**LAB: GPIO Digital I/O**

**- 7-segment Control & Debugging -**

**I. Overview**

In this lab, we will learn how to control multiples of digital I/O of GPIOs of the MCU. The 7-segment is composed of several LEDs and you will learn how to control each LEDs to show a number.

The objectives of this lab are learn how to

* Read and configure registers of digital GPIO of MCU
* Program firmware to control digital input/output pins
* Debug your program
* Create your own functions for GPIOs

**Preparation**:

* 7-segment, mini-breadboard, 8 - array resistor(330), jumper wire
* You need to read how to connect a 7-segment to an MCU

**II. Pre-Lab**

**A. 7-Segment**

- Circuit diagram of ‘Seven-Segment(Common anode type)’

|  |
| --- |
| 7 segment circuitì ëí ì´ë¯¸ì§ ê²ìê²°ê³¼ |
| (a) Circuit diagram of 7-segmgnet |
|  |
| (b) 5101ASR specification |

- Describe how to connect 7-egment to MCU pins.

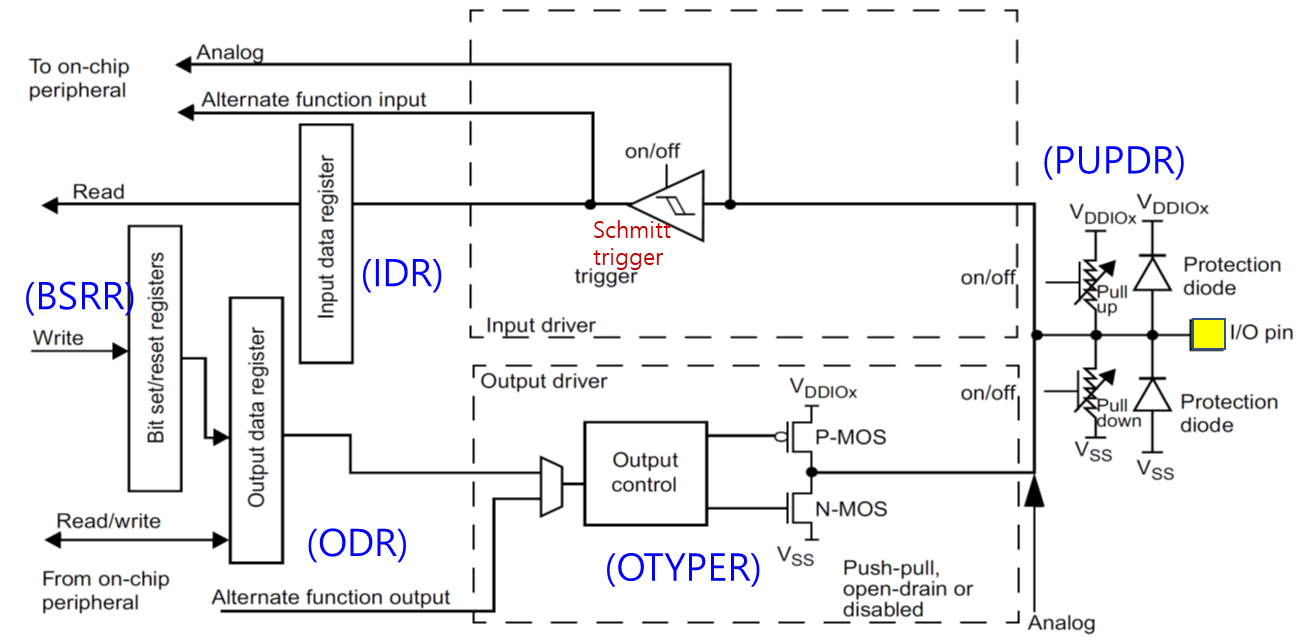
|  |
| --- |
|  |

**B. GPIO Register**

* List GPIO registers for this LAB

|  |  |  |
| --- | --- | --- |
| Type | Register Name | Description |
| GPIO | GPIOx\_MODER | Mode: Output/Input/Analog |
|  | GPIOx\_OTYPER | Output Type: Opendrain/Push-Pull |
|  | GPIOx\_OSPEEDR | Output Speed: |
|  | GPIOx\_PUPDR | Pull-Up Pull-Down: |
|  | GPIOx\_IDR | Input Data Register |
|  | GPIOx\_ODR | Output Data Register |

* Schematic



* Process of GPIO register initiation

|  |
| --- |
| 0. Enable Peripheral Clock (**AHB1ENR** )  1. Configure as Digital Output or Input (**GPIOx\_MODER** )  2. Configure pull-up/down resistors (**PUPDR**)  3. For Output: Configure Output Type (**OTYPE**)  4. For Output: Configure Output Speed (**OSPEEDR**)  5. Read Data or Output Data **(ORD / IDR)** |

**C. Register Setting**

**1. Pin Initialization & Set LED:** PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10  
 / Output / Push-Pull / Fast / No Pull-Up & No Pull-Down

* **GPIOx\_MODER:** Output

|  |
| --- |
| *Register map goes here* |

|  |  |  |
| --- | --- | --- |
| **Port/Pin** | **Description** | **Set output** |
| Port A Pin 5 | Clear Pin5 mode | GPIOA🡪MODER &=~(3<<(5\*2)) |
| Port A Pin 5 | Set Pin5 mode = Output | GPIOA🡪MODER |=\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin 6 | Clear Pin6 mode | GPIOA🡪MODER &=~\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin 6 | Set Pin6 mode = Output | GPIOA🡪MODER |=\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin Y | Clear PinY mode | GPIOA🡪MODER &=~\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin Y | Set PinY mode = Output | GPIOA🡪MODER |=\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin 5~9 | Clear Pin5~9 mode | GPIOA🡪MODER &=~\_\_\_\_\_\_\_\_\_\_\_ |
|  | Set Pin5~9 mode = Output | GPIOA🡪MODER |=\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port X Pin Y | Clear Pin Y mode | GPIOX🡪MODER &=~\_\_\_\_\_\_\_\_\_\_\_ |
|  | Set Pin Y mode = Output | GPIOX🡪MODER |=\_\_\_\_\_\_\_\_\_\_\_\_ |

* **GPIOx\_OTYPER:** Push-Pull

|  |
| --- |
| *Register map goes here* |

|  |  |  |
| --- | --- | --- |
| **Port/Pin** | **Description** | **Set output** |
| Port A Pin5 | Set Pin5 otype=push-pull | GPIOA🡪OTYPER =\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A PinY | Set PinY otype=push-pull | GPIOA🡪 OTYPER =\_\_\_\_\_\_\_\_\_\_\_\_ |

* **GPIOx\_OSPEEDR:** Fast

|  |
| --- |
| *Register map goes here* |

|  |  |  |
| --- | --- | --- |
| **Port/Pin** | **Description** | **Set output** |
| Port A Pin5 | Set Pin5 ospeed=Fast | GPIOA🡪OSPEEDR =\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A PinY | Set PinY ospeed=Fast | GPIOA🡪 OSPEEDR =\_\_\_\_\_\_\_\_\_\_\_\_ |

* **GPIOx\_PUPDR:** no pull-up no pull-down

|  |
| --- |
| *Register map goes here* |

|  |  |  |
| --- | --- | --- |
| **Port/Pin** | **Description** | **Set output** |
| Port A Pin 5 | Set Pin5 PUPD=no pullup/down | GPIOA🡪OTYPER =\_\_\_\_\_\_\_\_\_\_\_\_ |
| Port A Pin Y | Set PinY PUPD=no pullup/down | GPIOA🡪 OTYPER =\_\_\_\_\_\_\_\_\_\_\_\_ |

**2. Pin Initialization & Read PushButton:** Port C Pin 13 / Input // Pull-Up

* **GPIOx\_MODER:** Input

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

GPIOC\_MODER |=

* **GPIOx\_PUPDR:** pull-up

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

GPIOC\_PUPDR =

**III. Tutorial**

**A. Keil compiler**

* Follow TA’s tutorial to connect 7-segment to designated pins
* Create a simple program to show ‘0’ on 7-segment. (TA code explanation). Call the project as “LAB\_7Segment.”

|  |  |
| --- | --- |
|  | #include "stm32f4xx.h"  #include "myGPIO.h"  #include "myRCC.h"  int main(void) {  RCC\_HSI\_init();    GPIO\_init(GPIOA, 5, OUTPUT); // a  GPIO\_init(GPIOA, 6, OUTPUT); // b  GPIO\_init(GPIOA, 7, OUTPUT); // c  GPIO\_init(GPIOB, 6, OUTPUT); // d  GPIO\_init(GPIOC, 7, OUTPUT); // e  GPIO\_init(GPIOA, 9, OUTPUT); // f  GPIO\_init(GPIOA, 8, OUTPUT); // g  GPIO\_init(GPIOB, 10, OUTPUT); // dp    while(1){  int ON = 0b00000011;  GPIO\_write(GPIOA, 5, (ON >> 7) & 1);  GPIO\_write(GPIOA, 6, (ON >> 6) & 1);  GPIO\_write(GPIOA, 7, (ON >> 5) & 1);  GPIO\_write(GPIOB, 6, (ON >> 4) & 1);  GPIO\_write(GPIOC, 7, (ON >> 3) & 1);  GPIO\_write(GPIOA, 9, (ON >> 2) & 1);  GPIO\_write(GPIOA, 8, (ON >> 1) & 1);  GPIO\_write(GPIOB, 10, ON & 1);  }  } |

* To compile the program, click on ‘**Compile’** button
* Check if 7-segment is showing the number ‘0’.

**B. Debugging**

* Copy and paste the given source code on ‘**main.c**’(LAB\_GPIO\_LED\_Button). This is an example code of turning one LED *without* button input (on next page).
* You will learn how to use debugging mode to check how the registers change
* See the ‘**Tutorial\_Debugging’** for instructions

|  |  |
| --- | --- |
|  | #include "stm32f4xx.h"  #define LED\_PIN 5  int main(void){  RCC->CR |= RCC\_CR\_HSION;    while((RCC->CR & RCC\_CR\_HSIRDY) == 0) {;}    RCC->CFGR &= ~(RCC\_CFGR\_SW);  RCC->CFGR |= RCC\_CFGR\_SW\_HSI;    while((RCC->CFGR & RCC\_CFGR\_SWS) != 0){;}    RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN;    GPIOA->MODER &= ~(3 << 2\*LED\_PIN);  GPIOA->MODER |= 1<<(2\*LED\_PIN);    GPIOA->OSPEEDR &= ~(3<<2\*LED\_PIN);  GPIOA->OSPEEDR |= 2<<2\*LED\_PIN;    GPIOA->OTYPER &= ~(1<<LED\_PIN);    GPIOA->PUPDR &= ~(3<<2\*LED\_PIN);    while(1){  GPIOA->ODR |= 1 << LED\_PIN;  }  } |

**IV. Exercise/Demo**

1. **Create a code to count up a number from ‘0’ to ‘9’ on 7-segment as button is pressed. It should roll back to ‘0’ after number ‘9’.** 
   * Button Input: Port C Pin 13 / Pull-up
   * Digital Output: PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10 / Push-Pull / Low Speed
   * Draw a flowchart or pseducode for this program

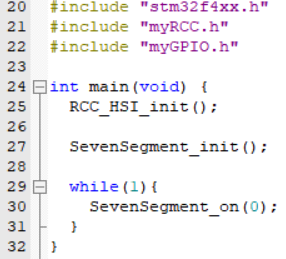
|  |
| --- |
|  |

* + Debug your program and check how the registers and variables change

1. **Create user defined function****s for GPIO control including 7-segment**

* Create functions to control 7-segments. Try to use *arrays or structures* with 7-segment control functions. One function should be display the character of the input argument.

|  |  |
| --- | --- |
| **Source File** | **Function** |
| myRCC.h  myRCC.c | void RCC\_HSI\_init(); |
| myGPIO.h  myGPIO.c | void GPIO\_init(GPIO\_TypeDef \*port, int pin, int mode) |
| void GPIO\_mode(GPIO\_TypeDef \*port, int pin, int mode) |
| void GPIO\_otype(GPIO\_TypeDef \*port, int pin, int type) |
| void GPIO\_ospeed(GPIO\_TypeDef \*port, int pin, int speed) |
| void GPIO\_pupd(GPIO\_TypeDef \*Port, int pin, int pupd) |
| void GPIO\_write (GPIO\_TypeDef \*port, int pin, int output); |
| int GPIO\_read(GPIO\_TypeDef \*port, int pin) |
| void SevenSegment\_init () |
| void SevenSegment\_on (uint8\_t num) |

****

**V. Questions**

1. What are cathode/anode in 7-segment? Which one would turn ON when output pin is ‘High’?
2. How can you control multiple 7-segments with minimum number of GPIO pins?

**V. Appendix**