General Purpose I/O (GPIO)

Prepared by:
Omar Samir

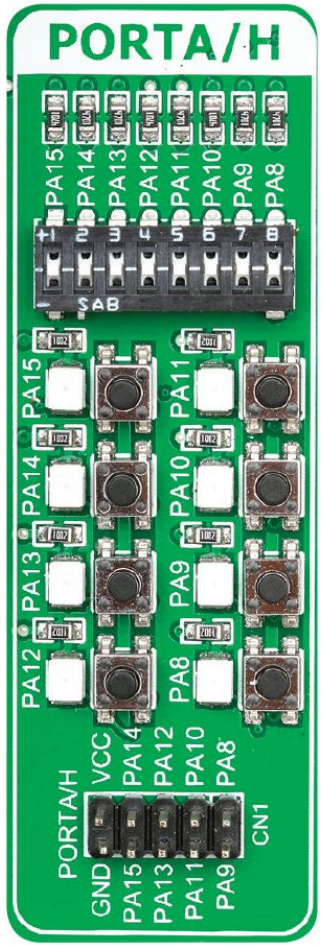
IO Ports

STM32F407VG ARM Microcontroller has 80 Digital IO Pins as follows:

1. PORTA has 16 pins from PA0 to PA15
2. PORTB has 16 pins from PB0 to PB15
3. PORTC has 16 pins from PC0 to PC15
4. PORTD has 16 pins from PD0 to PD15
5. PORTE has 16 pins from PE0 to PE15

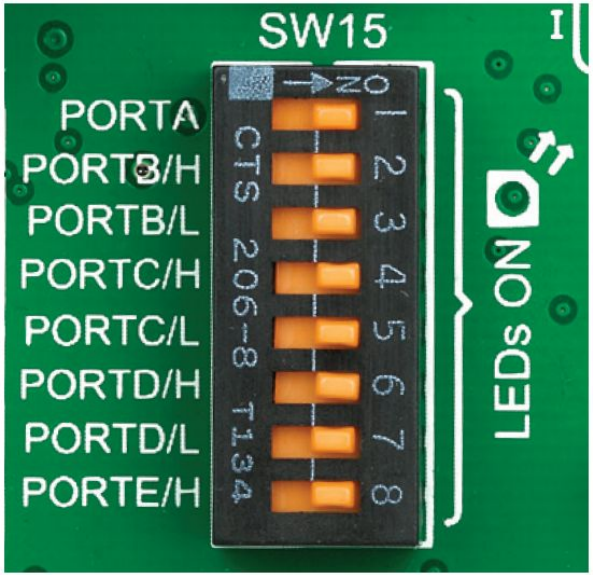
Board Specific

- EasyMx Pro provides a group for each PORT containing the following:
  - LEDs
  - Buttons
  - Headers
  - Tri-state pull up/down switches



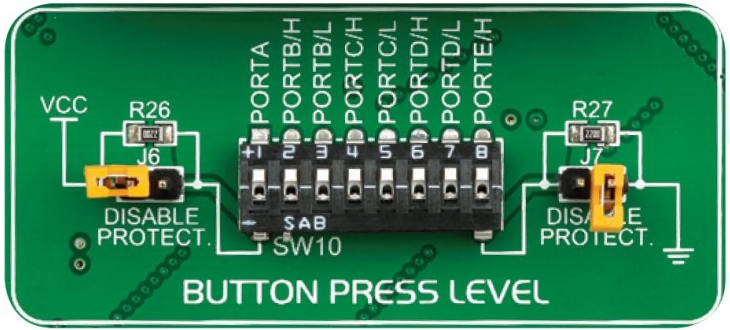
LEDs

- LEDs can be used to see the value of each pin if it is high or low
- To enable LEDs of a specific PORT you need to enable its switch in SW15



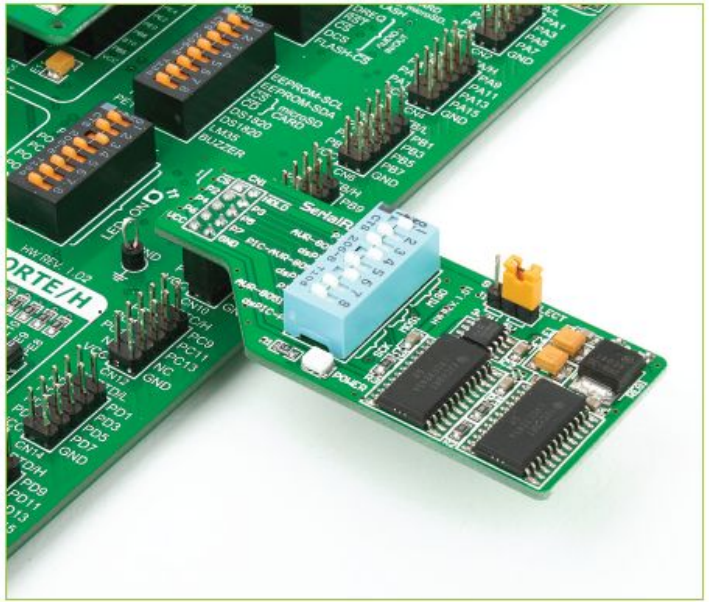
Buttons

- Push buttons can be used to control input pins of the microcontroller
- Tri-state switches (SW10) is used to control the functionality when a push button is pressed as follows:
  - Placing in VCC position will apply logic one when a push button is pressed
  - Placing in GND position will apply logic zero when a push button is pressed
  - Placing in the middle position will make buttons has no effect
- **DON'T EVEN TRY TO REMOVE THE JUMPERS OF SW10 AS YOU MAY DAMAGE THE MCU !!!**



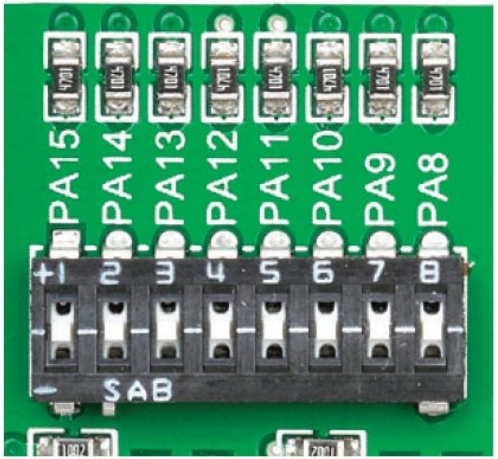
Headers

Headers can be used to transfer signals to or from our board



Tri-state pull up/down switches

- Middle position disables both pull-up and pull-down feature from the PORT pin
- Up position connects the resistor in pull-up state to the selected pin
- Down position connects the resistor in pull-down state to the selected PORT pin.



Digital IO Programming

- Each PORT has some registers associated with it
- Some registers are configuration registers (i.e. to control the direction to be input or output)
- Some are data register (i.e. input and output data registers)
- MikroC for ARM has built in GPIO library which simplifies these operations

Configure a pin as Output

GPIO_Digital_Output (&address, pin_mask);

Address:

- GPIOA_BASE
- GPIOB_BASE
- GPIOC_BASE
- GPIOD_BASE
- GPIOE_BASE

Example:

// Set GPIOC pins 0 and 1 as digital output

GPIO_Digital_Output(&GPIOC_BASE, _GPIO_PINMASK_0 | _GPIO_PINMASK_1);

```
// Pin mask
const _GPIO_PINMASK_0   = 0x1;
const _GPIO_PINMASK_1   = 0x2;
const _GPIO_PINMASK_2   = 0x4;
const _GPIO_PINMASK_3   = 0x8;
const _GPIO_PINMASK_4   = 0x10;
const _GPIO_PINMASK_5   = 0x20;
const _GPIO_PINMASK_6   = 0x40;
const _GPIO_PINMASK_7   = 0x80;
const _GPIO_PINMASK_8   = 0x100;
const _GPIO_PINMASK_9   = 0x200;
const _GPIO_PINMASK_10  = 0x400;
const _GPIO_PINMASK_11  = 0x800;
const _GPIO_PINMASK_12  = 0x1000;
const _GPIO_PINMASK_13  = 0x2000;
const _GPIO_PINMASK_14  = 0x4000;
const _GPIO_PINMASK_15  = 0x8000;
const _GPIO_PINMASK_LOW = 0x00FF;
const _GPIO_PINMASK_HIGH= 0xFF00;
const _GPIO_PINMASK_ALL = 0xFFFF;
```

Configure a pin as Input

GPIO_Digital_Input (&address, pin_mask);

Example:

// Set GPIOC pins 0 and 1 as digital input

GPIO_Digital_Input(&GPIOC_BASE, _GPIO_PINMASK_0 | _GPIO_PINMASK_1);

Output Registers

Each port has its own output register as follows:

- GPIOA_ODR for PORTA
- GPIOB_ODR for PORTB
- GPIOC_ODR for PORTC
- GPIOD_ODR for PORTD
- GPIOE_ODR for PORTE

Example:

```
GPIOA_ODR = 0xFFFF; // output logic one to all pins of PORTA
```

Input Registers

Each port has its own input register as follows:

- GPIOA_IDR for PORTA
- GPIOB_IDR for PORTB
- GPIOC_IDR for PORTC
- GPIOD_IDR for PORTD
- GPIOE_IDR for PORTE

These registers can be used to read inputs from pins.

But wait what about the debouncing period??

Button Library

The Button Library provides routines for detecting button presses and debouncing

unsigned int Button(**unsigned int** *port, **unsigned int** pin, **unsigned int** time, **unsigned int** active_state);

Parameters:

- port: button port address
- pin: input pin number (from 0 to 15)
- time: debouncing period in milliseconds
- active_state: to check for logical zero or one

Example:

Button(&GPIOA_IDR, 0, 1, 1)) // detect logical one on PA0 pin with 1 millisecond debouncing period

Include Button Library

1. From view select Library Manager
2. A window with all libraries will appear
3. Open MikroE Libraries
4. Open System Libraries
5. Check on Button

Requirement 1 (LED Blinking)

It is required to make the LEDs of PORTs D and E blink every 0.5 second (Toggle the LEDs every 0. 5 second).

Hints:

- Use Delay_ms(milliseconds);
- Don’t forget all board specific switches for LEDs and Buttons.

Requirement 2

It is required to make the LEDs of PORTs D and E turn on row by row every 100ms then after they are all on make them turn off row by row every 100ms and repeat the process for infinity.

Hints:

- Try to recognize the pattern

Requirement 3 (Counter)

It is required to use the pins PD0, PD1, PD2, PD3 as a binary counter from 0 to 15. Configure PB0 and PB1 as input pins. PB0 is used as incrementer and PB1 is a decrementer. When the counter achieves 15 then PB0 will do nothing. When the counter achieves 0 then PB1 will do nothing. The transition must happen on the rising edge of the push buttons.