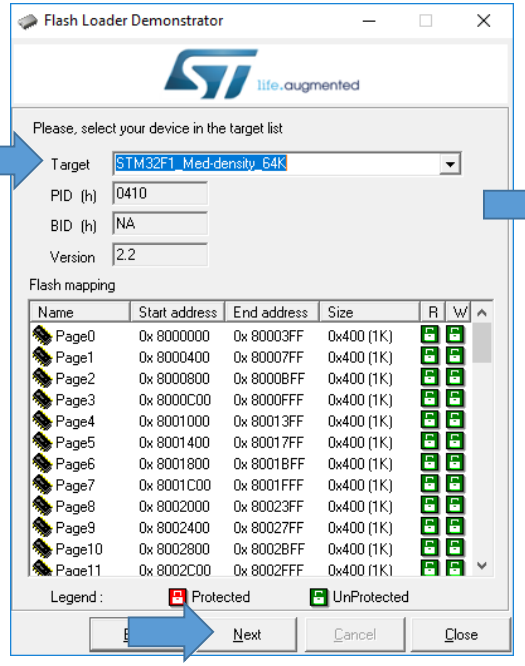
Select COM port and click Next



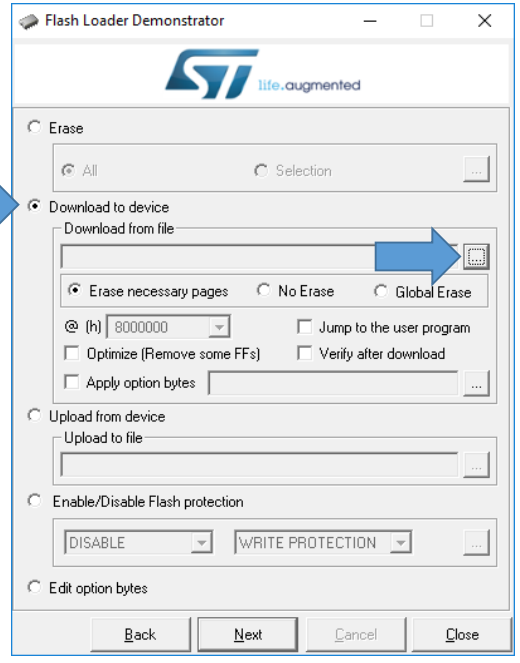
Verify Target and click Next



Select Target (64K) and click Next

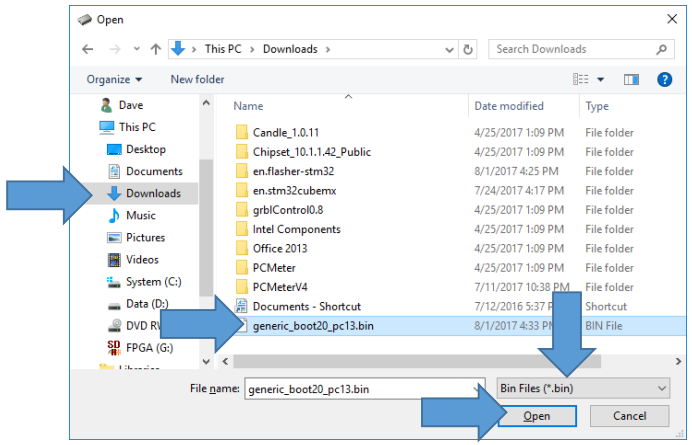


Select Download and Browse

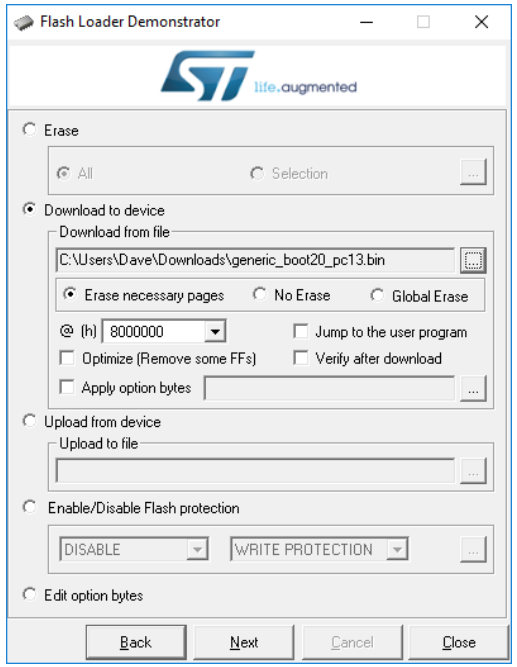


Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial with ST-Micro's Flash Loader Demonstrator

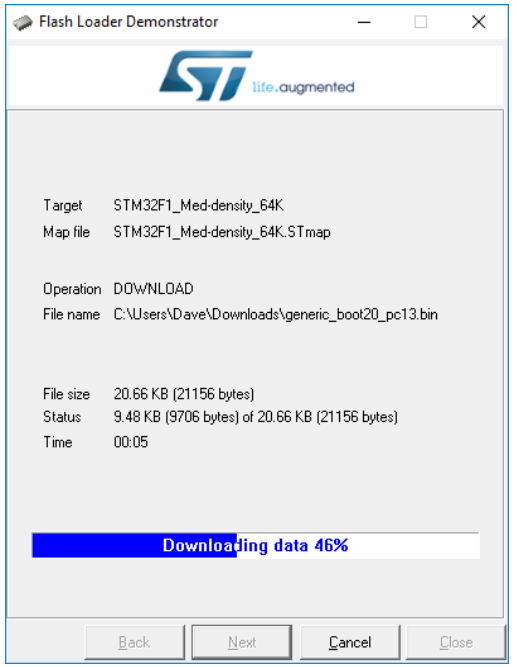
Select Directory,
Select Bin Files (*.bin),
Select Bootload File,
and Select Open,



Verify File
and click Next



File will start
Downloading



When Download Finishes
Click Close and remove
USB-to-Serial & Serial Port*

*Use Device Manager with Show Hidden Devices to remove old Serial Port

Switching Downloading from USB-to-Serial to USB Bootloader

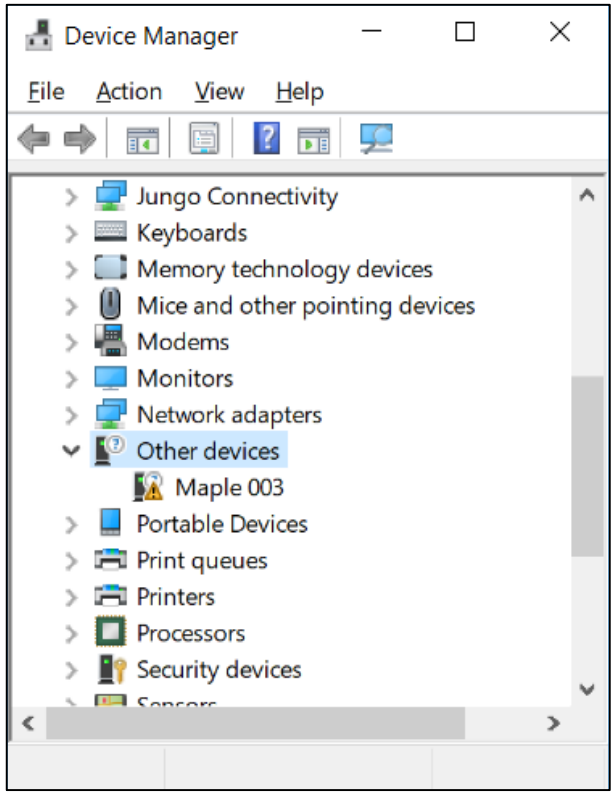
After downloading/installing USB Bootloader

- Remove USB-to-Serial connections RDX -> PA9/TDX -> PA10 and 5V/Gnd
- Reset Boot0 and Boot1 jumpers to Low (0)
- Add 1.8K between 3.3V and PA12 to shunt internal 10K USB resistor
- Remove old Serial Port using Device Manager with Show Hidden Devices
- Add USB-to-PC cable to STM32F103C8/Blue Pill
- Use Device Manager “View > Show hidden devices” to check Maple driver

Checking Maple Driver Correctly Loaded

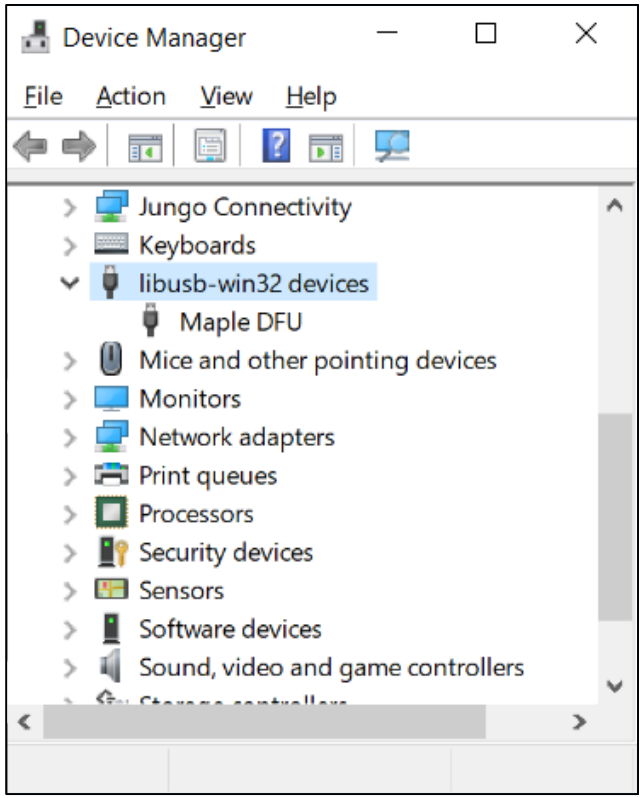
Device Manager: View > Show hidden devices

Other devices/Maple 003 => ***Missing Maple Driver***



<u>Reinstall</u>* Maple DFU drivers and Restart PC

libusb-win32 device/Maple DFU => ***Correct Maple Driver***



* **Click** Documents\Arduino\hardware\Arduino_STM32-master\drivers\win**install_drivers.bat**
Click Documents\Arduino\hardware\Arduino_STM32-master\drivers\win**install_STM_COM_drivers.bat**
Restart computer to complete the driver installation process

Entry Level STM32/ARM Programming

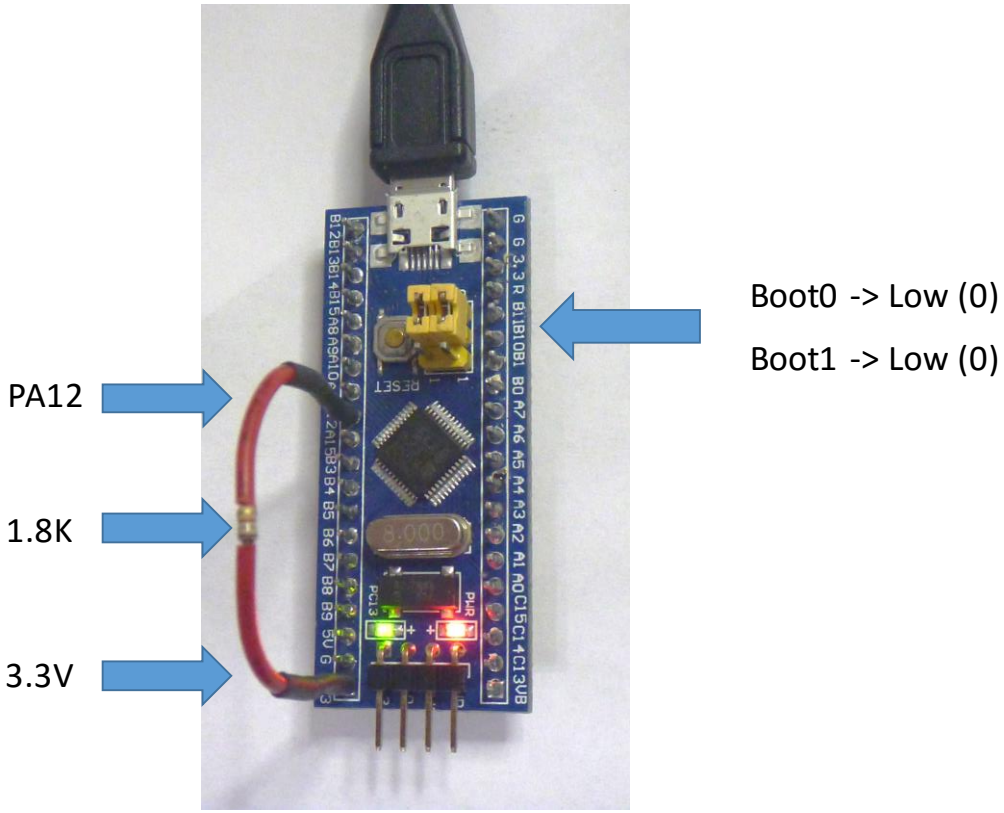
Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader

For USB-bootloader download, set Boot0 and Boot1 pins low (0),

plus add 1.8K shunt resistor between PA12 and 3.3V



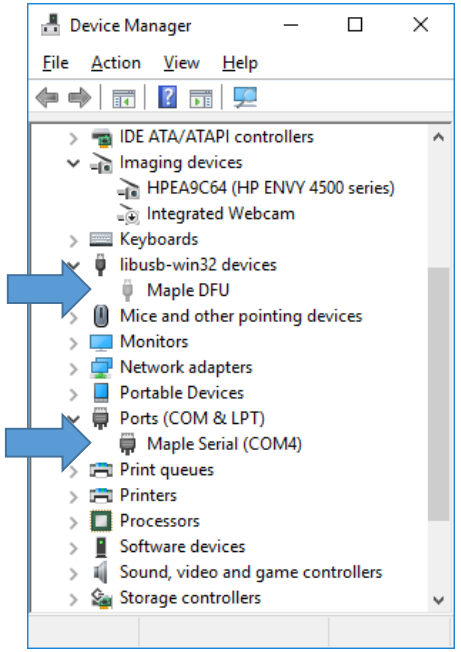
Downloading “Blink_and_Print” Sketch to Blue-Pill F103C8 using USB-Bootloader

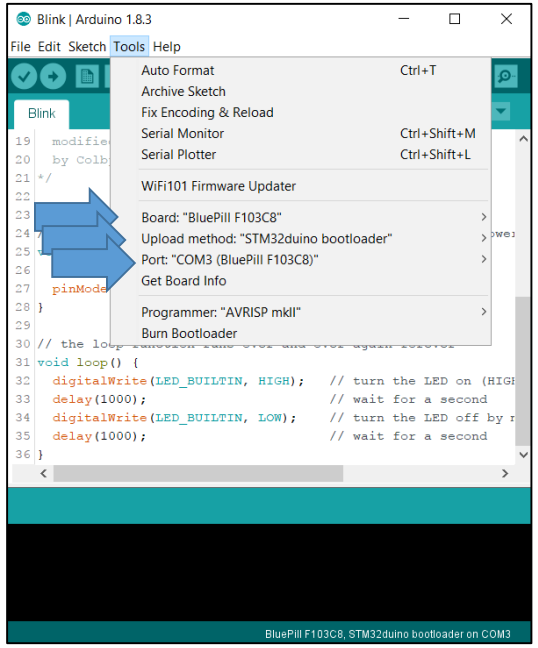
With Boot0 and Boot1 Low (0), Connect USB to PC, and observe with Device Manager*

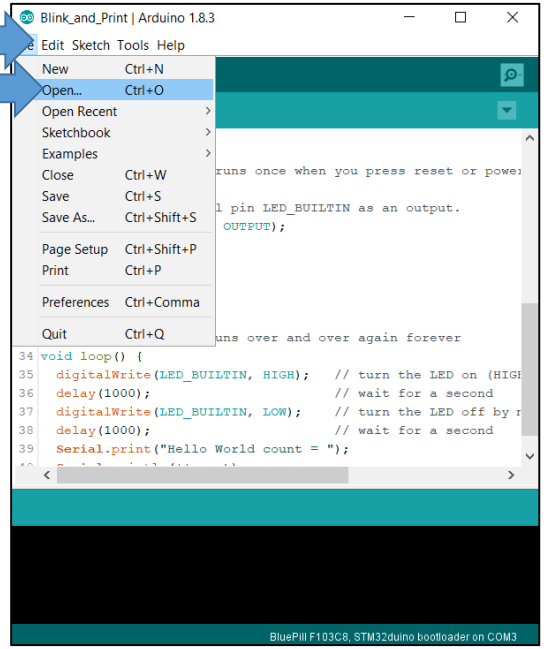
First Verify Board, Select STM32duino bootloader, and Verify/Set Port

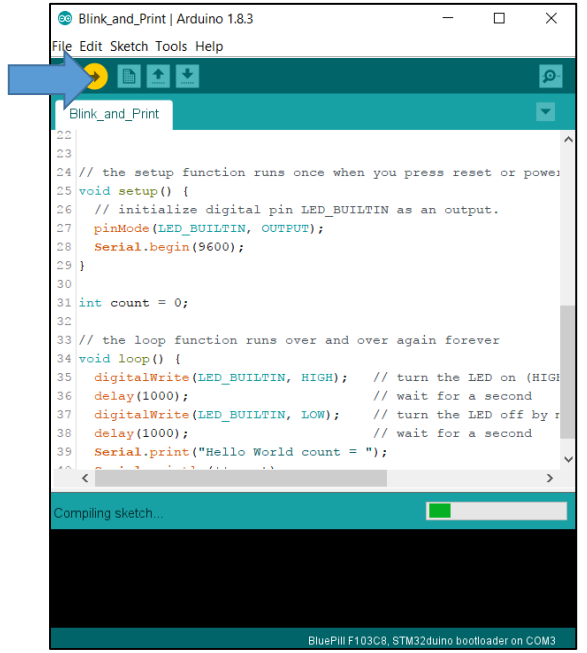
Then Open saved Blink_and_Print unless already open

Start Compile/Download









*use Show Hidden Devices

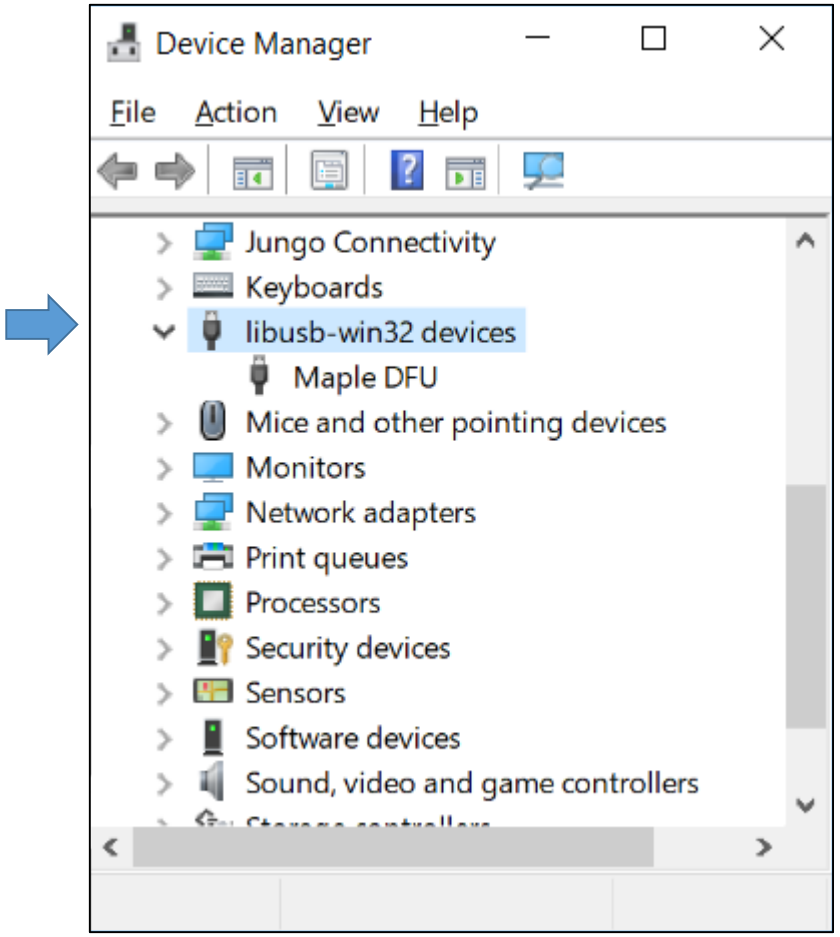
Maple Serial (COMx) only show up after the first USB bootloader download.

WARNING: If the Maple DFU shows up under “Other devices” with a question mark, the Maple DFU drivers are missing.

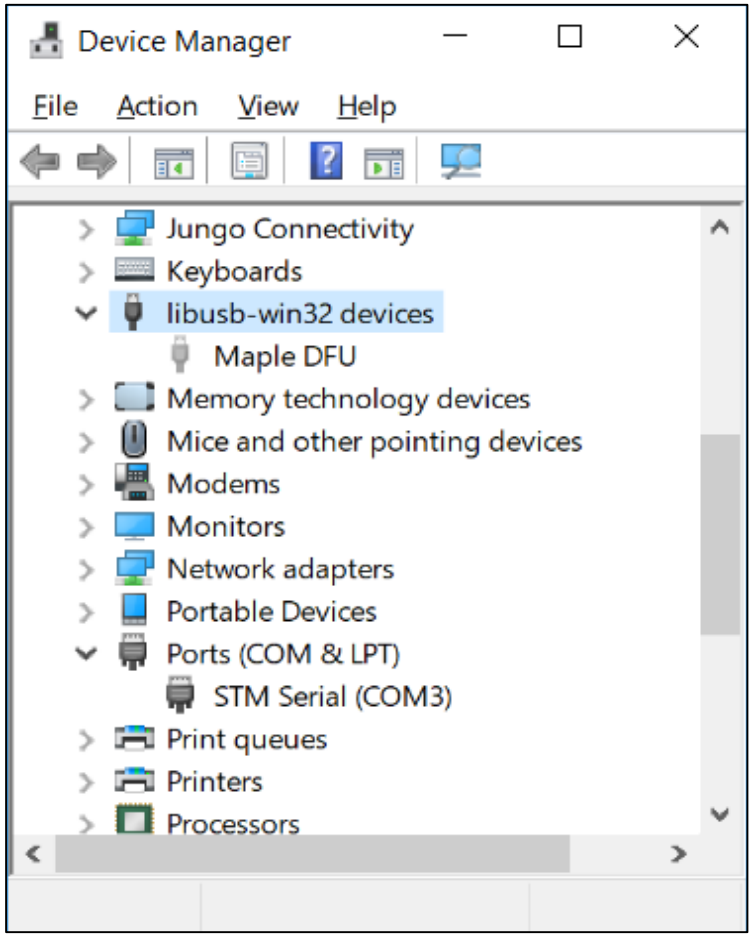
Checking STM32F103C8/Blue Pill Interface

Device Manager: View > Show hidden devices

Before first sketch download



After first sketch download



Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Updating ST-LINK Clone Firmware

(e.g. www.aliexpress.com > “st-link v2 stlink mini stm8stm32” from $1.76 with free shipping)

Updating ST-LINK Clone Firmware

- **Download and run** ST-LINK, ST-LINK/V2, ST-LINK/V2-1 USB driver signed for Windows7, Windows8, Windows10 (en.stsw-link009.zip) part number: “STSW-LINK009” from: http://www.st.com/en/embedded-software/stsw-link009.html

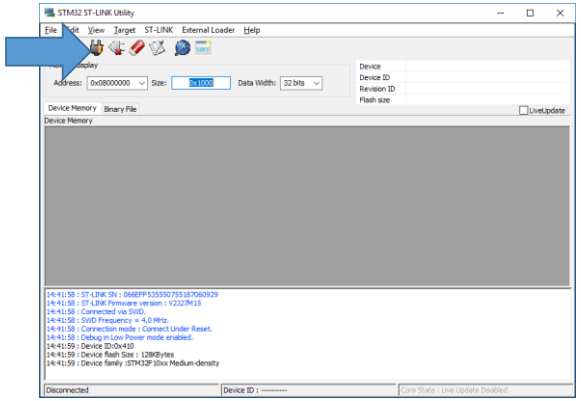
- **Download, install, and start** STM32 ST-LINK utility (en.stsw-link004.zip) part number: “STSW-LINK004” from: http://www.st.com/en/embedded-software/stsw-link004.html

- **Plug ST-LINK into PC and in STM32 ST-Link utility select** “Connect to target”

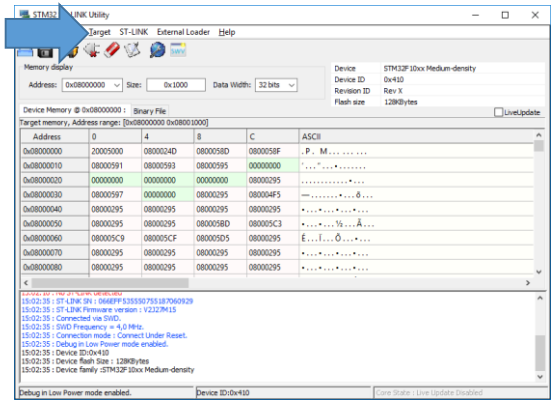
- **Update ST-Link firmware**
  - Select: “ST-LINK” > “Firmware update”
  - Select: “Device Connect”
    Firmware Version: XXXX
    Upgrade firmware to: V2.J27.M15
    Click: “Yes”

Updating ST-LINK Clone Firmware

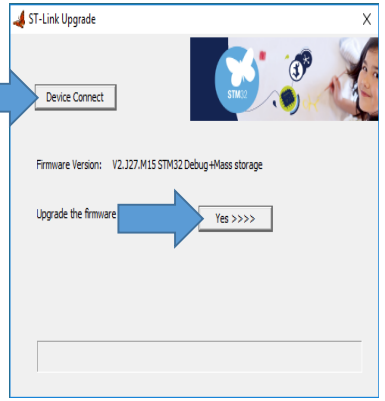
Connect to target



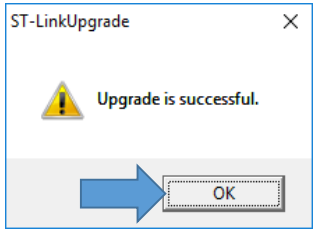
Update ST-Link firmware



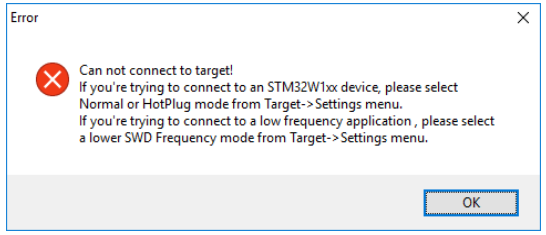
Device Connect & Yes



Accept (OK)



Bad ST-Link



Entry Level STM32/ARM Programming

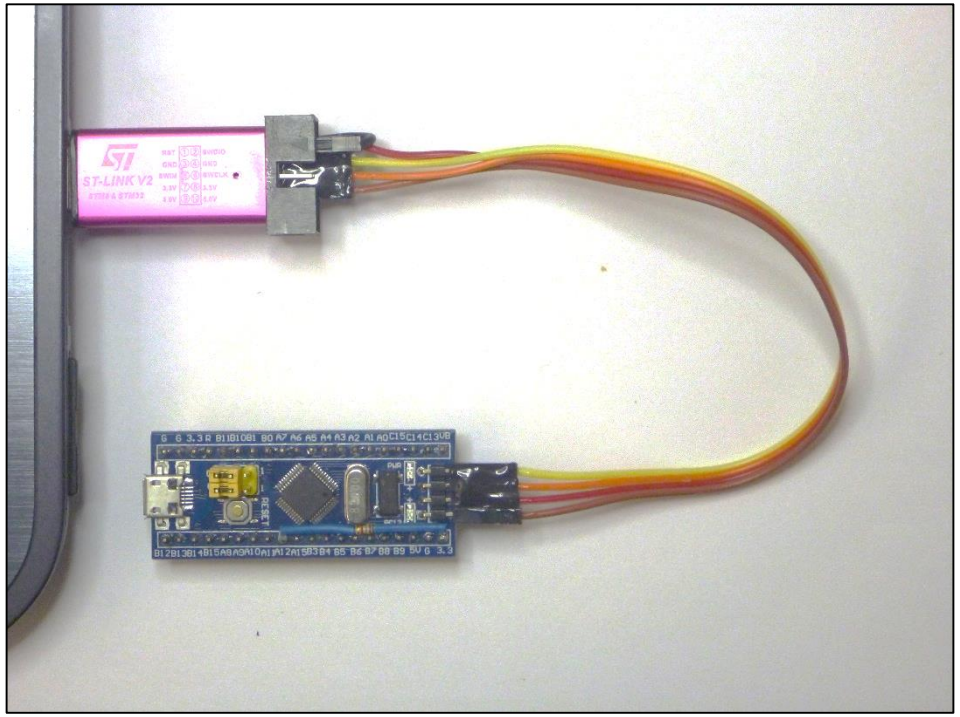
Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM Setup

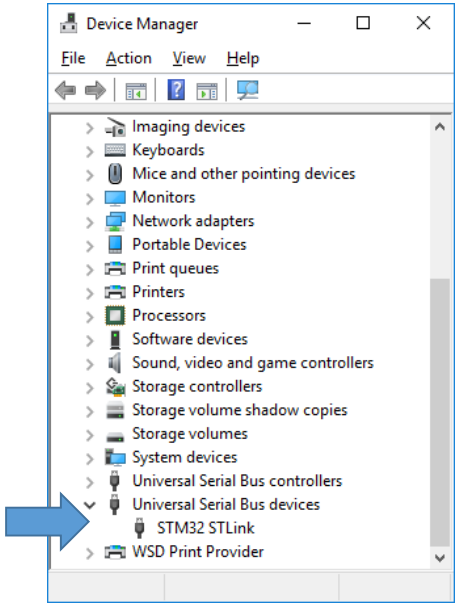
- Remove USB-to-PC cable
- Check Boot0 and Boot1 jumpers to Low (0)
- With or W/O 1.8K between 3.3V and PA12 shunting internal 10K
- Remove old Serial Port using Device Manager with Show Hidden Devices
- Connect ST-LINK Clone to Blue Pill F103C8 and PC

Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM Setup



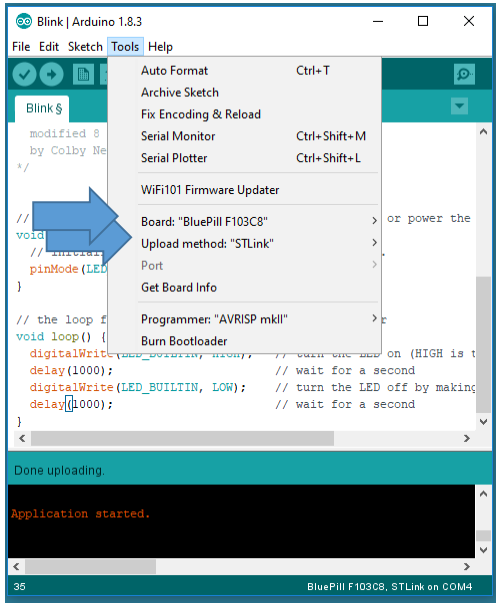
Downloading Blink Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM#

Connect Blue-Pill STM32F103C8 to ST-LINK clone, and observe with Device Manager*

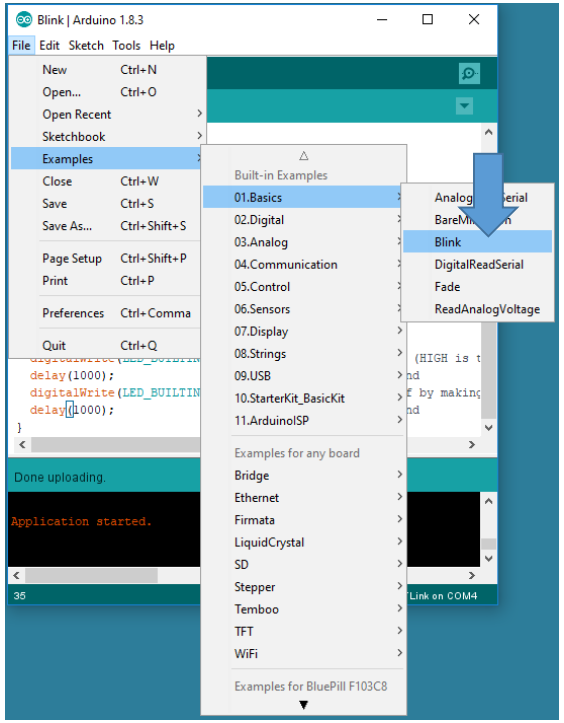


*use Show Hidden Devices to remove old devices first

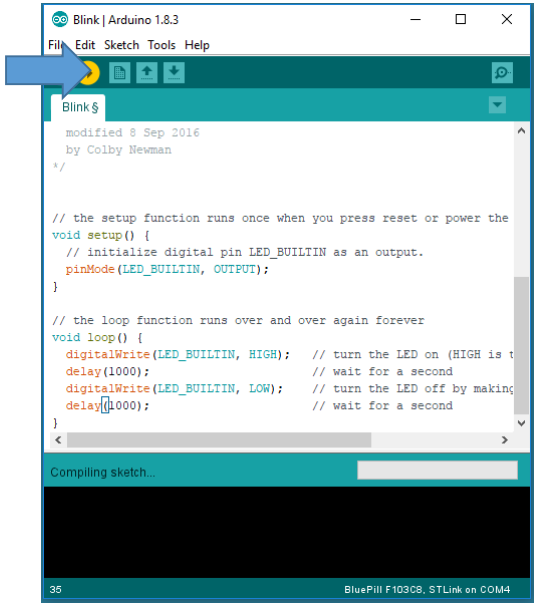
Select board, and Upload method: “STLink”



Select Sketch
File > 01.Basic > Blink



Start Compile/Download



ST-LINK Clone does not support Serial COM

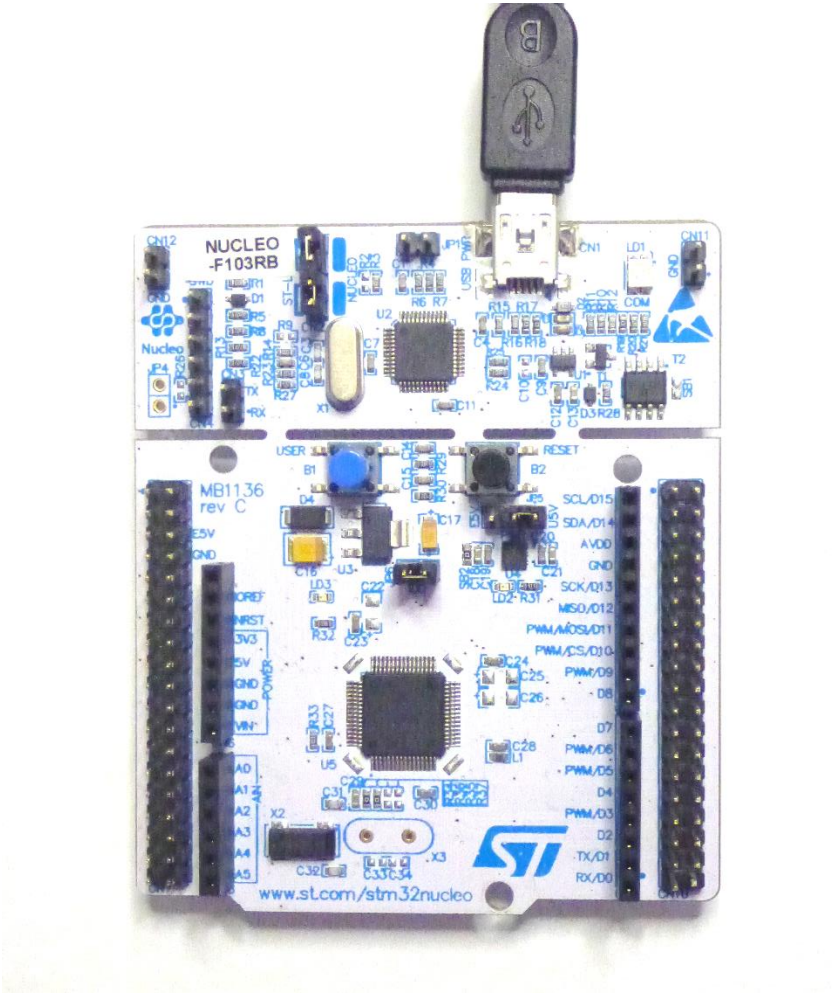
Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK clone firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo-F103RB using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control

Downloading Sketch to Nucleo-F103RB using built-in ST-LINK with virtual COM Setup

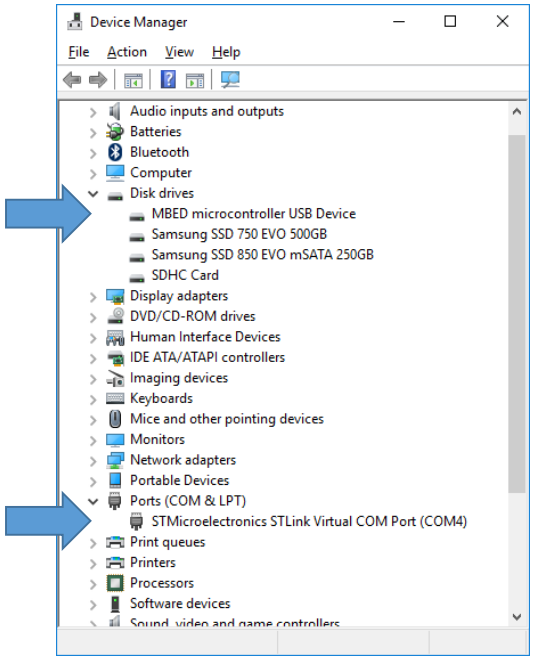
Check/Update ST-Link firmware using STM32 ST-LINK utility



Downloading Blink_and_Print Sketch to Nucleo-F103RB using built-in ST-LINK with virtual COM

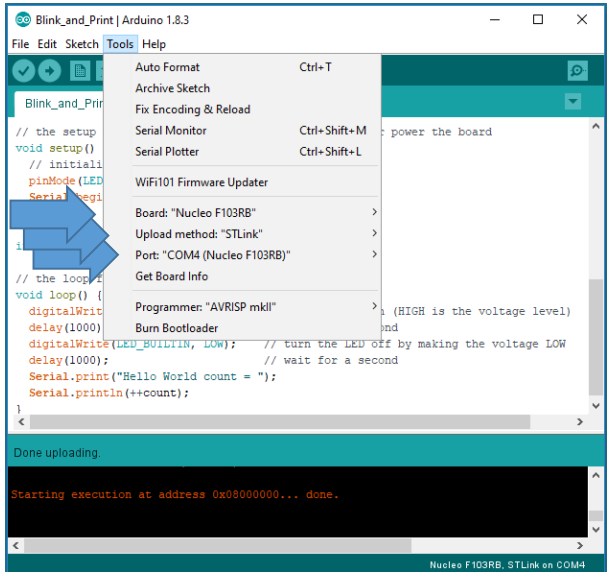
Note: Arduino IDE identifies proper UART and compiles using appropriate UART

Connect Nucleo board's ST-LINK to PC using USB cable, and observe with Device Manager*

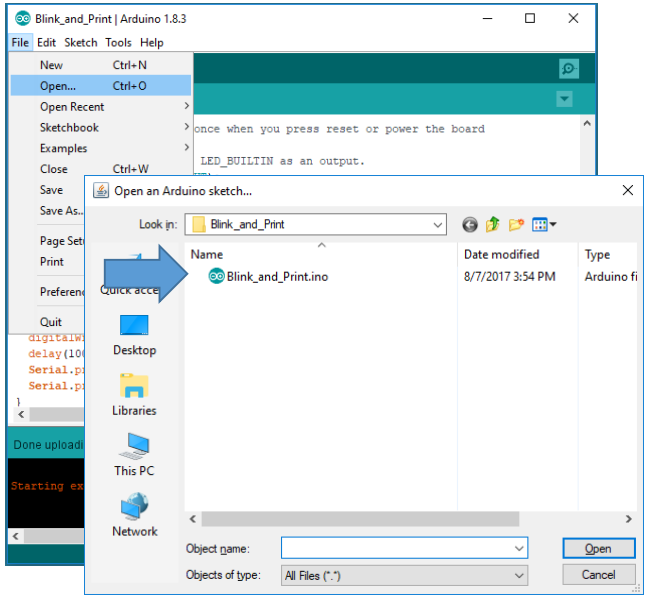


*use Show Hidden Devices to remove old devices first

Select board, Upload method: "STLink", and Port: "COM4 …"

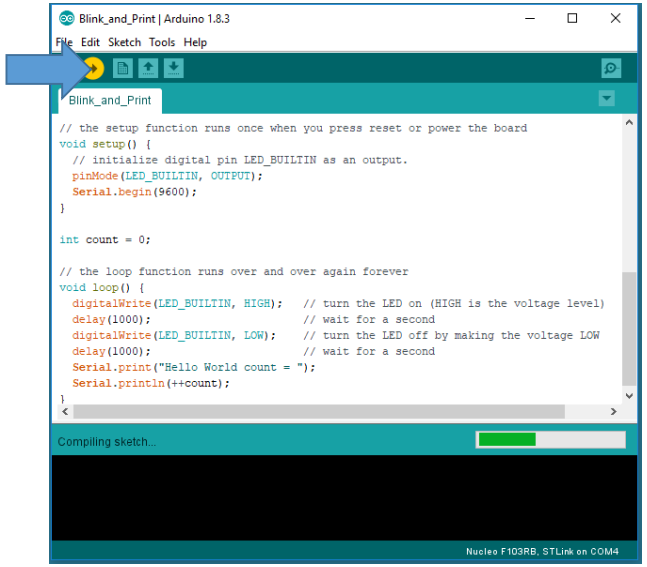


Then File > Open… and Select Blink_and_Print.ino

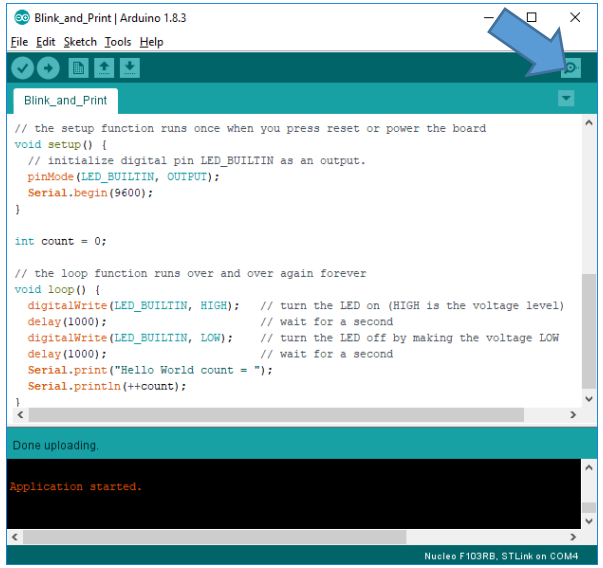


Downloading Blink_and_Print Sketch to Nucleo board using built-in ST-LINK with virtual COM

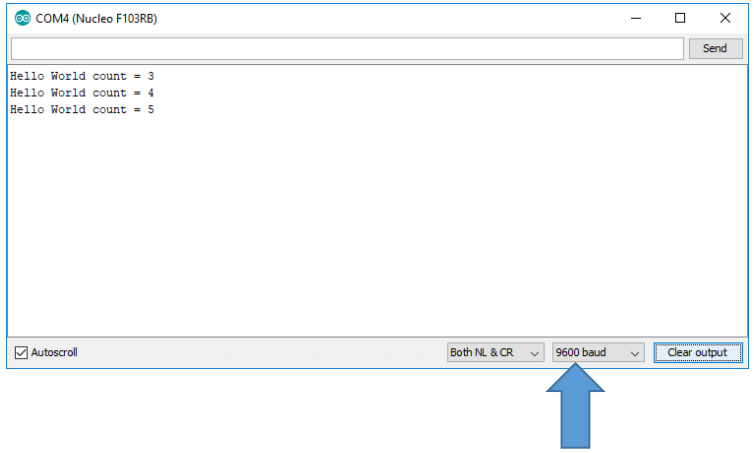
Start Compile/Download



Select Serial Monitor



Set baud rate and observe results



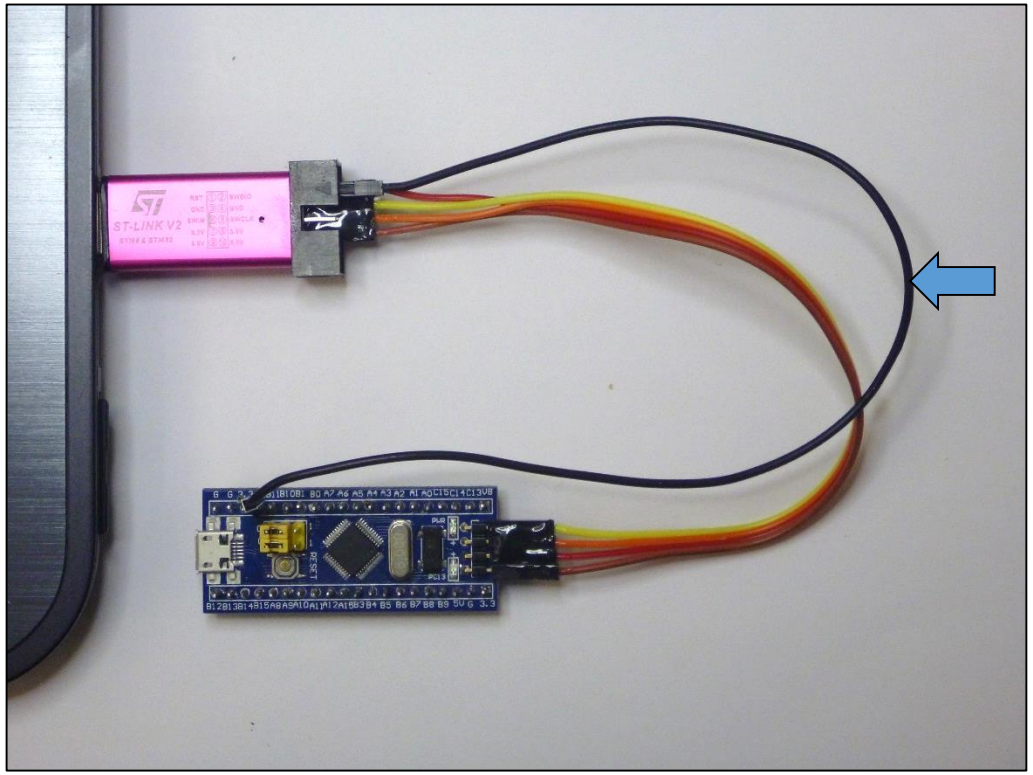
Entry Level STM32/ARM Programming

Arduino IDE with example Sketches for STM32/ARM processors

- Arduino STM32 History
- Arduino AVR and STM32/ARM Entry Level Devices
- Arduino IDE without/with STM32F1xx/STM32 Cores by ST-Microelectronics
- Installing STM32F1xx/STM32 Cores by ST-Microelectronics into Arduino IDE
- Downloading Arduino Sketch to Blue-Pill F103C8 using USB-to-Serial
- Downloading USB-Bootloader to Blue-Pill F103C8 using USB-to-Serial
- Downloading Sketch to Blue-Pill F103C8 using USB-Bootloader
- Updating ST-LINK firmware
- Downloading Sketch to Blue-Pill F103C8 using ST-LINK Clone w/o virtual COM
- Downloading Sketch to Nucleo board using built-in ST-LINK with virtual COM
- Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control wire

Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control wire

Add Reset wire from ST-LINK Clone's Reset Pin to Blue-Pill F103C8 Reset Pin

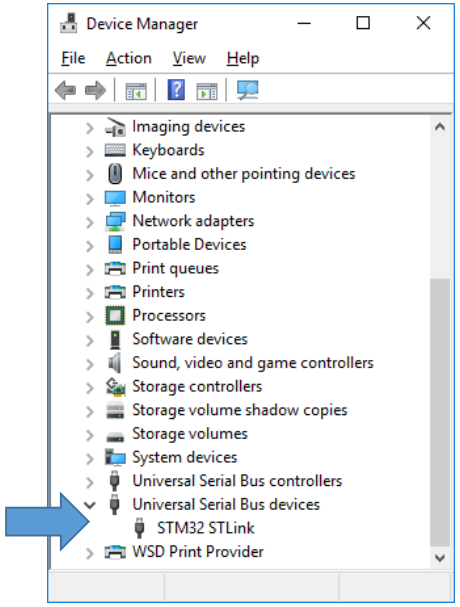


Un-Bricking Blue-Pill F103C8 using ST-LINK Clone with Reset control wire

- Check Boot0 and Boot1 jumpers to Low (0)
- With or W/O 1.8K between 3.3V and PA12 shunting internal 10K
- Remove old Serial Port using Device Manager with Show Hidden Devices
- Connect ST-LINK Clone with Reset control wire to Blue Pill F103C8 and PC

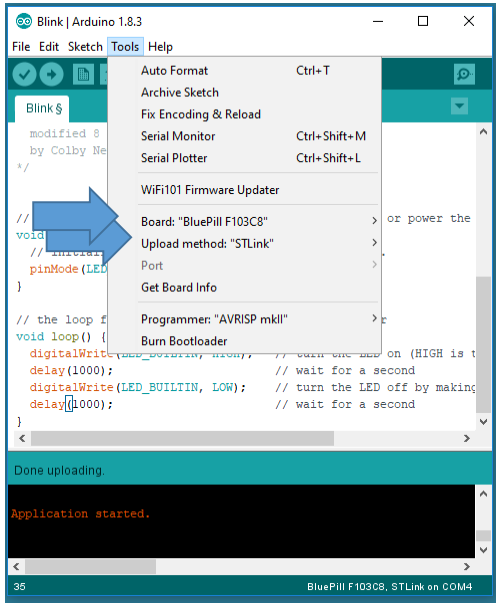
Un-Bricking Blue-Pill F103C8 using ST-LINK Clone# with Reset control wire

Connect Blue-Pill STM32F103C8 to ST-LINK clone, and observe with Device Manager*

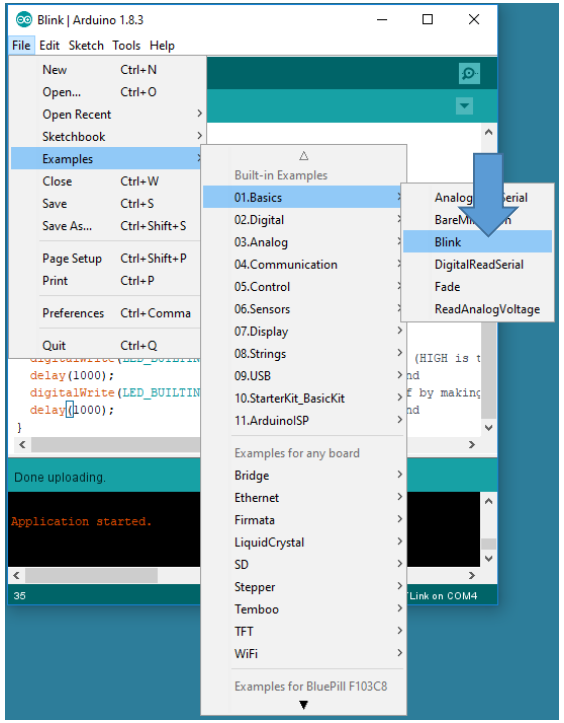


*use Show Hidden Devices to remove old devices first

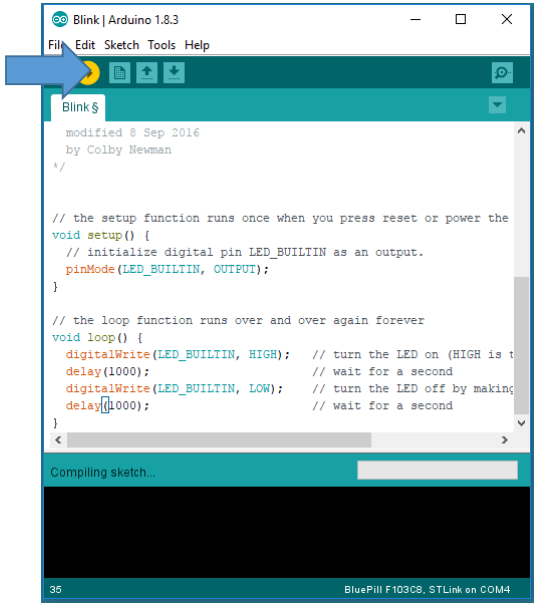
Select board, and Upload method: “STLink”



Select Sketch
File > 01.Basic > Blink



Start Compile/Download



Note: ST-LINK Clone doesn’t support Serial COM

STM32/ARM Programming Complexity Levels

- Entry Level STM32/ARM Programming (today's presentation)

  Arduino IDE using Sketch examples for STM32/ARM processors

- Intermediate Level STM32/ARM Programming (today's presentation)

  Arduino IDE beyond Sketch examples for STM32/ARM processors

- Advanced Level STM32/ARM Programming (potential future presentation)

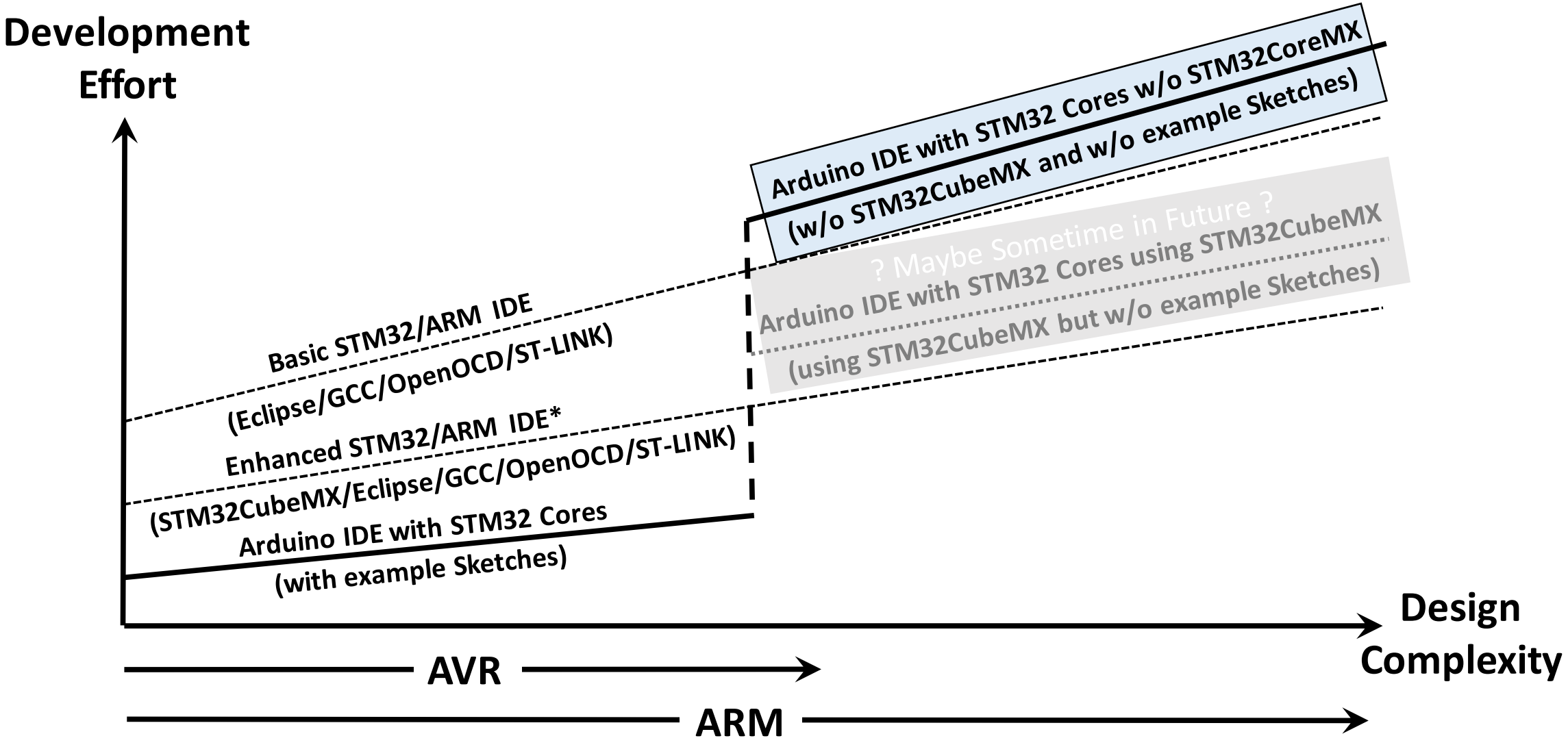
  STM32CubeMX/Eclipse/GCC/ST-LINK programming STM32/ARM processors

- Expert Level STM32/ARM Programming (beyond my paygrade)

  Adding/Integrating new STM32/ARM processor/board to Arduino IDE

  (reference: Add a new variant (board) · stm32duino/wiki Wiki …)

Arduino IDE vs STM32/ARM IDEs



*e.g. Atollic TrueSTUDIO

Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

Arduino IDE ***with*** STM32F1xx/STM32 Cores* by ST-Microelectronics

STM32F1xx Boards

- **Nucleo-F103RB** **128KB 20KB 72MHz**
- **STM32VL Discovery STM32F100RB** **128KB 8KB 24MHz**
- **Blue Pill STM32F103C8** **64KB 20KB 72MHz**
- **MapleMini STM32F103CB** **128KB 20KB 72MHz**

AVR Boards

- Pro Mini 16KB 2KB 16MHz
- Uno R3 16KB 2KB 16MHz

Entry Level
Show-and-Tell

STM32 Boards (with ST-Link)

- Nucleo 144
  - Nucleo-F207ZG 1024KB 128KB 120MHz
  - **Nucleo-F429ZI** **2048KB 256KB 180MHz**
- Nucleo 64
  - Nucleo-F030R8 64KB 8KB 48MHz
  - Nucleo-F091RC 256KB 32KB 48MHz
  - **Nucleo-F103RB** **128KB 20KB 72MHz**
  - Nucleo-F303RB 128KB 40KB 72MHz
  - **Nucleo-F401RE** **512KB 96KB 84MHz**
  - **Nucleo-F411RE (DPRG alt CPU)** **512KB 128KB 100MHz**
  - Nucleo-L053R8 64KB 8KB 32MHz
  - Nucleo-L152RE 512KB 80KB 32MHz
  - **Nucleo-L476RG** **1024KB 128KB 80MHz**
- Nucleo 32
  - Nucleo-L432KC 256KB 64KB 80MHz
- Discovery
  - **STM32F100RB-DISCVL** **128KB 8KB 24MHz**
  - STM32F407G-DISC1 1024KB 192KB 168MHz
  - **STM32F746G-DISCOVERY** **1024KB 340KB 216MHz**

* ST-Microelectronics is in the process of integrating Blue Pill and MapleMini into single STM32 Boards package

Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals

- **Download** ST-Microelectronics' Medium-density performance line ARM®-based 32-bit MCU STM32F103x8/STM32F103xB – Datasheet from:

  http://www.st.com/content/ccc/resource/technical/document/datasheet/33/d4/6f/1d/df/0b/4c/6d/CD00161566.pdf/files/CD00161566.pdf/jcr:content/translations/en.CD00161566.pdf

- **Download** ST-Microelectronics' STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM®-based 32-bit MCUs - Reference Manual – RM0008 from:

  http://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf

- **Download** ST-Microelectronics' Description of STM32F1 HAL and Low-Layer Drivers - User manual – UM1850 from:

  http://www.st.com/content/ccc/resource/technical/document/user_manual/72/52/cc/53/05/e3/4c/98/DM00154093.pdf/files/DM00154093.pdf/jcr:content/translations/en.DM00154093.pdf

Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

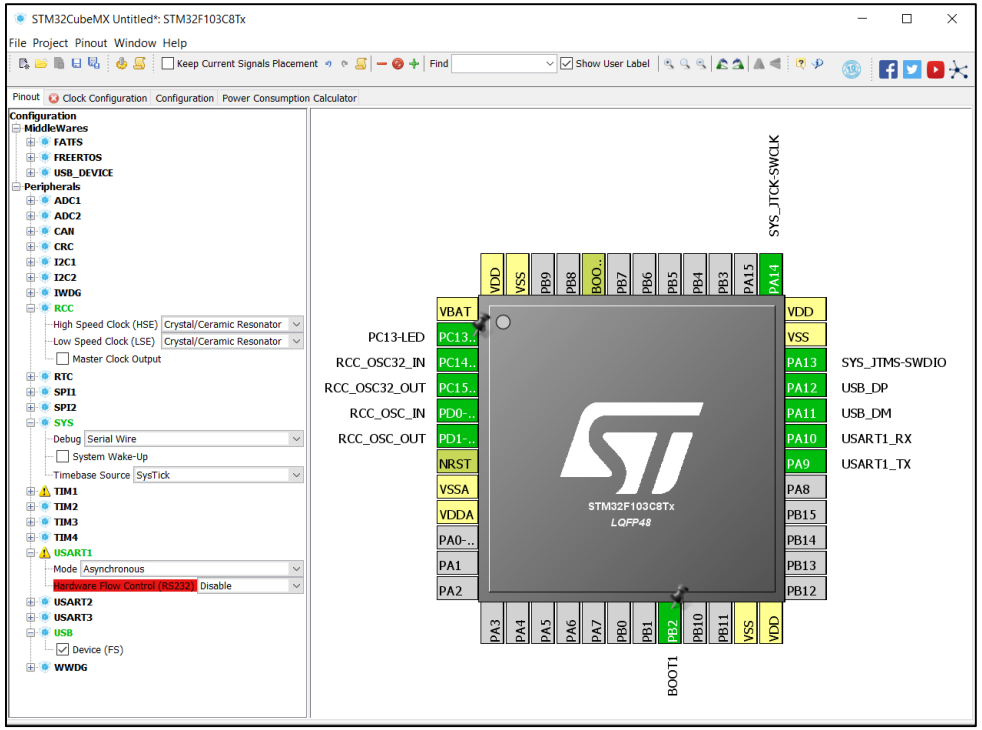
Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill

- **Download and Install** ST-Microelectronics' STM32CubeMX v4.22.0 (en.stm32cubemx.zip) part number: STM32CubeMX from:

  http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html

- **Open STM32F103C8/Blue Pill in STM32CubeMX with Function Selects and Pin Labels**
  - Select "New Project"
  - In "MCU Selector", enter "STM32F103C8" in the search field and select STM32F103C8"
  - In "Pinout" select:
    - "RCC" with Crystal/Ceramic Resonator for High and Low Speed Clocks
    - "SYS" with Serial Wire Debug and leave SysTick as Timebase Source
    - "USART1" with Asynchronous Mode (used for serial downloading)
    - "USB" with Device (FS) (used for USB bootloader downloading)
  - In "Package" select pin:
    - "PB2" as "GPIO_Input and "Enter User Label" = BOOT1
    - "PC13" as "GPIO_Output" and "Enter User Label" = PC13-LED

STM32CubeMX Opened With STM32F103C8/Blue Pill



- Create Folder C:/STM32duino/Workspace
- Select Project > Settings (Project Settings) > Project Location > Browse to above Folder
- Add Project Name “STM32F103C8 BluePill” and click OK
- Select File > Save Project and close

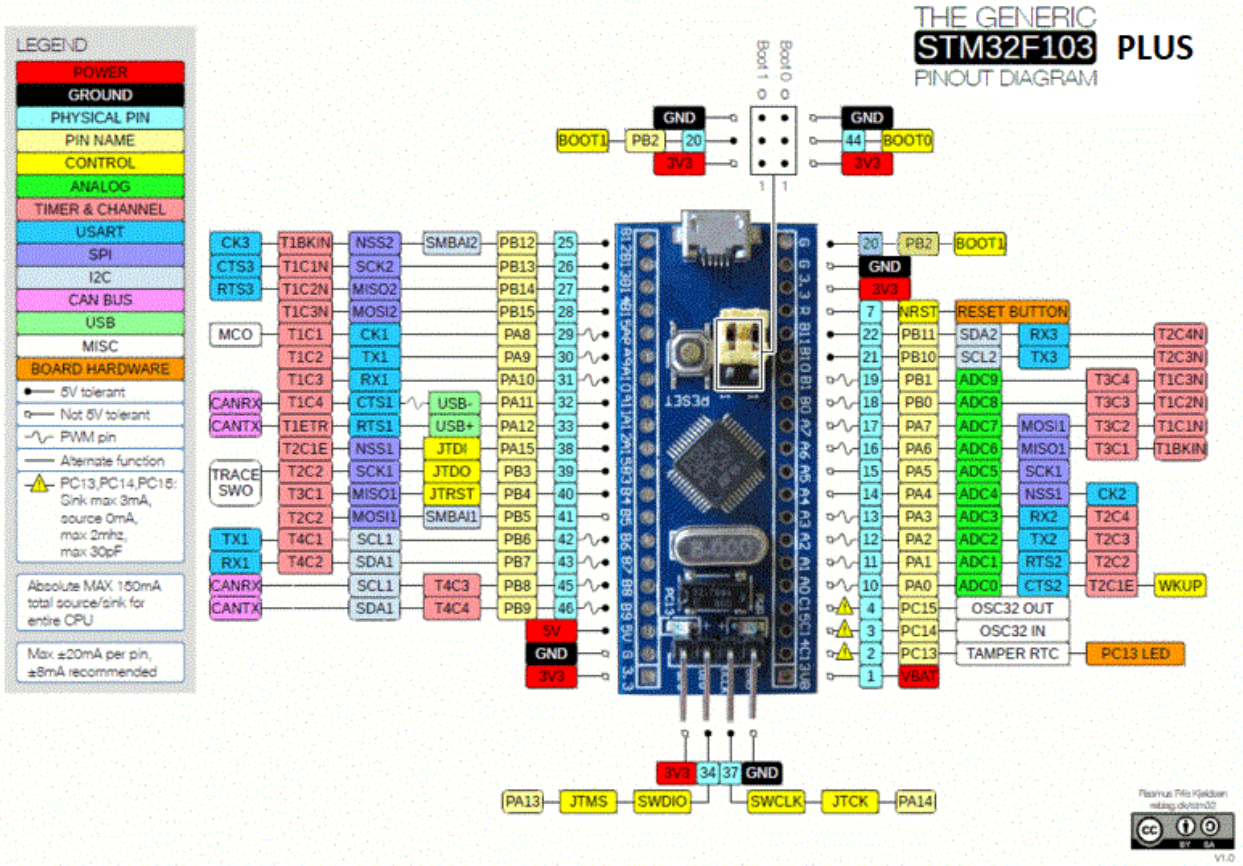
Intermediate Level STM32/ARM Programming

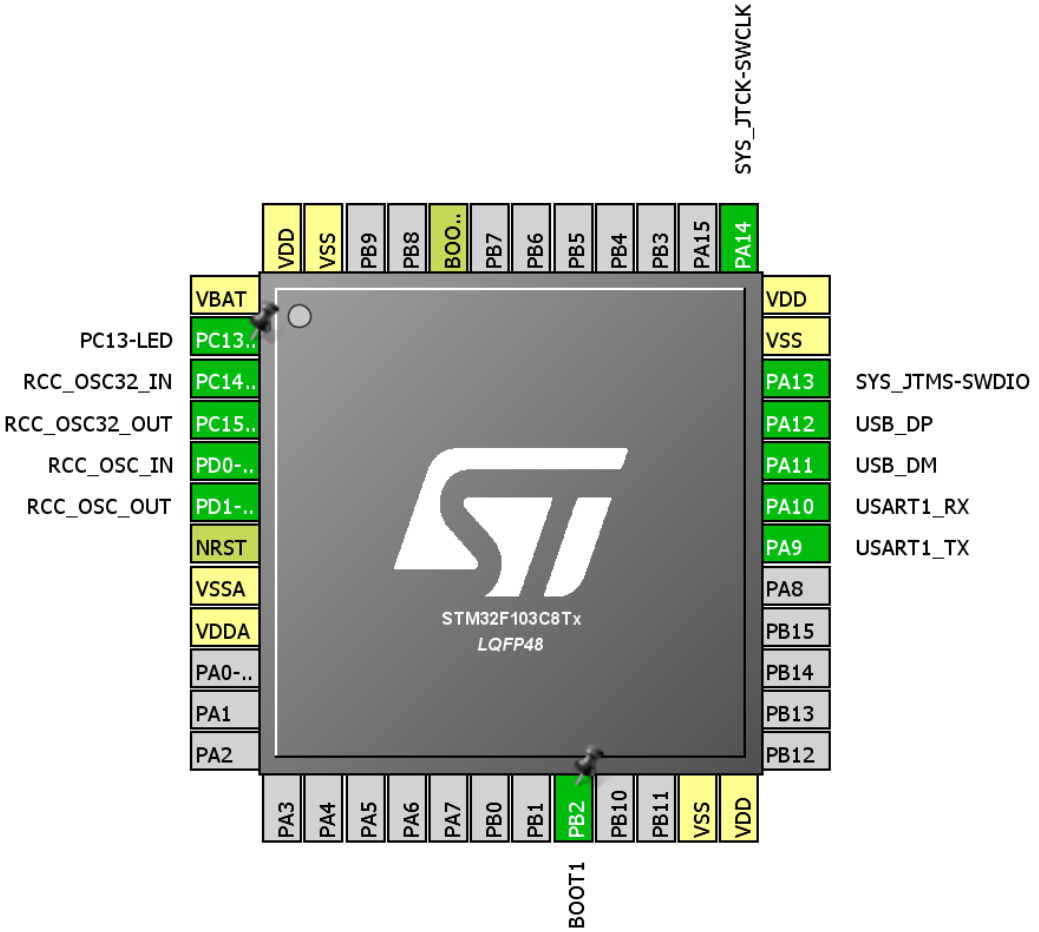
Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32F103C8 Pinout vs STM32F103C8 in STM32CubeMX





Intermediate Level STM32/ARM Programming

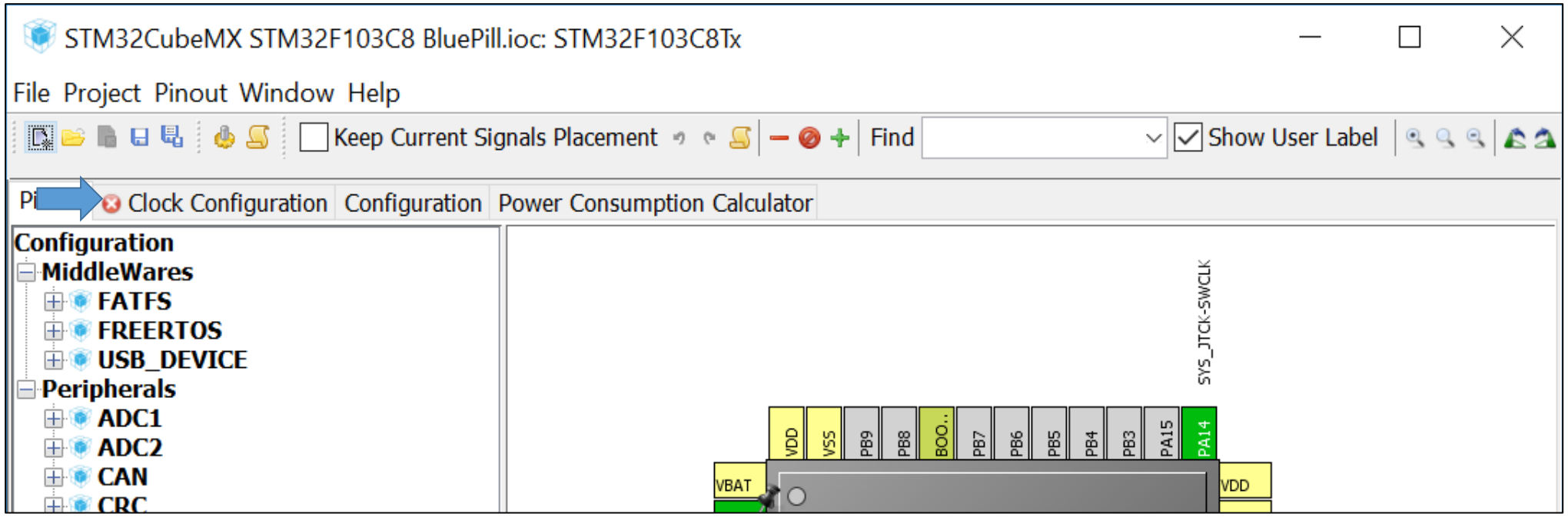
Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

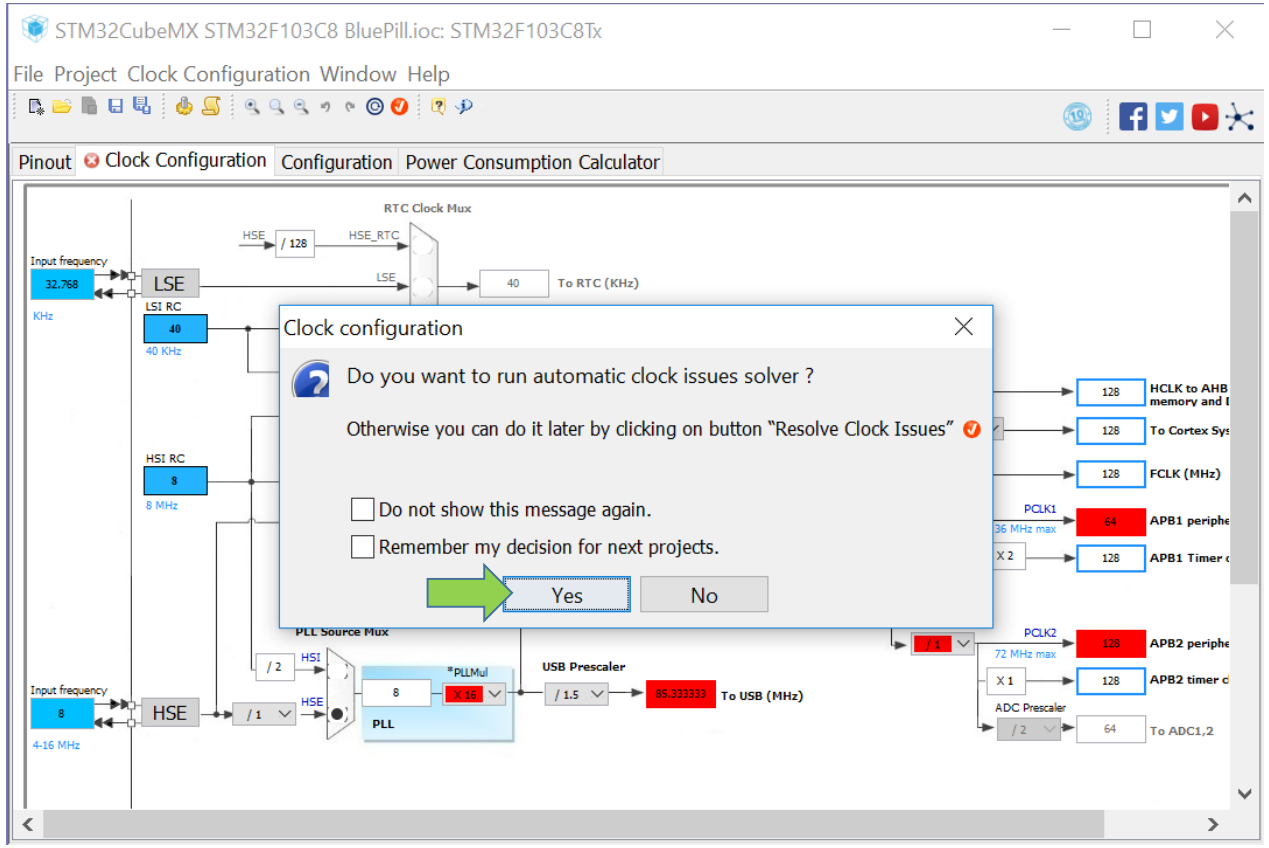
STM32/ARM Programming Blink using STM32CubeMX to Generate Code

- Open STM32CubeMX
- Select “Load Project”
- Select STM32F103C8 BluePill.ioc and Open
- *Notice:* Clock Configuration has an error flag



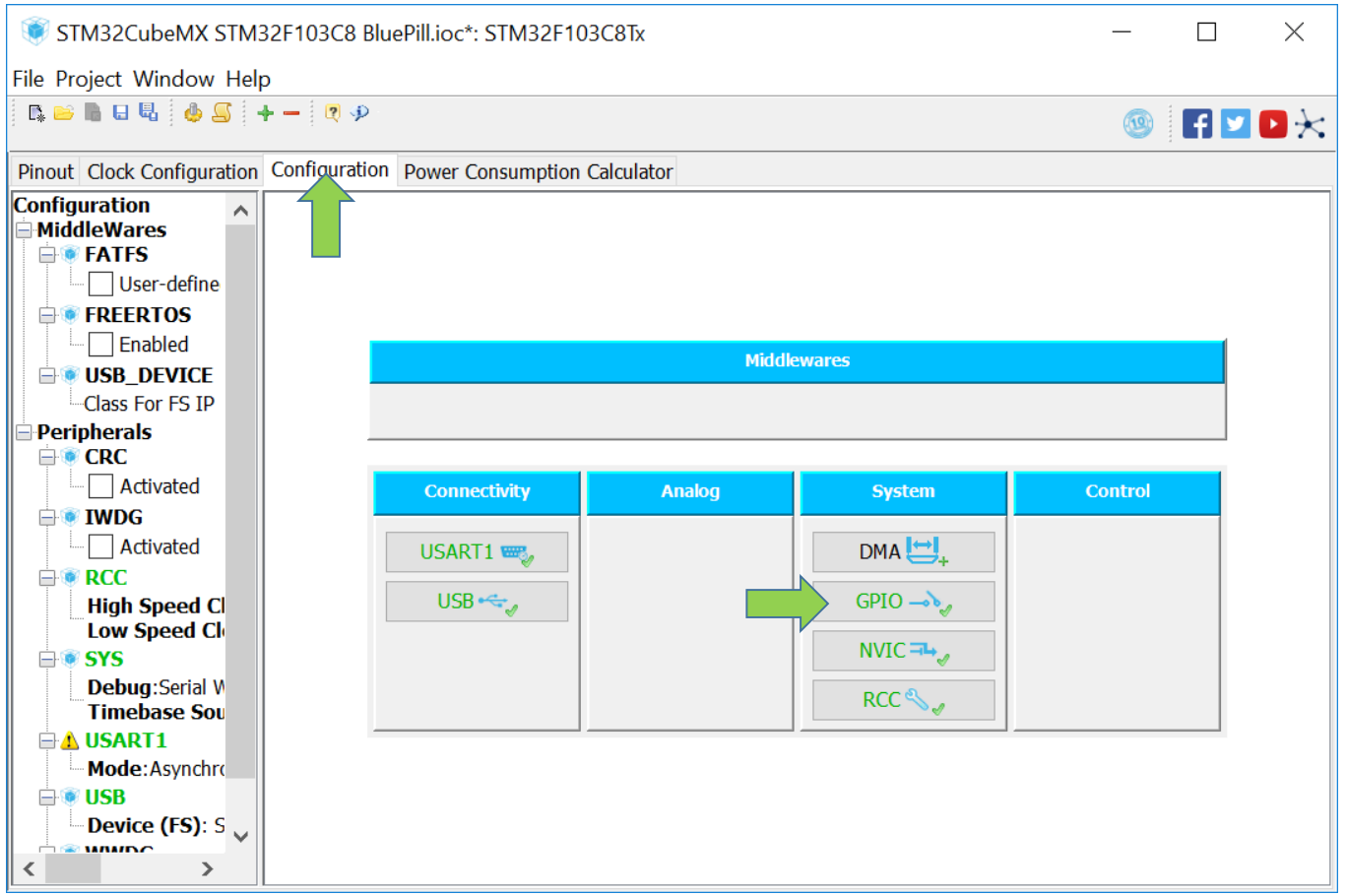
STM32/ARM Programming Blink using STM32CubeMX to Generate Code

- Click/open Clock Configuration
- Answer Yes to “Do you want to run automatic clock issues solver” and *notice* fixed (no red)



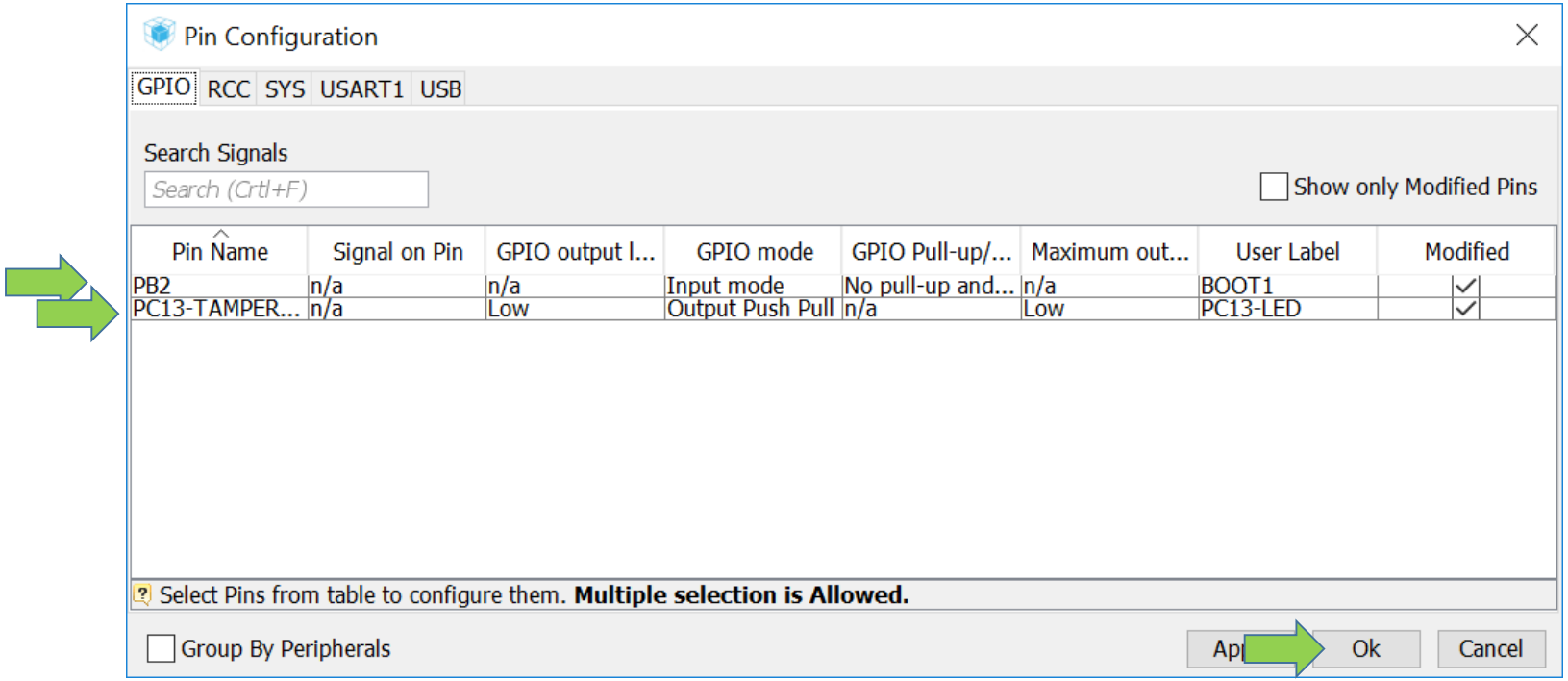
STM32/ARM Programming Blink using STM32CubeMX to Generate Code

- Select Configuration and then GPIO



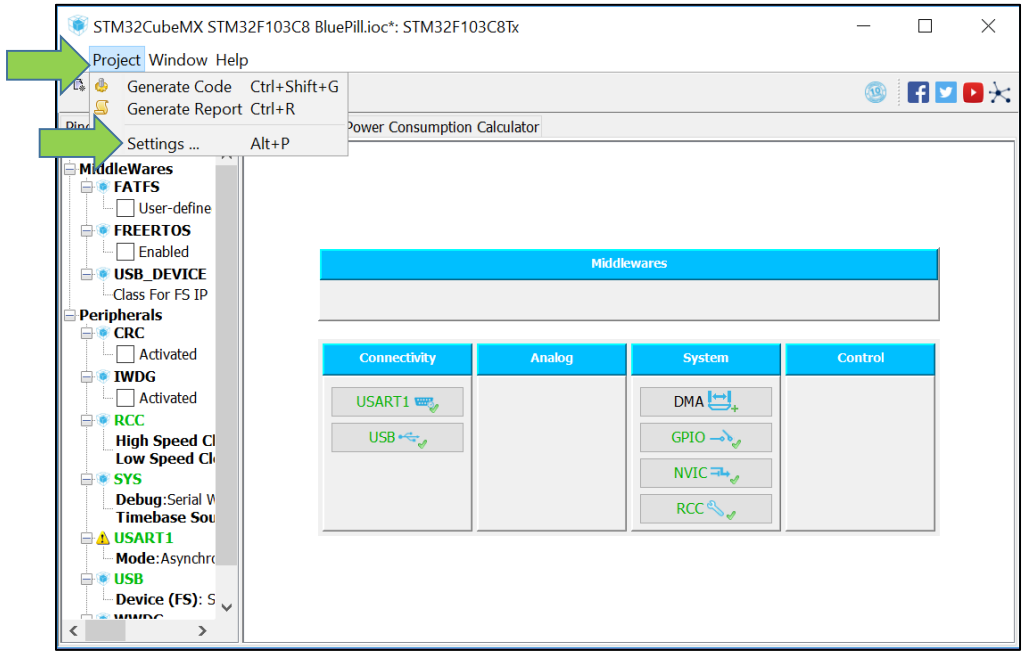
STM32/ARM Programming Blink using STM32CubeMX to Generate Code

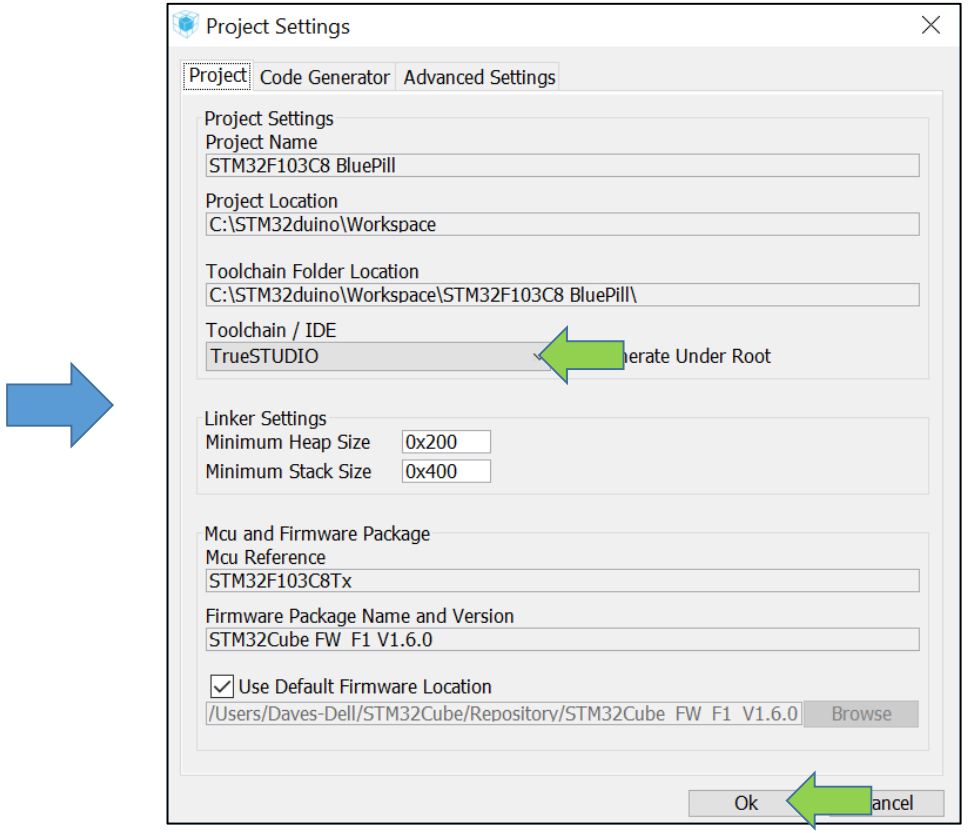
- In Pin Configuration panel and *notice:*
  - PB2 already has Input mode and User Label: BOOT1
  - PC13 already has Output Push Pull and User Label: PC13-LED
- Click OK to close Pin Configuration panel



STM32/ARM Programming Blink using STM32CubeMX to Generate Code

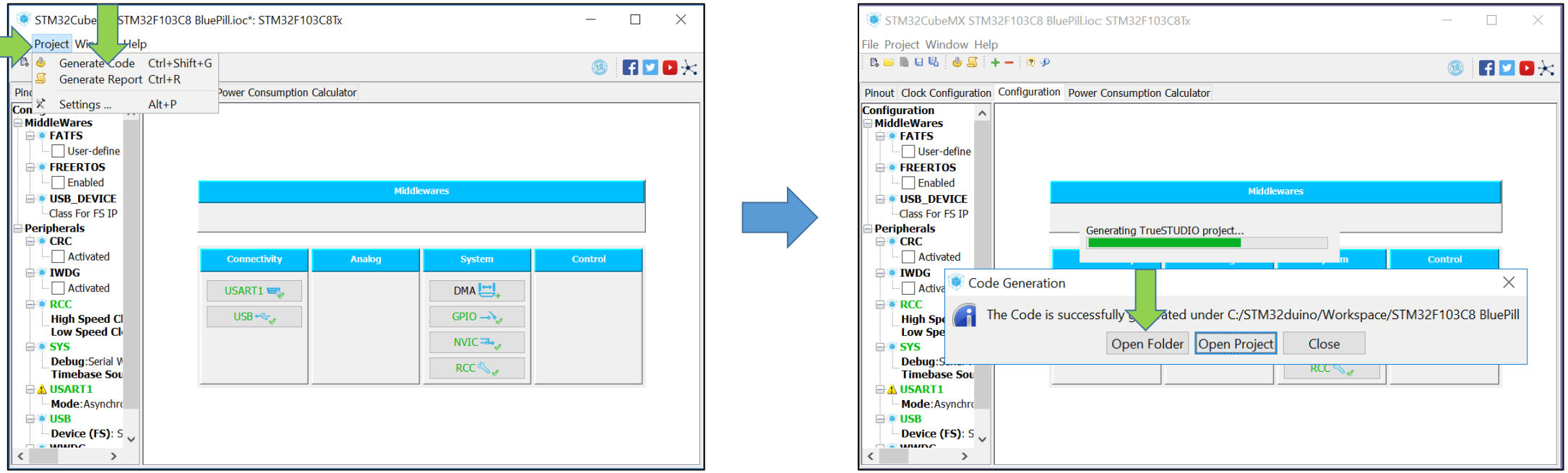
- Select Project > Settings (Project Settings) > Toolchain/IDE > TrueSUDIO and click OK





STM32/ARM Programming Blink using STM32CubeMX to Generate Code

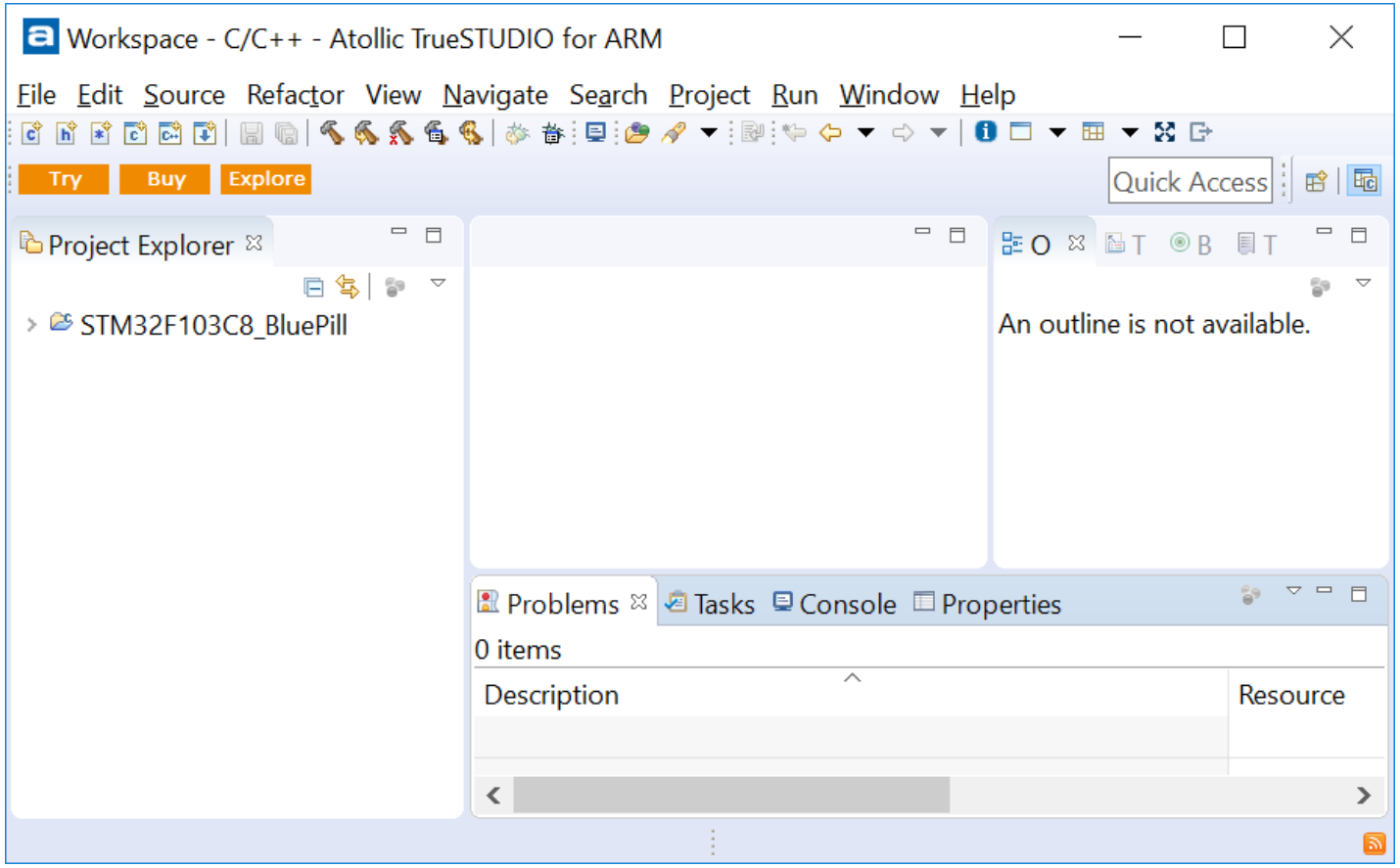
- Select Project > Generate Code and select Open Folder* (to use File Explorer to view files)



* Note: If you already have Atollic's TrueSTUDIO installed and If TrueSTUDIO's Workspace is already set to C:\STM32duino\Workspace, then clicking ***Open Project*** and OK will open TrueSTUDIO with the STM32F103C8 BluePill directory showing up in TrueSTUDIO's Project Explorer.

STM32/ARM Programming Blink using STM32CubeMX to Generate Code

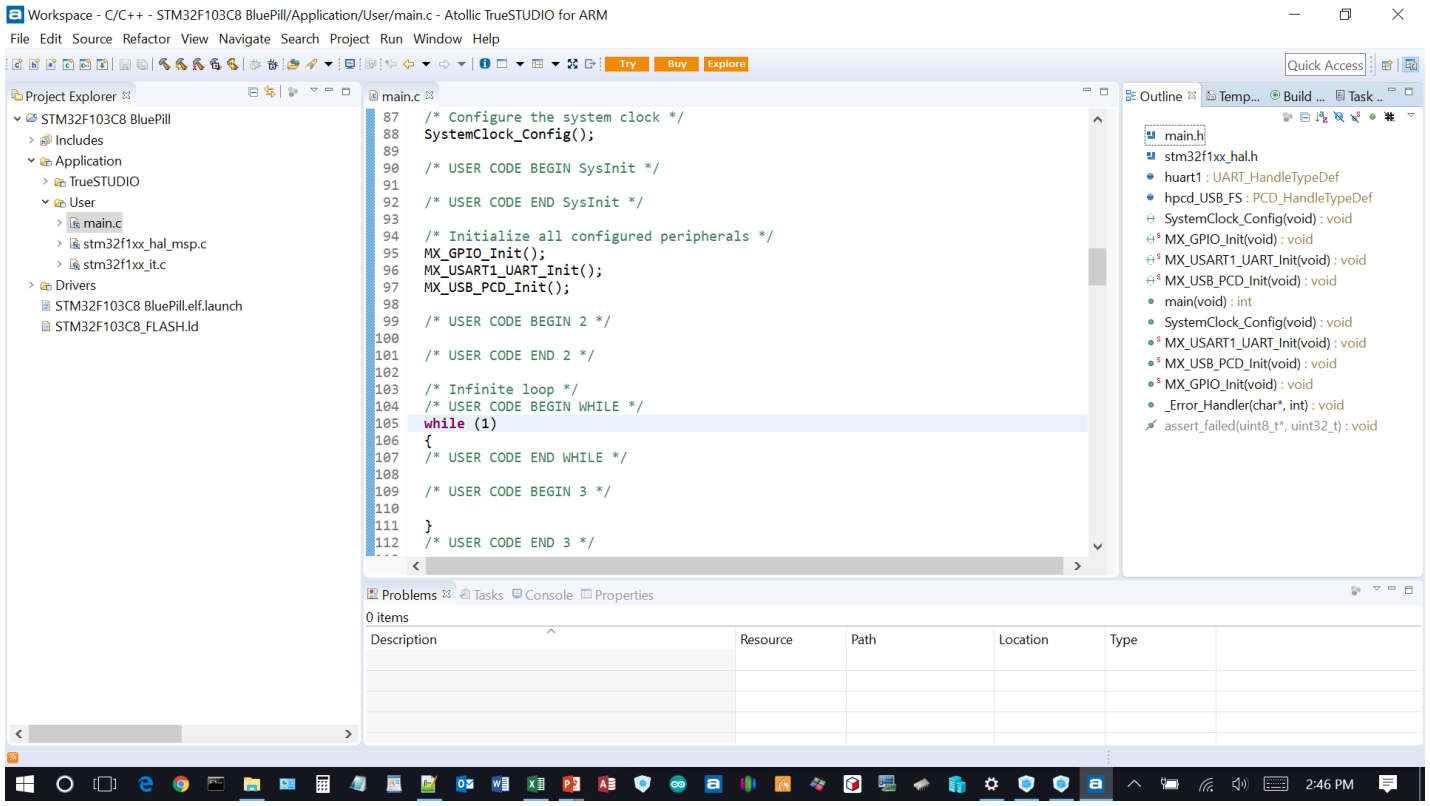
(STM32CubeMX > Generate Code > **Open Project** - *assuming TrueSTUDIO installed and workspace setup*)



STM32/ARM Programming Blink using STM32CubeMX to Generate Code

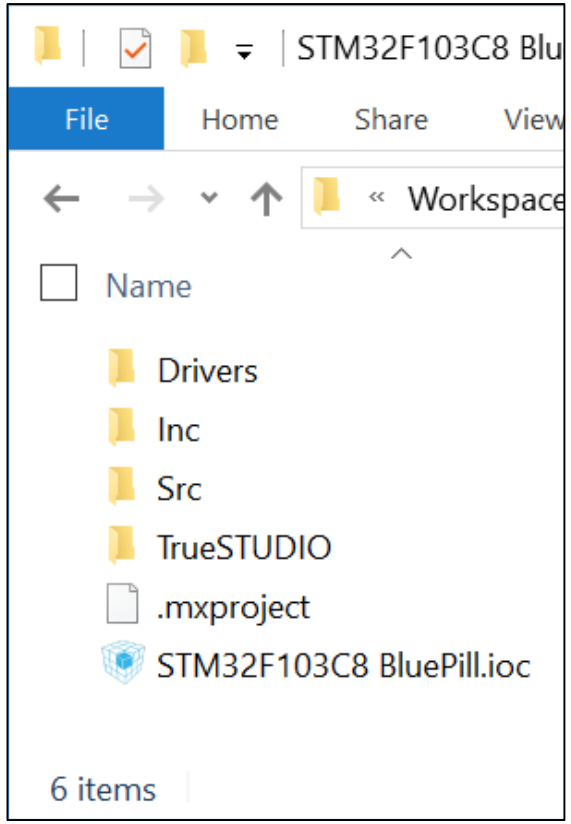
(In TrueSTUDIO: opening STM31F103C8 BluePill > Application > User > main.c)

Note: This skeleton of main.c was generated by STM32CubeMX to initialize the STM31F103C8 BluePill we defined in STM32CubeMX *and is ready for the user to add their code between the /* USER CODE BEGIN XXX */ and /* USER CODE END XXX */ tokens.*



STM32/ARM Programming Blink using STM32CubeMX to Generate Code

(STM32CubeMX > Generate Code > Open Folder - opens File Explorer)



- Drivers
  - CMSIS and STM32F1xx_HAL_Driver standard definitions
- Inc
  - **main.h – LED pin/port definitions**
  - stm32f1xx_hal_conf.h – HAL configuration defines
  - stm32f1xx_it.h – interrupt handler prototypes
- Src
  - main.c
    - calls to and high level code for HAL and SystemClock initialization routines
    - **MX_GPIO_Init() LED pin/port initialization/configuration routine**
    - While(1) loop
  - stm32f1xx_hal_msp.c – NVIC interrupt routine mapping
  - stm32f1xx_it.c – NVIC interrupt handler routines
  - system_stm32f1xx.c – System clock initialization routine
- TrueSTUDIO - project settings and linker script/command files
- .mxproject – STM32CubeMX's design file/directory information
- STM32F103C8 BluePill.ioc – STM32CubeMX's design specifics

Suggest installing "Notepad++" (https://notepad-plus-plus.org/) for viewing code files.

STM32/ARM Programming Blink using STM32CubeMX

LED pin/port initialization code Generated by STM32CubeMX

main.h

```
#ifndef __MAIN_H
#define __MAIN_H

#define LED_Pin GPIO_PIN_13
#define LED_GPIO_Port GPIOC

void _Error_Handler(char *, int);

#define Error_Handler() _Error_Handler(__FILE__, __LINE__)

#endif /* __MAIN_H */
```

NOTE: STM32CubeMX enables GPIOC clock before writing to GPIOC

NOTE: This is how HAL writes pin values GPIO_PIN_SET/RESET

Code unique to this application

main.c

```
int main(void) {
        HAL_Init();
        SystemClock_Config();
        MX_GPIO_Init();
         while (1) {
          /* USER CODE */
          }
}
static void MX_GPIO_Init(void) {
          /* GPIO Ports Init Structure */
          GPIO_InitTypeDef GPIO_InitStruct;
          /* GPIO Ports Clock Enable */
          __HAL_RCC_GPIOC_CLK_ENABLE();
          /*Configure GPIO pin Output Level */
          HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
          /*Configure GPIO pin : LED_Pin */
          GPIO_InitStruct.Pin = LED_Pin;
          GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
          GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
          HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);
}
```

STM32/ARM Programming Blink using STM32CubeMX

LED pin/port HAL Initialization Code Generated by STM32CubeMX as Arduino Sketch

```
/*
   STM32CubeMX and HAL Blink Blue Pill F103C8
   Turns on an LED on for one second, then off for one second, repeatedly.
   LED pin/port initialization code Generated by STM32CubeMX plus HAL I/O
   substituted for Arduino routines
*/

// LED pin/port definitions generated by STM32CubeMX
#define LED_Pin GPIO_PIN_13                    // STM32CubeMX
#define LED_GPIO_Port GPIOC                    // STM32CubeMX

// MX_GPIO_Init prototype generated by STM32CubeMX
static void MX_GPIO_Init(void);                // STM32CubeMX

// the setup function runs once when you press reset or power the board
void setup() {
   // initialize digital pin LED_BUILTIN as an output.
   // pinMode(LED_BUILTIN, OUTPUT);           // Arduino
   MX_GPIO_Init();                            // STM32CubeMX
}
```

STM32/ARM Programming Blink using STM32CubeMX

LED pin/port HAL Loop Code Hand Generated as Arduino Sketch

NOTE: This is how HAL writes pin values GPIO_PIN_SET/RESET

NOTE: This is how HAL writes pin values GPIO_PIN_SET/RESET

```
// the loop function runs over and over again forever
void loop() {
  // turn the LED on (HIGH is the voltage level)
  // digitalWrite(LED_BUILTIN, HIGH);                                    // Arduino
  HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);    // HAL
  delay(1000);                                  // wait for a second
  // turn the LED off by making the voltage LOW
  //digitalWrite(LED_BUILTIN, LOW);                                     // Arduino
  HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET); // HAL
  delay(1000);                                  // wait for a second
}
```

STM32/ARM Programming Blink using STM32CubeMX

LED pin/port HAL Initialization Code Generated by STM32CubeMX as Arduino Sketch

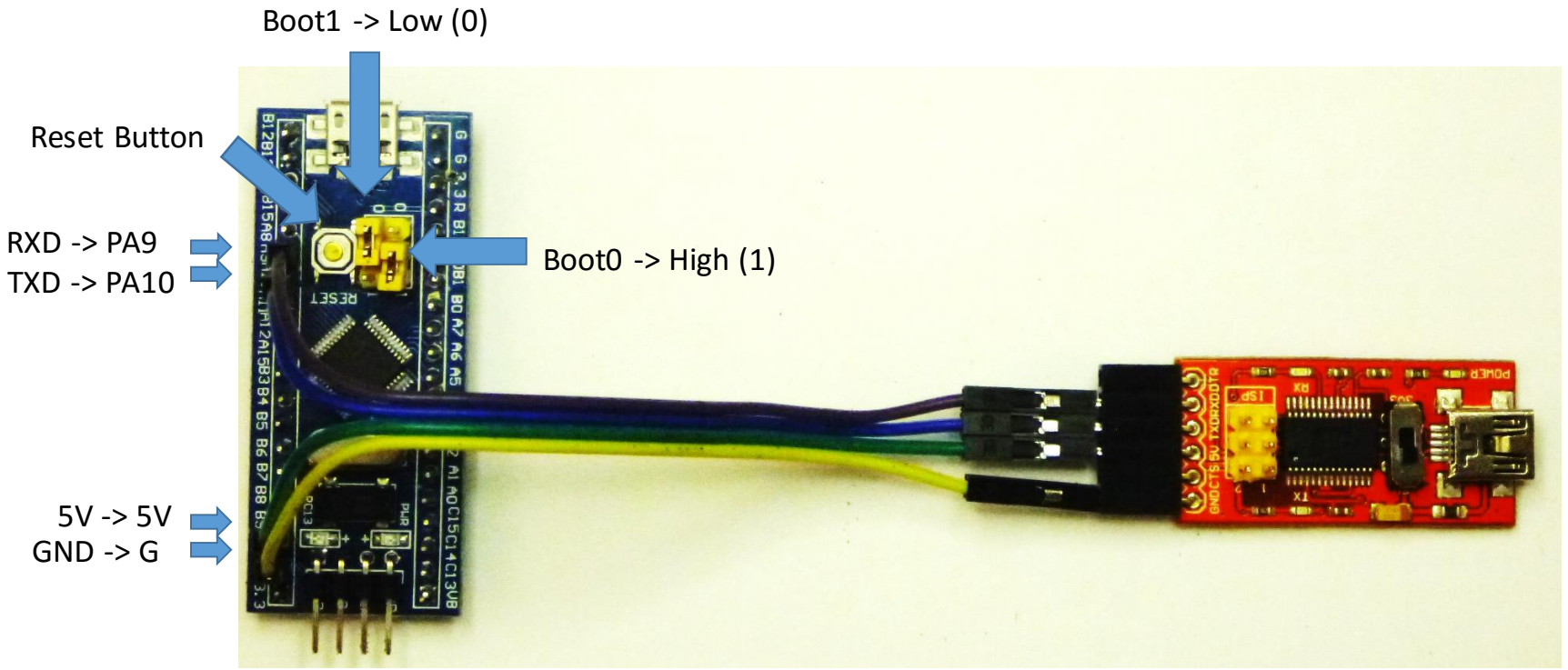
NOTE: STM32CubeMX enables GPIOC clock before writing to GPIOC

NOTE: This is how HAL writes pin Initial values GPIO_PIN_SET/RESET

```
// MX_GPIO_Init routine generated by STM32CubeMX
static void MX_GPIO_Init(void) {
   /* GPIO Ports Init Structure */
   GPIO_InitTypeDef GPIO_InitStruct;
   /* GPIO Ports Clock Enable */
   __HAL_RCC_GPIOC_CLK_ENABLE();
   /*Configure GPIO pin Output Level */
   HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
   /*Configure GPIO pin : LED_Pin */
   GPIO_InitStruct.Pin = LED_Pin;
   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
   HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);
}
```

STM32/ARM Programming Blink using STM32CubeMX

Downloading STM32CubeMX plus HAL I/O Sketch to Blue-Pill F103C8 using USB-to-Serial



Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32/ARM Programming Blink using STM32F10xxx Reference Manual

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.

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

Bit 4 **IOPCEN:** IO port C clock enable
Set and cleared by software.
0: IO port C clock disabled
1: IO port C clock enabled

Bit 3 **IOPBEN:** IO port B clock enable
Set and cleared by software.
0: IO port B clock disabled
1: IO port B clock enabled

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

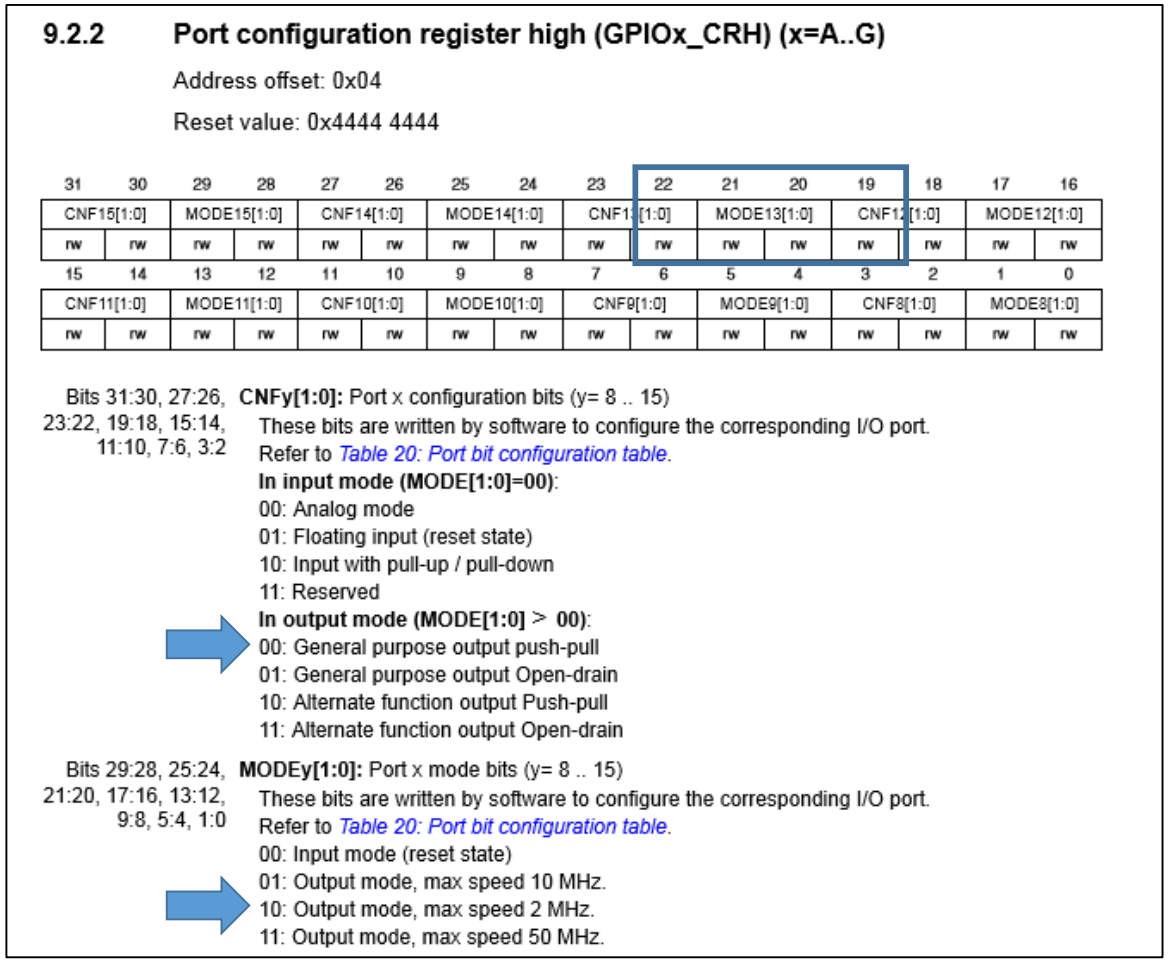
Bit 1 Reserved, must be kept at reset value.

Bit 0 **AFIOEN:** Alternate function IO clock enable
Set and cleared by software.
0: Alternate Function IO clock disabled
1: Alternate Function IO clock enabled

Each IO port has its own clock and clock enable bit.

```
// enable Port C clock
RCC->APB2ENR |= 0x0010;
```

STM32/ARM Programming Blink using STM32F10xxx Reference Manual



9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

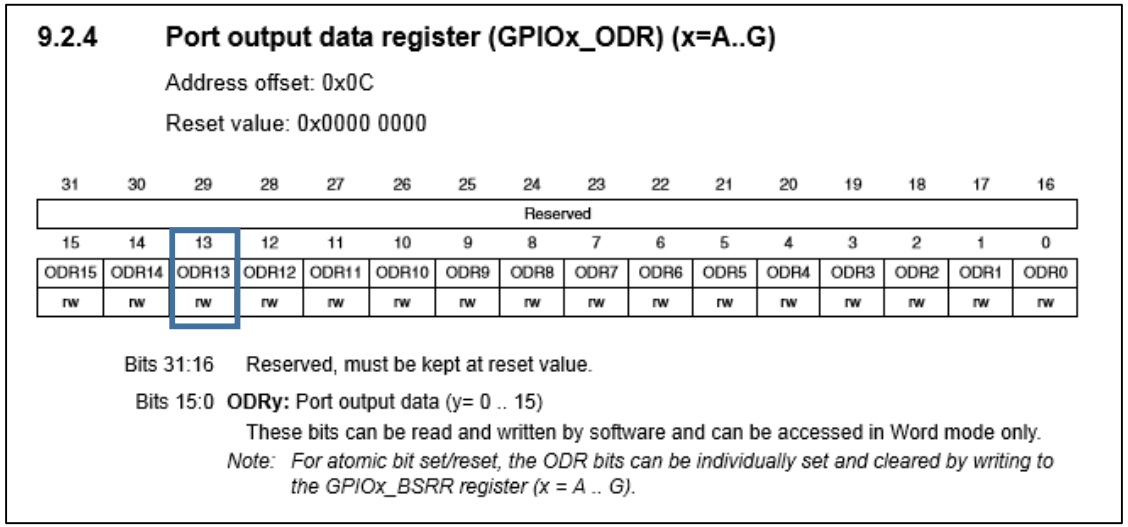
setup

```
// clear Port C Bit 13's output configuration and mode
GPIOC->CRH &= ~ 0X00F00000;
// set Port C Bit 13's output configuration and mode
GPIOC->CRH |=  0X00200000; // output push-pull slow
```

***Alternate* setup**

```
// clear/set Port C Bit 13's output configuration and mode
GPIOC->CRH =  (GPIOC->CRH & ~ 0X00F00000) |
                                        0X00200000;
```

STM32/ARM Programming Blink using STM32F10xxx Reference Manual



9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).

loop

```
GPIOC->ODR &= ~ 0X00002000; // clear Port C Bit 13's output
HAL_Delay(1000);            // wait for a second
GPIOC-> ODR |= 0X00002000;  // set Port C Bit 13's output
HAL_Delay(1000);            // wait for a second
```

STM32/ARM Programming Blink using STM32F10xxx Reference and HAL Manuals

LED PC13 pin/port Initialization Code Hand Generated

```
/*
   STM32F10xxx Reference and HAL Hand Coded Blink Blue Pill F103C8
   Turns on an LED on for one second, then off for one second, repeatedly.
   LED PC13 pin/port Initialization and Loop Code Generated by Hand based
   on STM32F10xxx Reference and HAL Manuals substituted for Arduino routines
*/

// the setup function runs once when you press reset or power the board
void setup() {
   // initialize digital pin LED_BUILTIN as an output.
   // pinMode(LED_BUILTIN, OUTPUT);                    // Arduino
   // enable Port C clock                              // Hand Generated
   RCC->APB2ENR |= 0x0010;
   // clear/set Port C Bit 13's output configuration and mode
   GPIOC->CRH =  (GPIOC->CRH & ~ 0X00F00000)  | 0X00200000;
}
```

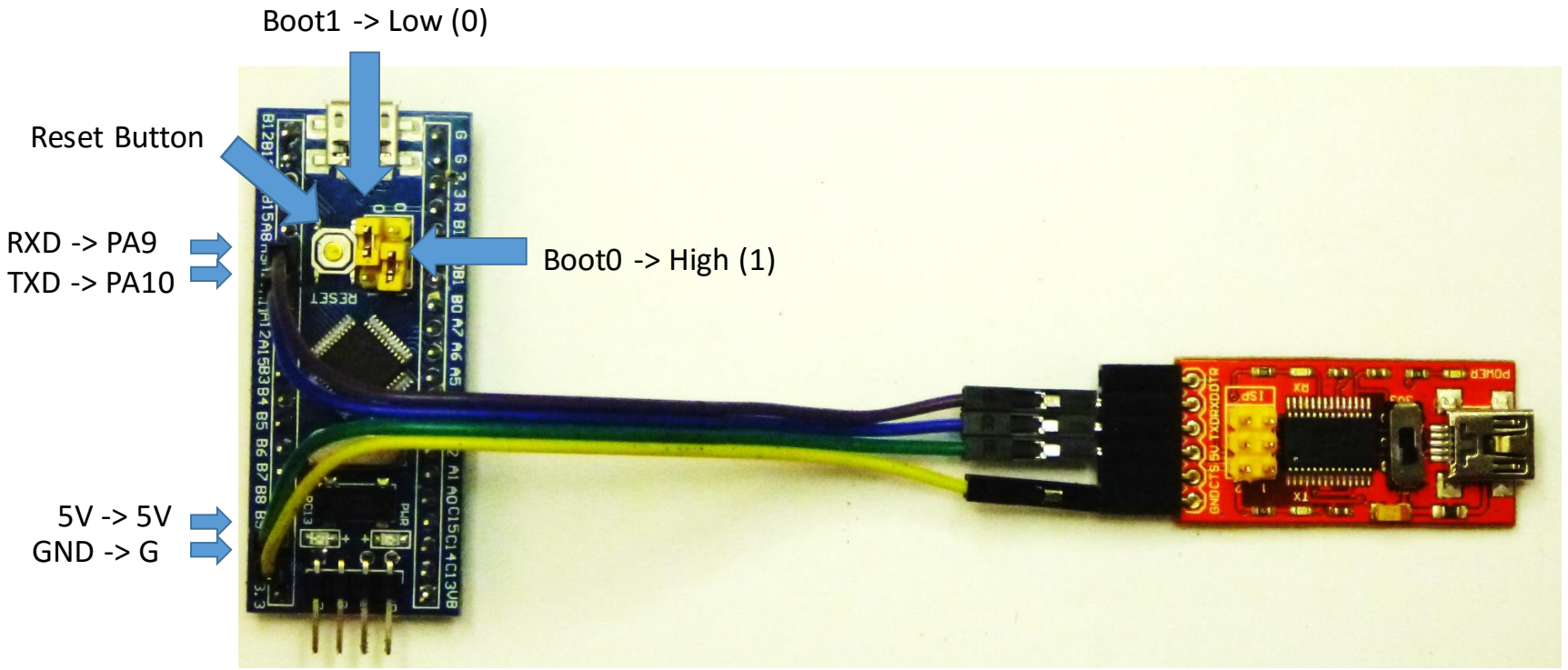
STM32/ARM Programming Blink using STM32F10xxx Reference and HAL Manuals

LED PC13 pin/port Loop Code Hand Generated

```
// the loop function runs over and over again forever
void loop() {
  // turn the LED on (HIGH is the voltage level)
  // digitalWrite(LED_BUILTIN, HIGH);          // Arduino
  GPIOC->ODR |= 0X00020000;                    // Hand Generated
  // delay(1000);                              // wait for a second (Arduino)
  HAL_Delay(1000);                             // wait for a second (HAL)
  // turn the LED off by making the voltage LOW
  //digitalWrite(LED_BUILTIN, LOW);           // Arduino
  GPIOC->ODR &= ~ 0X00020000;                  // Hand Generated
  // delay(1000);                              // wait for a second (Arduino)
  HAL_Delay(1000);                             // wait for a second (HAL)
}
```

STM32/ARM Programming Blink using STM32F10xxx Reference Manual

Downloading Hand Generated Sketch to Blue-Pill F103C8 using USB-to-Serial



Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual

- **Setup**
  - Enable IO Port Clock before configuring IO pins (RCC->APB2ENR |= 32 bit pattern)
  - Write IO pin Configuration and Mode registers (GPIOx->CRL, GPIO->CRH)
- **Set/Clear IO Bits**
  - 32 bit GPIOx->ODR read-modify-write sequence
    - Set: GPIOx->ODR |= 32 bit pattern to set (only lower 16 bits used);
    - Clear: GPIOx->ODR &= ~ 32 bit pattern to clear (only lower 16 bits used);
  - 32 bit GPIOx->XXX Atomic operation
    - Set/Reset register GPIOx->BSRR |= 32 bit set/reset bit pair pattern;
    - Reset register GPIOx->BRR |= 32 bit pattern to clear (only lower 16 bits used);
- **Writing Multiple Bits** (e.g. Nibble, Byte) with read-modify-write sequence
  - Clear bits to write using GPIOx &= ~ (32 bit data mask);
  - Set bits to write using GPIOx |= (32 bit data) & (32 bit data mask);
  - Alternate: Single line version using
    GPIOx = (GPIOx & ~ (32 bit data mask)) | ((32 bit data) & (32 bit data mask));

Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

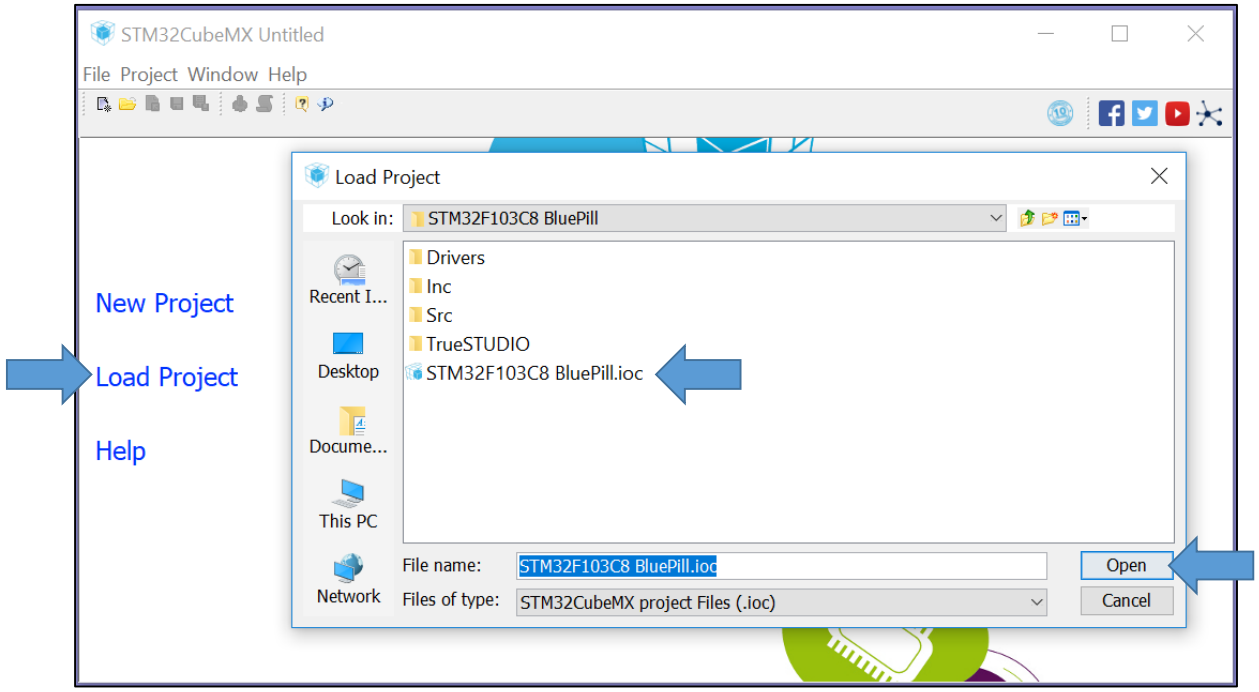
- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32/ARM Programming Quadrature Decoder using STM32CubeMX to Generate Code

Opening STM32F103 C8 BluePill in STM32CubeMX

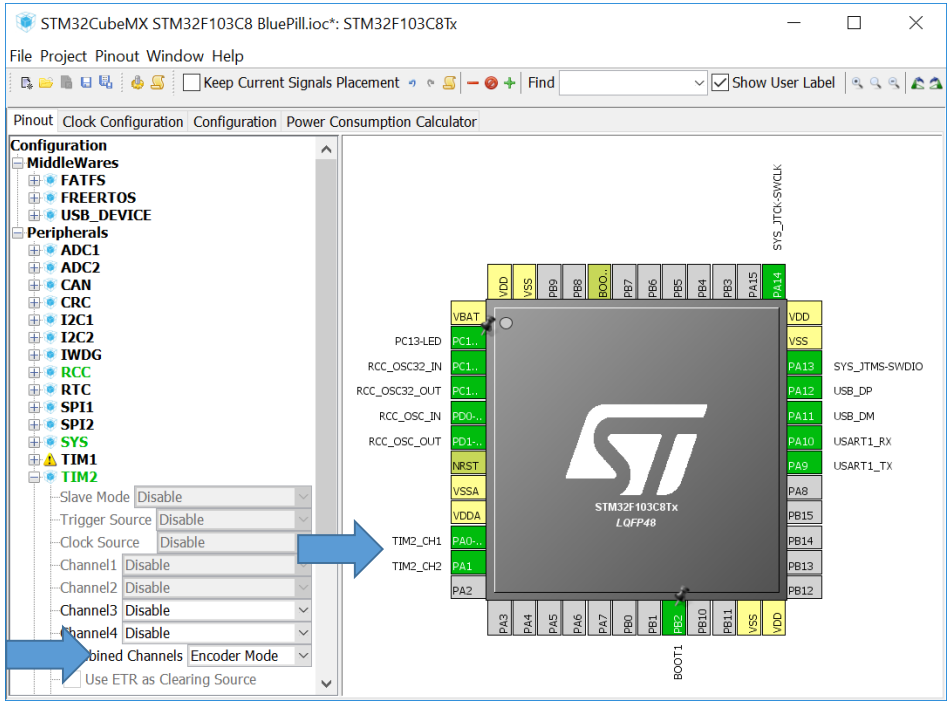
- Open STM32CubeMX
- Select “Load Project”
- Select STM32F103C8_BluePill.ioc and Open



STM32/ARM Programming Quadrature Decoder using STM32CubeMX to Generate Code

Configuring TIM2 in Encoder Mode

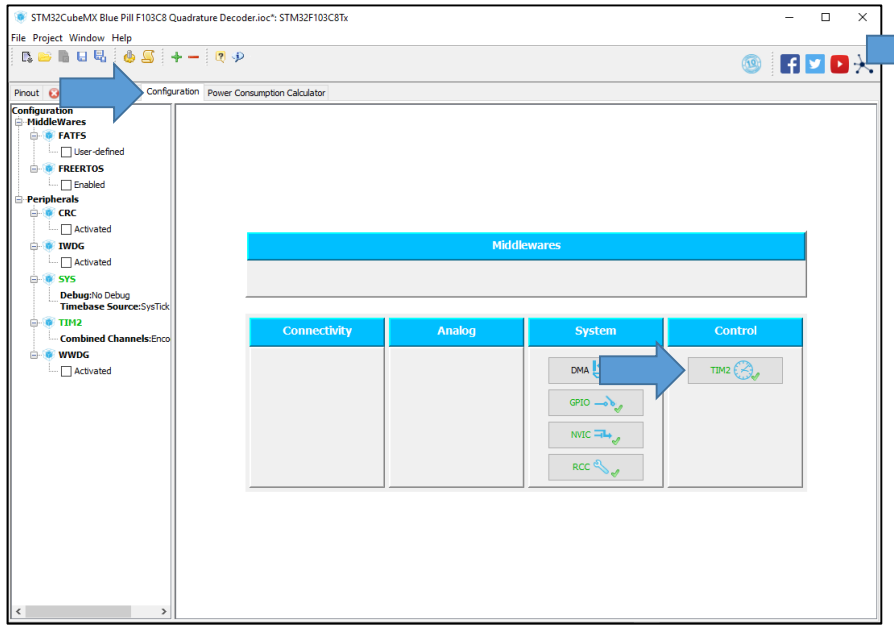
- Click + next to TIM2, select “Combined Channels”, select “Encoder Mode”
- Observe pins PA0 and PA1 get labeled “TIM1_CH1” and “TIM1_CH2” which are encoder channels A & B

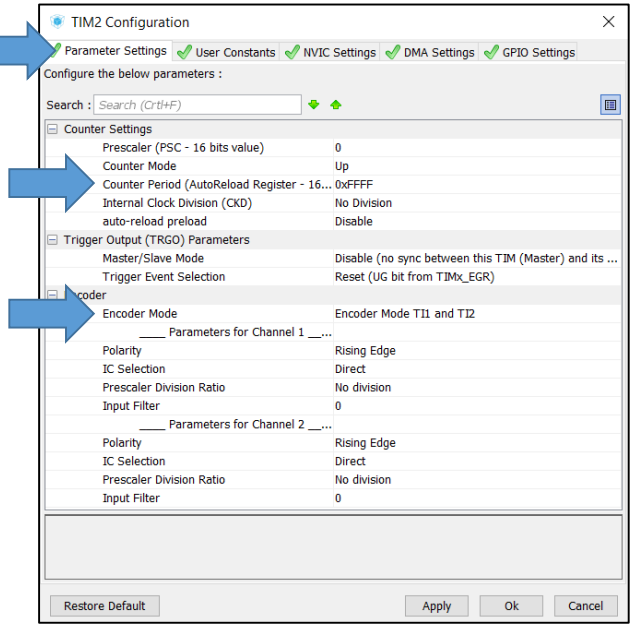


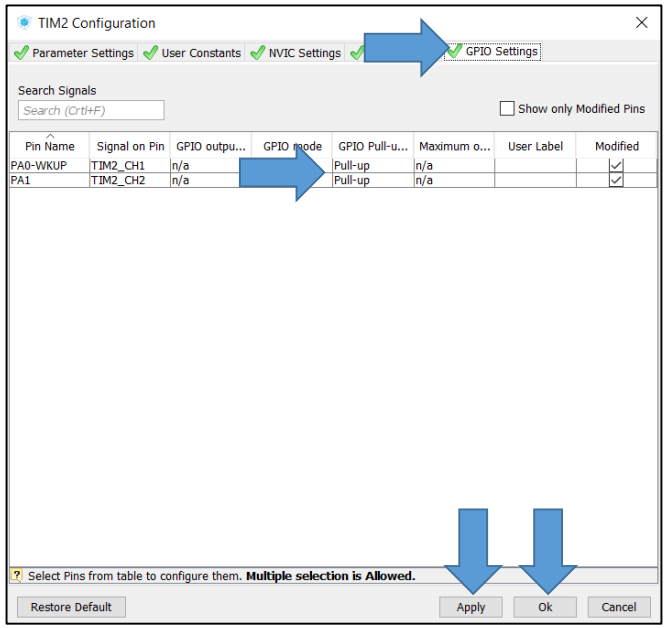
STM32/ARM Programming Quadrature Decoder using STM32CubeMX

Configuring TIM2 and GPIO Settings in STM32CubeMX

- In STM32CubeMX select “Configuration” tab
- Select/Click TIM2 under Control and Parameter Settings Tab, enter 0xFFFF for Counter Period, select Encoder Mode TI1 and TI2, click Apply and OK
- Click GPIO Setting Tab and change GPIO Pull-up/Pull-down values for PA0 and PA1 to Pull-Up (for testing purposes only)
- Click Apply and OK







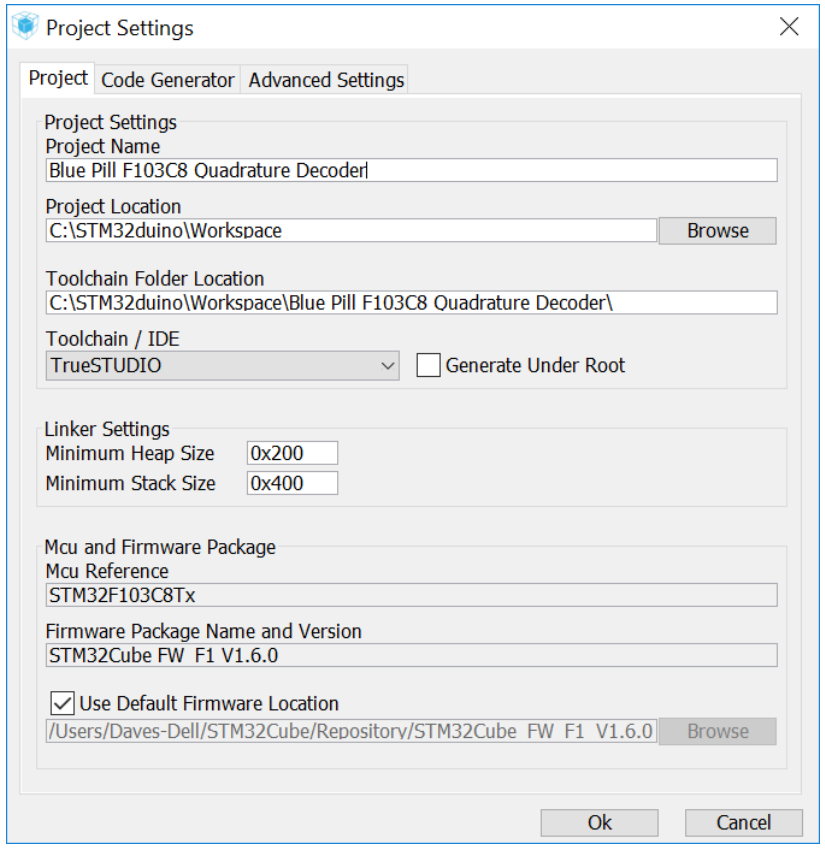
STM32/ARM Programming Quadrature Decoder using STM32CubeMX

Naming, Configuring, and Saving STM32CubeMX Project

- Select File > Save Project As..

  This opens the Project Settings panel
- Project Settings > Project > Project Name

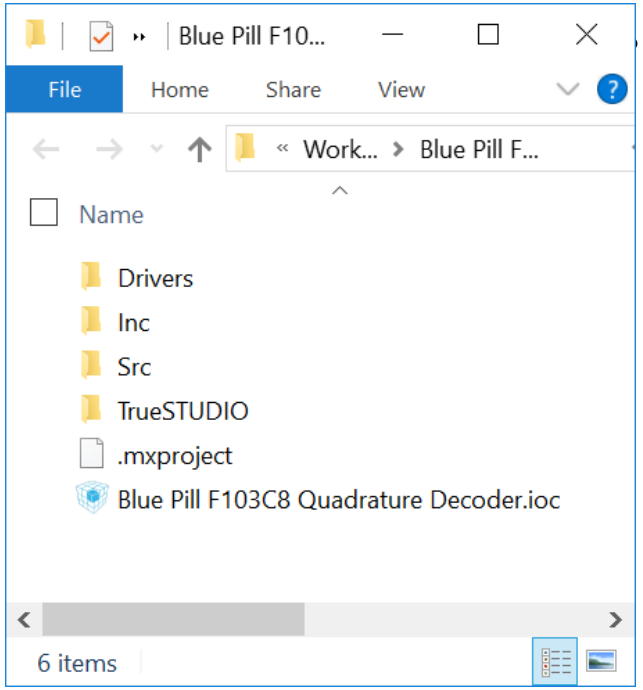
  Enter Blue Pill F103C8 Quadrature Decoder
- Leave Project Location and Toolchain/IDE alone
- Click OK



STM32/ARM Programming Quadrature Decoder using STM32CubeMX

Generating and saving initialization code using STM32CubeMX

- In STM32CubeMX Select Project > Generate Code



- In Code Generation "The Code is successfully …" message: select "Open Folder"
  - Drivers
    - CMSIS and STM32F1xx_HAL_Driver standard definitions
  - Inc
    - main.h – error handler prototype
    - stm32f1xx_hal_conf.h – HAL configuration defines
    - stm32f1xx_it.h – interrupt handler prototypes
  - Src
    - main.c
      - calls to and high level code for HAL and SystemClock initialization routines
      - **MX_TIM2_Init() and MX_GPIO_Init() initialization/configuration routines**
      - While(1) loop
    - stm32f1xx_hal_msp.c – HAL TIM Encoder initialization routine
      - **HAL_TIM_Encoder_MspInit(TIM_HandleTypeDef* htim_encoder)**
    - stm32f1xx_it.c – NVIC interrupt handler routines
    - system_stm32f1xx.c – System clock initialization routine

Suggest installing "Notepad++" (https://notepad-plus-plus.org/) for viewing code files.

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

main.c GPIO pin/port and TIM2 initialization code Generated by STM32CubeMX

Initialization Call

```
TIM_HandleTypeDef htim2;
int main(void) {
        HAL_Init();
        SystemClock_Config();
        MX_GPIO_Init();
        MX_TIM2_Init();
 while (1)  {
   /* USER CODE */
 }
}
```

GPIO Initialization

```
static void MX_GPIO_Init(void)
{
  /* GPIO Ports Clock Enable */
__HAL_RCC_GPIOA_CLK_ENABLE();
}
```

Code unique to this application

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

TIM2 initialization routine Generated by STM32CubeMX

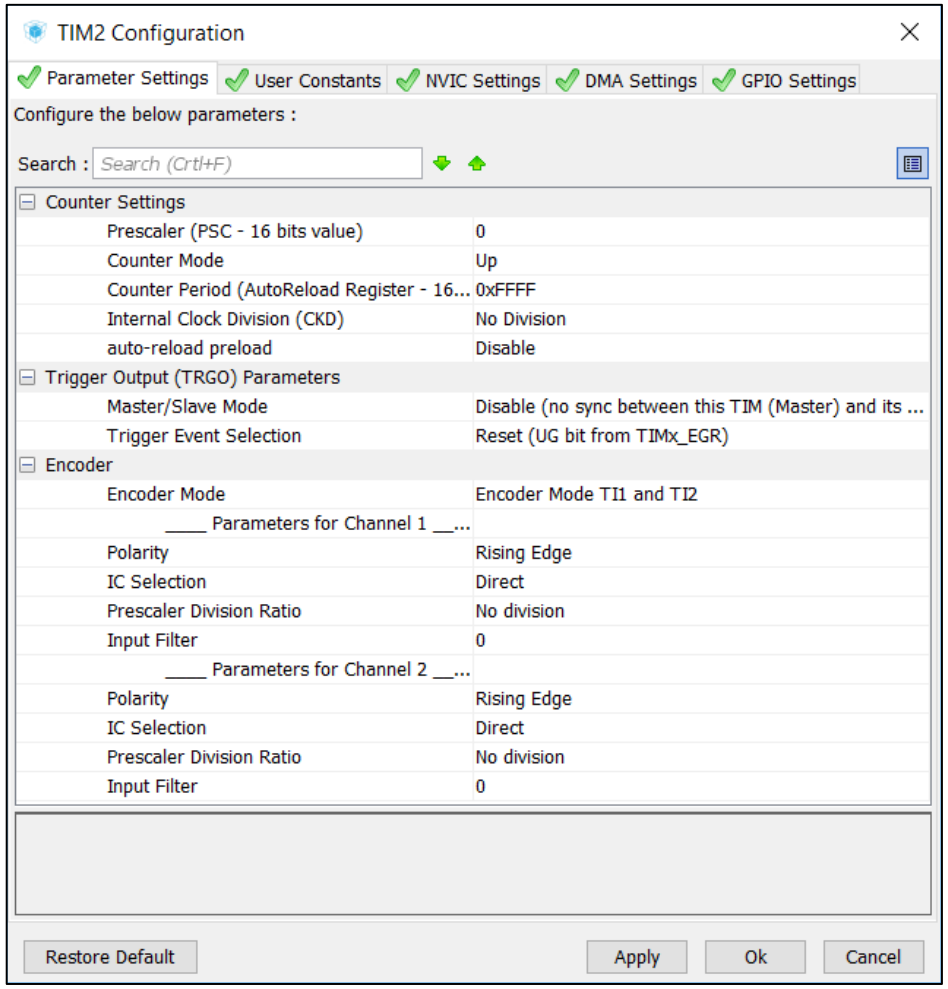
```
/* TIM2 init function */
static void MX_TIM2_Init(void)
{
  TIM_Encoder_InitTypeDef sConfig;
  TIM_MasterConfigTypeDef sMasterConfig;
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 0xFFFF;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 0;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC2Filter = 0;
```

```
  if (HAL_TIM_Encoder_Init(&htim2, &sConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}
```

Code unique to this application

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

TIM2 STM32CubeMX Settings vs Generated Code




```
TIM_HandleTypeDef  htim2;
htim2.Instance = TIM2;
htim2.Init.Prescaler  = 0;
htim2.Init.CounterMode  = TIM_COUNTERMODE_UP;
htim2.Init.Period  = 0xFFFF;
htim2.Init.ClockDivision  = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload  = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

```
TIM_MasterConfigTypeDef  sMasterConfig;
sMasterConfig.MasterOutputTrigger  = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode  = TIM_MASTERSLAVEMODE_DISABLE;
```

```
TIM_Encoder_InitTypeDef  sConfig;
sConfig.EncoderMode  = TIM_ENCODERMODE_TI12;
sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC1Selection  = TIM_ICSELECTION_DIRECTTI;
sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
sConfig.IC1Filter = 0;
sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC2Selection  = TIM_ICSELECTION_DIRECTTI;
sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
sConfig.IC2Filter = 0;
```

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

HAL_TIM_Encoder_Init (TIM2 GPIO Configuration) routine Generated by STM32CubeMX

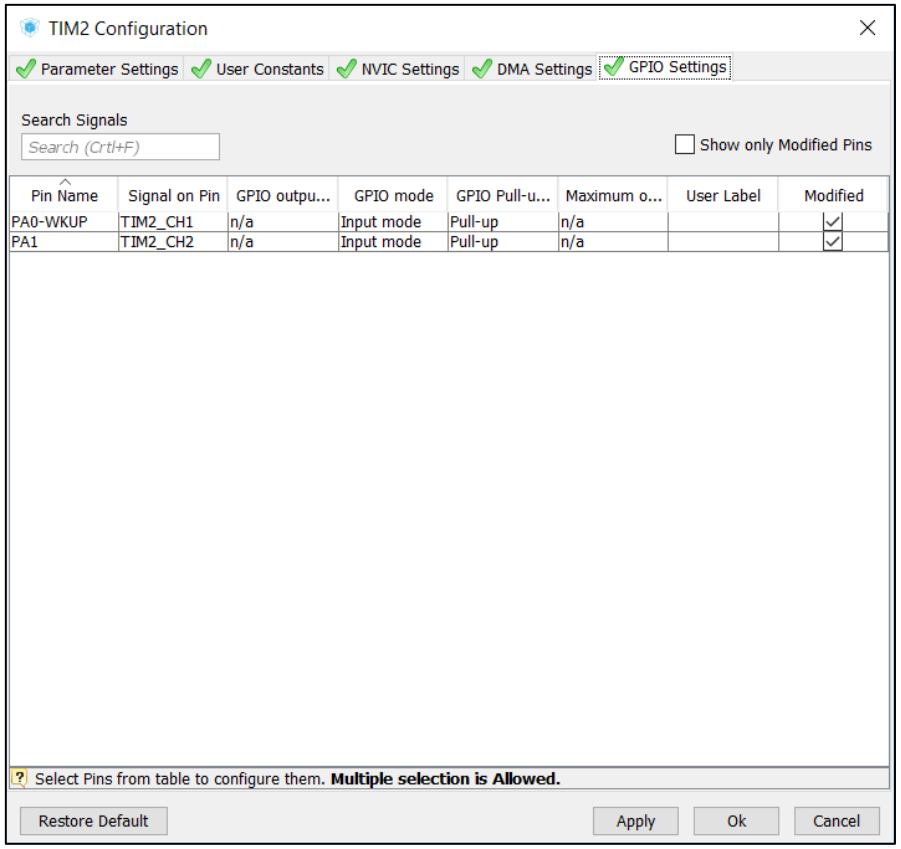
```
void HAL_TIM_Encoder_MspInit(TIM_HandleTypeDef*
htim_encoder)
{

 GPIO_InitTypeDef GPIO_InitStruct;
 if(htim_encoder->Instance==TIM2)
 {
 /* USER CODE BEGIN TIM2_MspInit 0 */

 /* USER CODE END TIM2_MspInit 0 */
  /* Peripheral clock enable */
  __HAL_RCC_TIM2_CLK_ENABLE();
```

```
/**TIM2 GPIO Configuration
  PA0-WKUP     ------> TIM2_CH1
  PA1     ------> TIM2_CH2
  */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

 /* USER CODE BEGIN TIM2_MspInit 1 */

 /* USER CODE END TIM2_MspInit 1 */
 }

}
```

Code unique to this application

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

TIM2 STM32CubeMX Settings vs Generated Code




```
GPIO_InitTypeDef GPIO_InitStruct;
```

```
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

LED pin/port HAL Initialization Code Generated by STM32CubeMX as Arduino Sketch

```
/*
   STM32CubeMX and HAL Quadrature Encoder for Blue Pill F103C8
   LED pin/port initialization code Generated by STM32CubeMX as
   Arduino routines
*/

// MX_TIM_2 private variable
TIM_HandleTypeDef htim2;

// MX_GPIO_Init and MX_TIM2_Init prototypes generated by STM32CubeMX
static void MX_GPIO_Init(void);                 // STM32CubeMX
static void MX_TIM2_Init(void);                 // STM32CubeMX

// the setup function runs once when you press reset or power the board
void setup() {
   // initialize digital pins and TIM2
   MX_GPIO_Init();                              // STM32CubeMX
   MX_TIM2_Init();                              // STM32CubeMX
}
```

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

GPIO and TIM2 HAL Initialization Code Generated by STM32CubeMX as Arduino Sketch

```
/* GPIO init function */
static void MX_GPIO_Init(void)
{
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
 /*** copied from HAL_TIM_Encoder_Init ***/
  GPIO_InitTypeDef GPIO_InitStruct;
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /*** end copy from HAL_TIM_Encoder_Init ***/
}
```

```
/* TIM2 init function */
static void MX_TIM2_Init(void)
{
  TIM_Encoder_InitTypeDef sConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 0xFFFF;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 0;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig);
}
```

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

GPIO and TIM2 HAL Initialization Code Generated by STM32CubeMX Arduino Sketch: Error Messages

Arduino: 1.8.3 (Windows 10), Board: "BluePill F103C8, Serial"

C:\Users\<name>\Documents\Arduino\Quadrature_Decoder\Quadrature_Decoder.ino: In function 'void MX_TIM2_Init()':

Quadrature_Decoder:51: error: 'struct TIM_Base_InitTypeDef' has no member named 'AutoReloadPreload'

htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

^

Quadrature_Decoder:51: error: 'TIM_AUTORELOAD_PRELOAD_DISABLE' was not declared in this scope

htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

^

exit status 1

'struct TIM_Base_InitTypeDef' has no member named 'AutoReloadPreload'

****

STM32/ARM Programming Quadrature Decoder using STM32CubeMX

GPIO and TIM2 HAL Initialization Code Generated by STM32CubeMX Arduino Sketch: Conclusions

- STM32CubeMX Provides a great graphical representation of a STM32Fxxx part's capabilities
  - Lists part's Peripherals with ability to select/modify capabilities
  - Shows pin Functions and names based on selected Peripheral capabilities
  - Provides ability to refine Peripheral pin Functional capabilities
- STM32CubeMX Provides ability to generate STM32Fxxx part's initialization code
  - Generated code provides great guidance for hand coding initialization code
  - Hand coded version ***may or may not*** compile error free as an Arduino/ARM sketch
    - When generated code doesn't compile error free:
      - First see if there's an easy fix, e.g. missing *.c/*.h code
      - If not, suggest using STM32CubeMX graphical representation and generated code plus STM32Fxxx Reference Manual as basis for hand generating initialization code.

Intermediate Level STM32/ARM Programming

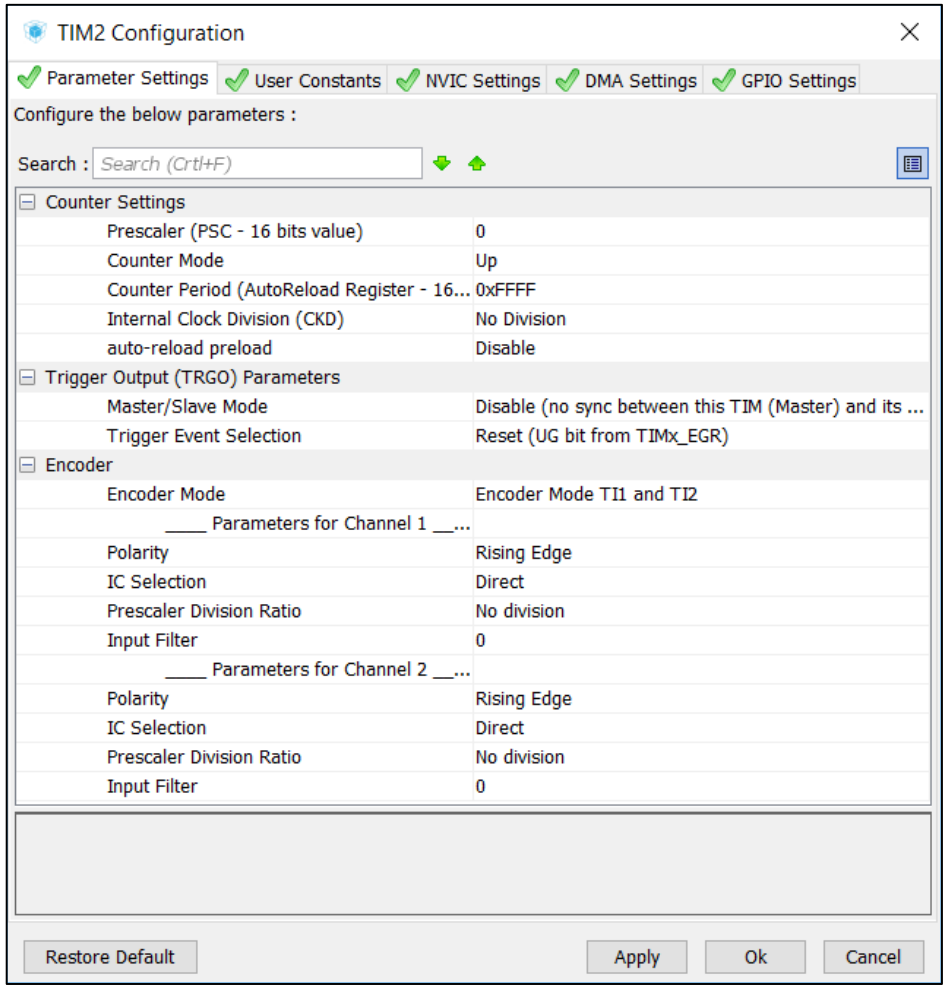
Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

TIM2 STM32CubeMX Settings vs Generated Code




```
TIM_HandleTypeDef  htim2;
htim2.Instance = TIM2;
htim2.Init.Prescaler  = 0;
htim2.Init.CounterMode  = TIM_COUNTERMODE_UP;
htim2.Init.Period  = 0xFFFF;
htim2.Init.ClockDivision  = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload  = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

```
TIM_MasterConfigTypeDef  sMasterConfig;
sMasterConfig.MasterOutputTrigger  = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode  = TIM_MASTERSLAVEMODE_DISABLE;
```

```
TIM_Encoder_InitTypeDef  sConfig;
sConfig.EncoderMode  = TIM_ENCODERMODE_TI12;
sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC1Selection  = TIM_ICSELECTION_DIRECTTI;
sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
sConfig.IC1Filter = 0;
sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC2Selection  = TIM_ICSELECTION_DIRECTTI;
sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
sConfig.IC2Filter = 0;
```

STM32F10xxx Ref Manual

15 General-purpose timers (TIMx)

15.3 TIMx functional description

15.3.12 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and ***SMS=011 if it is counting on both TI1 and TI2 edges.***

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

STM32F10xxx Ref Manual

15.3.12 Encoder interface mode – continued

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter jus counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the user must configure TIMx_ARR register (e.g. 0xFFFF) before starting. In the s way, the capture, compare, prescaler, trigger output features continue to work as normal.

Figure 134 gives an example of counter operation, showing count signal generation and direction control. ***It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positior near to one of the switching points.***

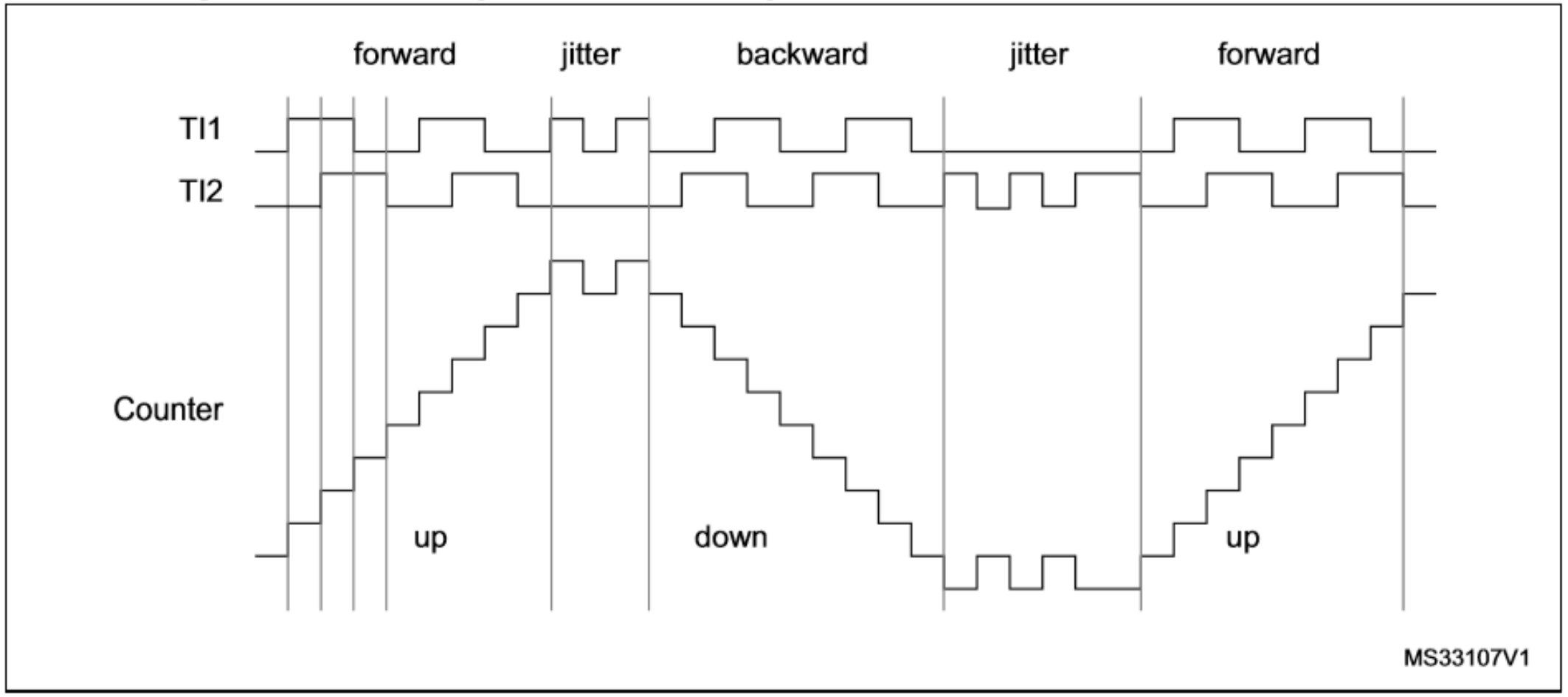
For this example we assume that the configuration is the following:

- CEN = 1 (TIMx_CR1 reg, Counter is enabled)
- SMS= '011' (TIMx_SMCR reg, both inputs are active on both rising and falling edges)
- CC2S= '01', CC1S= '01' (TIMx_CCMR1 reg, TI1FP1 mapped on TI1, and TI2FP2 mapped on TI2)
- CC1P= '0', CC1E = '1', IC1F = '0000' (TIMx_CCER reg, TI1FP1 noninvert, TI1FP1=TI1, enabled, no filter)
- CC2P= '0', CC2E = '1', IC2F = '0000' (TIMx_CCER reg, TI2FP2 noninvert, TI1FP2=TI2, enabled, no filter)
- ARR = 0xFFFF (TIMx_ARR auto reload register value)

STM32F10xxx Ref Manual

15.3.12 Encoder interface mode – continued

Figure 134. Example of counter operation in encoder interface mode



STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

STM32CubeMX – TIM2 Configuration

- TIM2 Counter Settings
  - Prescaler: 0
  - Counter Mode: UP
  - Period: 0xFFFF
  - Clock Division: No Division
  - Auto Reload Preload: Disable
- TIM2 Trigger Output
  - Master/Slave Mode: Disable
  - Trigger Selection: Reset
- TIM2 Encoder
  - Encoder Mode: TI1 and TI2
  - Channel 1
    - Polarity: Rising Edge
    - IC Selection: Direct
    - Prescale Division: No Division
    - Input Filter: 0
  - Channel 2
    - Polarity: Rising Edge
    - IC Selection: Direct
    - Prescale Division: No Division
    - Input Filter: 0

STM32F10xxx Reference Manual – Register Settings

TIM2_CR1: CEN = '1' [Counter Enabled]
DIR = '0' [Counter Mode: UP] – reset/default
TIN2_ARR ARR = 0xFFFF [Auto Reload value]
CKD = '00' [Clock Division: No Division] – reset/default
ARPE = '0' [Auto Reload Preload: Disable] – reset/default
TIM2_CR2: TI1S = '0' [TIM2_CH1 pin connected to TI1 input] – reset/default

TIM2_SMCR: SMS = '011' [Encoder mode 3 - Both Inputs Active]

TIM2_CCMR1: CC1S = '01' , IC1F ='0000' [TI1FP1 mapped to TI1, and no filter]
TIM2_CCER: CC1P = '0', CC1E = '1', [TI1FP1 noninverting, TI1FP1 = TI1, enabled]

TIM2_CCMR1: CC2S = '01', IC2F ='0000' [TI2FP2 mapped to TI2, and no filter]
TIM2_CCER: CC2P = '0', CC2E = '0', [TI2FP2 noninverting, TI2FP2 = TI2, enabled]

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

15.4 TIMx Registers

15.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

15.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.

011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

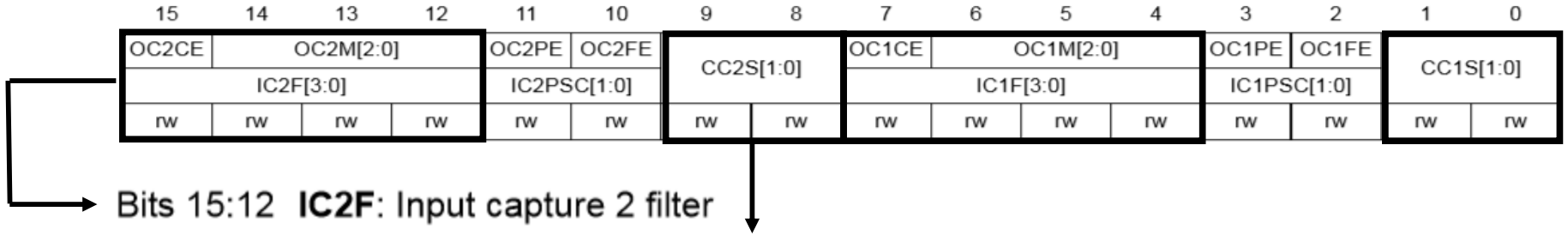
STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

15.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.



Bits 15:12 **IC2F**: Input capture 2 filter

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

15.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Bit 5 **CC2P**: Capture/Compare 2 output polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as input:
This bit selects whether IC1 or IC1 is used for trigger or capture operations.
0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted.
1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

15.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

15.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

Quadrature Decoder Initialization Code Generated by Hand

```
/*
   CMSIS Hand Coded Blink Blue Pill F103C8
   Turns on an LED on for one second, then off for one second, repeatedly.
   LED pin/port CMSIS Initialization and Loop Code Generated by Hand based
   on ST-Microelectronics' STM32CubeMX and STM32F10xxx Reference Manual
   substituted for Arduino routines
*/

// the setup function runs once when you press reset or power the board
void setup() {
   // initialize digital pin LED_BUILTIN as an output.
   // pinMode(LED_BUILTIN, OUTPUT);                    // Arduino
   // enable Port C clock                              // CMSIS Hand Generated
   RCC->APB2ENR |= 0x0010;
   // clear Port C Bit 13's output configuration/mode
   GPIOC->CRH &= ~ 0X00F00000;
   // set Port C Bit 13's output configuration/mode
   GPIOC->CRH |=  0X00200000;  // output push-pull slow
}
```

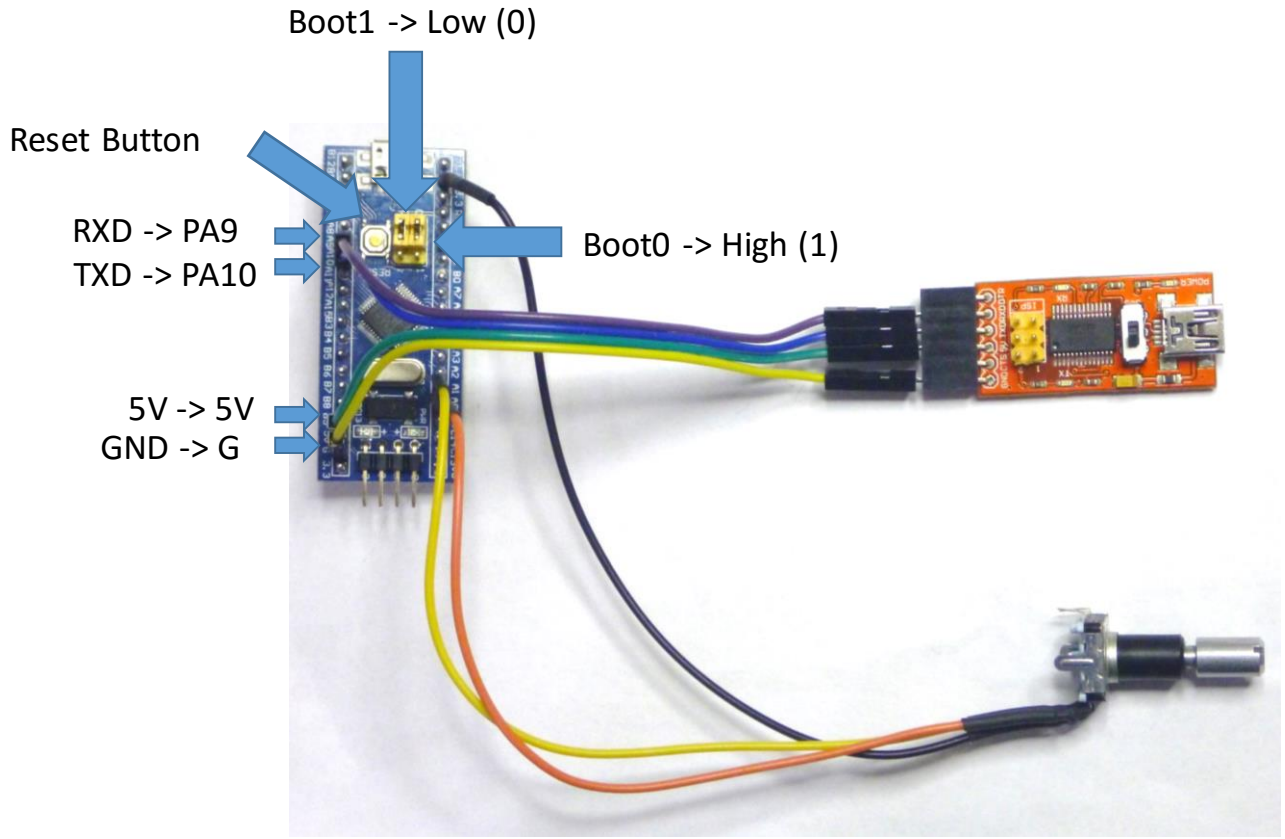
STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

Quadrature Decoder Loop Code Generated by Hand

```
// the loop function runs over and over again forever
void loop() {
   // turn the LED on (HIGH is the voltage level)
   // digitalWrite(LED_BUILTIN, HIGH);                 // Arduino
   GPIOC->ODR |= 0X00020000;                           // CMSIS Hand Generated
   delay(1000);                                        // wait for a second
   // turn the LED off by making the voltage LOW
   //digitalWrite(LED_BUILTIN, LOW);                   // Arduino
   GPIOC->ODR &= ~ 0X00020000;                         // CMSIS Hand Generated
   delay(1000);                                        // wait for a second
}
```

STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual

Downloading Hand Generated Sketch to Blue-Pill F103C8 using USB-to-Serial



Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

STM32/ARM Programming
Periodic Task Scheduling using HAL_SYSTICK_Callback()

Void “HAL_SYSTICK_Callback(void){}” is an ARM Cortex processor predefined “user routine” that is called by the system once every millisecond by the SysTick interrupt.

Unfortunately, it’s broken in the current release of the STM32F1xx/STM32 Cores by ST-Microelectronics, but will be fixed in the next release. It can be fix in the current release by adding the missing “HAL_SYSTICK_IRQHandler()” call in the “SysTick_Handler” in:

C:\Users\<name>\AppData\Local\Arduino15\packages\STM32\hardware\stm32\2017.8.31\cores\arduino\stm32\clock.c ***and***

C:\Users\<name>\AppData\Local\Arduino15\packages\STM32\hardware\stm32f1\2017.1.20\cores\arduino\libstm32f1\clock.c

Currently defined as:

```
void SysTick_Handler(void)
{
   HAL_IncTick();
}
```

Should be defined as:

```
void SysTick_Handler(void)
{
   HAL_IncTick();
   HAL_SYSTICK_IRQHandler();
}
```

STM32/ARM Programming
Periodic Task Scheduling using HAL_SYSTICK_Callback()

```
/* Sample one second sketch using HAL_SYSTICK_Callback() */
bool SysTickFlag = false;
int16_t SysTickCounter = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  HAL_Delay(1000);
  Serial.println("Hello World"); // print something to start
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  if (SysTickFlag) {
    Serial.println(" One second SysTick");
    SysTickFlag = false;
  }
}

// user defined HAL_SYSTICK_Callback
void HAL_SYSTICK_Callback(void) {
  if (--SysTickCounter < 0) {
    SysTickCounter = 1000;
    SysTickFlag = true;
  }
}
```

Intermediate Level STM32/ARM Programming

Based on ST-Microelectronics' integration of
CMSIS* and STM32CubeMX HAL definitions into Arduino IDE

- Arduino IDE STM32/ARM Entry to Advanced Level Devices
- Download STM32F10xxx Datasheet plus Reference and HAL/Low-Layer Driver Manuals
- Download/Install STM32CubeMX and Open STM32F103C8/Blue Pill
- STM32F103C8 Blue Pill Pinout vs STM32F103C8/Blue Pill in STM32CubeMX
- STM32/ARM Programming Blink using STM32CubeMX
- STM32/ARM Programming Blink using STM32F10xxx Reference Manual
- STM32/ARM Programming Bit, Nibble, Byte IO using STM32F10xxx Reference Manual
- STM32/ARM Programming Quadrature Decoder using STM32CubeMX
- STM32/ARM Programming Quadrature Decoder using STM32F10xxx Ref Manual
- STM32/ARM Programming Periodic Task Scheduling using HAL_SYSTICK_Callback()
- My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

* **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard

My Thoughts About STM32F1xx/STM32 Cores by ST-Microelectronics

Warning: This is a leading/bleeding edge technology – expect bugs

Assuming you are currently using Arduino IDE

- Great way to get started programming STM32/ARMs boards
- Easiest to use with existing/supported Arduino sketches
- Serial and ST-Link downloading worked great, but had problems with bootloader*
- STM32CubeMX is a great tool for design exploration
- STM32CubeMX is not as great for generating Arduino IDE code
- This technology is only useful for listed STM32Fxxx boards
- The STM32Fxxx Register definitions and their usage appears to be solid
- The STM32Fxxx CMSIS/HAL definitions and their usage appears to be having growing pains

My personal preference is an industrial strength ARM IDE like Atollic's TrueSTUDIO with STM32CubeMX

* My bootloader problems are likely just my lack of experience.