GPIO Operations on STM32 Microcontrollers using HAL

You can use the STM32CubeMX tool to create the necessary config. files to enable the GPIO Pins.
In this tutorial I'm going to explain how you can modify the generated GPIO configs and add additional GPIOs.

This tutorial uses the following equipment:

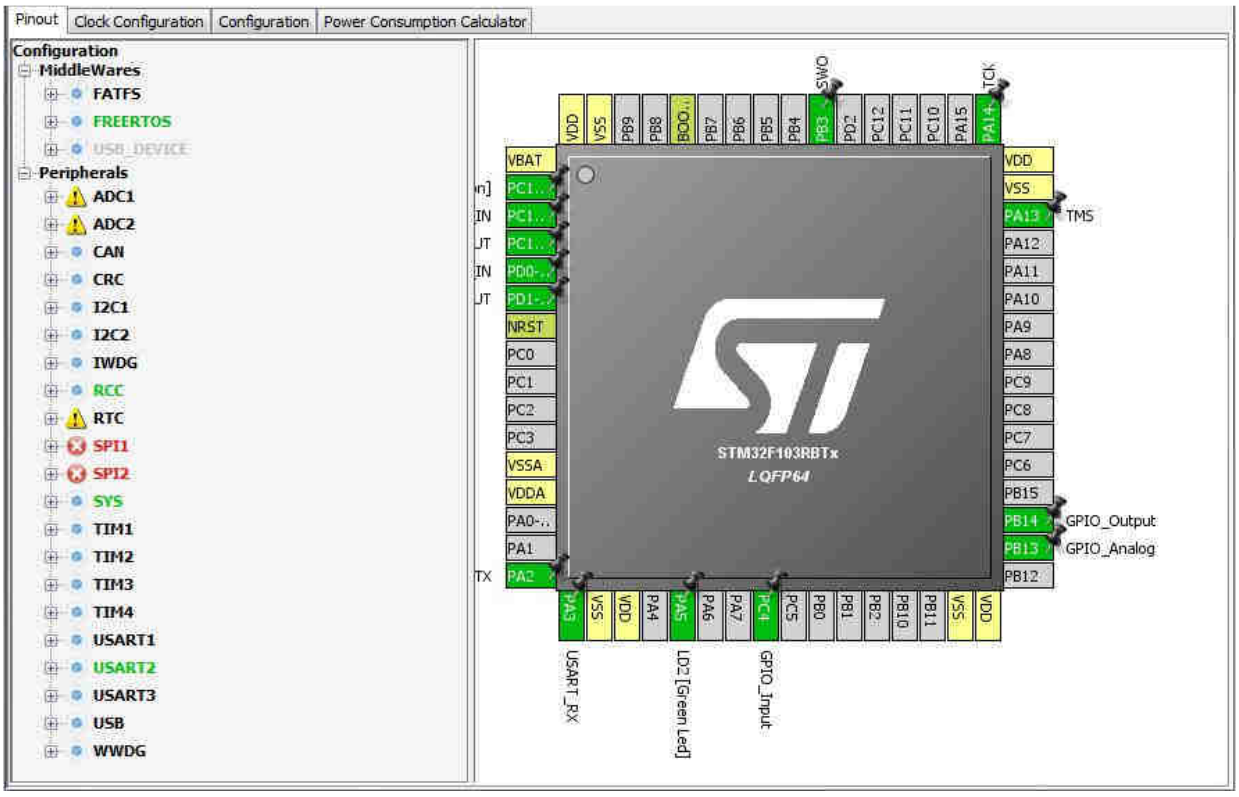
- NUCLEO-F103RB Board
- Keil uVision 5 with the necessary packages for Nucleo boards installed
- STLink USB Driver
- STM32CubeMX

Announcement: check out the updated tutorials
https://github.com/mnemocron/STM32-Tutorial

STM32CubeMX

Generating the config. files from STM32CubeMX.

1. Open STM32CubeMX and open a new project.
2. Select the Nucleo-F103RB from the Borards tab
3. Enable FreeRTOS
4. Set the RCC (HSE & LSE) to Crystal/Ceramic Resonator
5. Enable the USART2 port in Asynchronous mode
6. Set any GPIO to Output or Input (I am using PB13, PB14 and PC4)
7. Go to Project > Generate code
8. Enter a project name and select MDK-ARM V5
9. Generate the code and open the project in Keil uVision



Now let's see what the code generator did

```
187  void MX_GPIO_Init(void)
```

```
188 {
189
190   GPIO_InitTypeDef GPIO_InitStruct;
191
192   /* GPIO Ports Clock Enable */
193   __GPIOC_CLK_ENABLE();
194   __GPIOD_CLK_ENABLE();
195   __GPIOA_CLK_ENABLE();
196   __GPIOB_CLK_ENABLE();
197
198   /*Configure GPIO pin : B1_Pin */
199   GPIO_InitStruct.Pin = B1_Pin;
200   GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
201   GPIO_InitStruct.Pull = GPIO_NOPULL;
202   HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
203
204   /*Configure GPIO pin : LD2_Pin */
205   GPIO_InitStruct.Pin = LD2_Pin;
206   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
207   GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
208   HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
209
210   /*Configure GPIO pin : PC4 */
211   GPIO_InitStruct.Pin = GPIO_PIN_4;
212   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;         // digital Input
213   GPIO_InitStruct.Pull = GPIO_NOPULL;
214   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
215
216   /*Configure GPIO pin : PB13 */
217   GPIO_InitStruct.Pin = GPIO_PIN_13;
218   GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;        // analog Input
219   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
220
221   /*Configure GPIO pin : PB14 */
222   GPIO_InitStruct.Pin = GPIO_PIN_14;
223   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;     // digital Output
224   GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
225   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
226
227 }
```

The concept is simple, on line 190 an init struct is defined, this struct is filled with information. This information will be processed by the HAL library at the function call `HAL_GPIO_Init()`.

Lines 193 to 196 enable the clock for the GPIO ports.

The init struct consists of 4 values that can be set.

1. Pin
- The Pin(s) that are about to be initialised
- e.g. `GPIO_PIN_3` (numbers reach from 0 to 15, or `GPIO_PIN_ALL`)

2. Mode
- The mode of the selected pins (Input / Output / etc.)
- e.g. GPIO_MODE_INPUT
- Possible assignments are the following:

```
GPIO_MODE_INPUT                    floating input
GPIO_MODE_OUTPUT_PP                output push-pull
GPIO_MODE_OUTPUT_OD                output open drain
GPIO_MODE_AF_PP                    alternate function output push-pull
GPIO_MODE_AF_OD                    alternate function output open drain
```

```
GPIO_MODE_AF_INPUT                alternate function input
```

3. Pull
- Pull-up or Pull-down resistors for the specified pins.
- Can be the following:
  GPIO_NOPULL
  GPIO_PULLUP
  GPIO_PULLDOWN

4. Speed
- Specifies the speed for the selected pins
- Can be the following:
  GPIO_SPEED_LOW
  GPIO_SPEED_MEDIUM
  GPIO_SPEED_HIGH

How to add / remove / change GPIO pins

Example shows push-pull output declaration of three GPIO port A pins

It is really not that hard, just fill the init struct with the desired values and call the HAL_GPIO_Init() function with the corresponding GPIO port.

If you need yet another pin with the same specifications and GPIO port as a pin that has already been declared, it is even simpler. A bitwise or masking of the Pin argument with the new pin does the job.

```
221    /*Configure GPIO pin : PA0, PA1 and PA2 */
222    GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2;
223    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
224    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
225    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
226
```

Using a GPIO output inside the program

Changes of the output state of an output pin are written to the `GPIOx_ODR` register (output data register). This works best with masking.

Turning on an output pin

```
1  /* turn on PA0 */
2  GPIOA -> ODR |= GPIO_PIN_0;
```

Turning off an output pin

```
1  /* turn off PA0 */
2  GPIOA -> ODR &= ~GPIO_PIN_0;
```

Toggle an output pins state

```
1  /* toggle PA0 */
2  GPIOA -> ODR ^= GPIO_PIN_0;
```

An output pin can also be set using the integrated HAL library function

```
1  /* set PA0 */
2  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
3
4  /* reset PA0 */
5  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
6
```

Reading a GPIO input inside the program

The current state of an input can be read from its `GPIOx_IDR` register (input data register)
Again, this works best using bit masking.

Reading an input pin

```
1  /* read PC13 */
2  if(GPIOC -> IDR & GPIO_PIN_13)
3  {
4    /* user code */
5  }
```

Or with the HAL library function

```
1  /* read PC13 */
2  if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13))
3  {
4    /* user code */
5  }
```

Document Created by Simon Burkhardt

This tutorial is very basic and might not show the best way to use the STM32 environment.
It still might help you get into the whole HAL philosophy of STM if you are coming from another platform.

Simon Burkhardt
electronical engineering
simonmartin.ch

History:

V1.0	tested the code, created this document
V1.0.1	added contact info and cc0 notice