STM32 Overview

Ing. Davide Giacalone
Advanced Research Engineer, SRA
STM32 is more than a CPU

- Cortex®-M7
- Quad-SPI Flash
- SPI / TDM / DFSDM
- Camera IF
- SPDIF
- Ethernet
- TFT-LCD controller and MIPI®-DSI
- SPI
- SPI, UART, CAN, I²C
- SD Card
- USB
- Camera IF
STM32 Tools

- **Hardware Development Tools**
  - STM32 Nucleo boards
  - STM32 Discovery kits
  - STM32 Eval boards

- **STM32 Docs**
  - Datasheet
  - Reference Manual

- **STM32CubeMX**
  - Graphical tool
  - Easy configuration of STM32 microcontrollers peripherals
  - C code generation, compliant with STM32 MCU

- **STM32CubeIDE**
  - C/C++ development platform
  - Code generation, compilation, and debug for STM32 microcontrollers
STM32 boards

- Developing your own board
- Using existing STMicroelectronics boards
  - STM32 Nucleo boards
  - STM32 Discovery kits
  - STM32 Eval boards

<table>
<thead>
<tr>
<th>Board</th>
<th>ST-Link</th>
<th>ARDUINO connector</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32 Nucleo</td>
<td>Yes (V2.1/V3)</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>STM32 Discovery</td>
<td>Yes (V2.1/V3)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>STM32 Eval</td>
<td>Yes (V2.1/V3)</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
• Each Nucleo board is based on one of the STM32 MCU product lines

• Two types of extension resources:
  • Arduino Uno v3 connectivity.
  • Morpho headers for easy access to all MCU peripherals.

• Integrated ST-LINK/V2-1 debugger and programmer:
  • supports drag-and-drop flash programming.
  • can target on-board STM32 or external STM32-based application.
STM32 Nucleo key assets

Flexible board power supply
Through USB or external source

Integrated ST-Link/V2-1
Mass storage device flash programming

2 push buttons, 2 color LEDs

Direct access to all STM32 I/Os:
through Morpho extension headers

Flexible prototyping

Unlimited expansion capabilities

Simply expose the whole STM32 portfolio to the communities
Enlarging the STM32Nucleo family to cover whole STM32 portfolio

- STM32 with 32 pins
- STM32 with 64 pins
- STM32 with 144 pins

One STM32 Microcontroller
- Arduino Uno connectors
- Arduino Nano connectors
- ST Zio connectors: easy add-on
- ST morpho extension: direct access to all mcu I/Os

Flexible board power supply: through USB or external source

Integrated ST-Link/V2-1: drag & drop device flash programming

Ethernet & USB OTG (optional)

STM32 complete product range from ultra-low power to high performance
Examples of Nucleo Expansion boards

<table>
<thead>
<tr>
<th>Sense</th>
<th>Connect</th>
<th>Power Drive</th>
<th>Move Actuate</th>
<th>Translate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion &amp; environmental sensors</td>
<td>BLE</td>
<td>Power management</td>
<td>Motor drive</td>
<td>Audio amplifier</td>
</tr>
<tr>
<td></td>
<td>Wi-Fi</td>
<td>LED boost</td>
<td>Actuator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sub-GHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NFC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Updater settings and manage embedded software packages
• Create new project (standalone MCU or board)
• Pinout
• Peripheral configuration
• Clock tree
• NVIC configuration
• Additional Software
• Project generation
Create new project

MCUs selector

Board selector
Create new project – MCUs selector

You can filter MCUs with:

- Part Number
- Core
- Series
- Lines
- Peripherals
- ……..
Create new project – Board selector

You can filter Boards with:
• Part Number
• Vendor
• Type
• MCU series

Link to Documentation and web page
• Pinout & Configuration
• Clock Configuration
• Project Manager
• Tools
Pinout & Configuration view

Peripherals Configuration section

MCU pinout section
From the **Peripherals Configuration section**, the user can select the peripherals from a list and configure the peripheral modes required for the application. STM32CubeMX assigns and configures the pins accordingly.
For more advanced users, it is also possible to directly map a peripheral function to a physical pin using the Chip view. The signals can be locked on pins to prevent STM32CubeMX conflict solver from moving the signal to another pin.
Peripherals Configuration section (set interrupts)
On CubeMx, check Parameter Settings values for Usart2

Hello World on uart (1/2)
On STM32CubeIDE:
- Define the _write function
- Write the printf function

```c
/* USER CODE BEGIN PFP */

int _write(int fd, char* ptr, int len) {
  HAL_UART_Transmit(huart2, (uint8_t *) ptr, len, HAL_MAX_DELAY);
  return len;
}

/* USER CODE `"`*/

printf("hello\r\n");
```
Led blinking, solution 1 (not precise)

- On CubeMx, verify PA5 is set as GPIO_output
- Toggle the pin status
- Add a delay

```c
/* USER CODE BEGIN 2 */
90 /* USER CODE END 2 */
91 /* Infinite loop */
92 /* USER CODE BEGIN WHILE */
93 while (1)
94 { /* USER CODE BEGIN WHILE */
95 HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
96 HAL_Delay(500);
97 /* USER CODE END WHILE */
98 /* USER CODE BEGIN 3 */
99 }
100 /* USER CODE END 3 */
```
Led blinking, with timer and interrupt (1/2)

- Enable timer 10 and set its params
- Enable the related interrupt

1. Click on TIM10
2. Enable the flag
3. Set params
4. Open the NVIC Settings and enable the tim10 global interrupt
• Declare a global flag
• Make the timer 10 start in interrupt mode
• Redefine a callback function related to the elapsed time and set the flag
• Manage the flag in the main loop
1. Set PA5 as TIM2_CH1
2. Enable Clock Source
3. Set channel 1 as PWM Generation Ch1
4. Adjust Counter period and Pulse values
5. In the c code, let the timer 2 pwm start
   Remember to remove all the references to LD2
1. Check the PC13 pin is set as GPIO_EXTI13

2. Enable the EXTI line[15:10] interrupts
• Declare a global flag
• Redefine a callback function related to the pressed button and set the flag
• Manage the flag in the main loop

```c
46 /* USER CODE BEGIN PV */
47 volatile uint8_t button_pressed = 0;

298 /* USER CODE BEGIN 4 */
299
300 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
301 {
302  /* Prevent unused argument(s) compilation warning */
303  if(GPIO_Pin == GPIO_PIN_13)
304  {
305      button_pressed - 1;
306  }
307 }

109 /* Infinite loop */
110 /* USER CODE BEGIN WHILE */
111 while (1)
112 {
113     if(button_pressed)
114     {
115         button_pressed = 0;
116         printf("yes, you pressed a button!\r\n");
117     }
118 ```