LAB: Digital In/Out – 7-Segment Display

Date: 2021-10-14

Name(ID):Park JeongWoo

Partner Name:Lee JunGi

I. Introduction

In this lab, you are required to create a simple program to control a 7-segment display to show a decimal number (0~9).

●Hardware

NUCLEO-F411RE

One 7-segment display(5101ASR), array resistor (330 ohm), breadboard

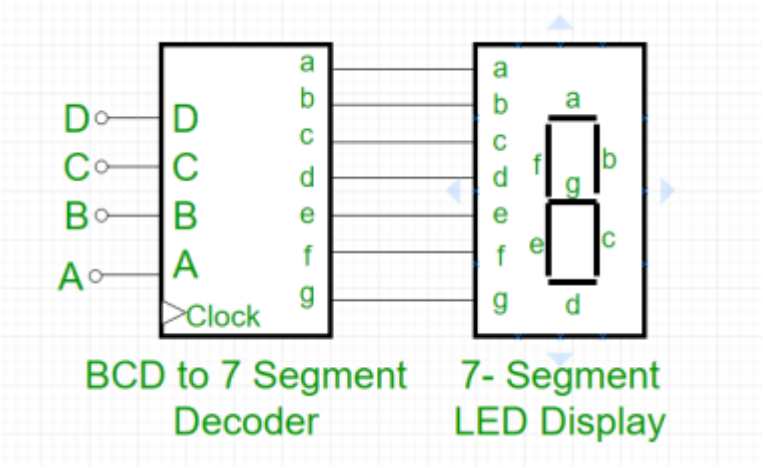
●Software

Keil uVision IDE, CMSIS, EC_HAL

II. Procedure

A. 7-Segment

Popular BCD 7-segment decoder chip are 74LS47, CD4511. Here, we are going to make the 7- segment decoder by the MCU programming.

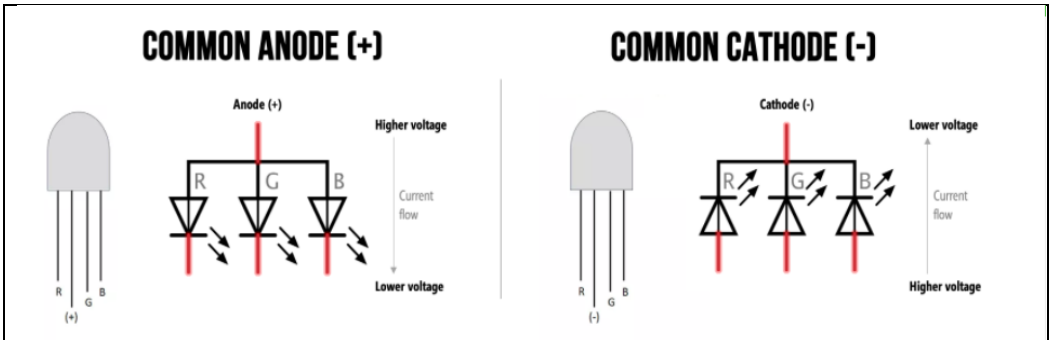


- Discussion

1) Draw the truth table for the BCD 7-segment decoder.

Decimal Digit	Input lines				Output lines								Display pattern
	A	B	C	D	a	b	c	d	e	f	g	dp	
0	0	0	0	0	1	1	1	1	1	1	0	0	0
1	0	0	0	1	0	1	1	0	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	0	2
3	0	0	1	1	1	1	1	1	0	0	1	0	3
4	0	1	0	0	0	1	1	0	0	1	1	0	4
5	0	1	0	1	1	0	1	1	0	1	1	0	5
6	0	1	1	0	1	0	1	1	1	1	1	0	6
7	0	1	1	1	1	1	1	0	0	1	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	0	8
9	1	0	0	1	1	1	1	1	0	1	1	0	9
x	x	x	x	x	x	x	x	x	x	x	x	x	No display

2) What are the common cathode and common anode of 7-segment



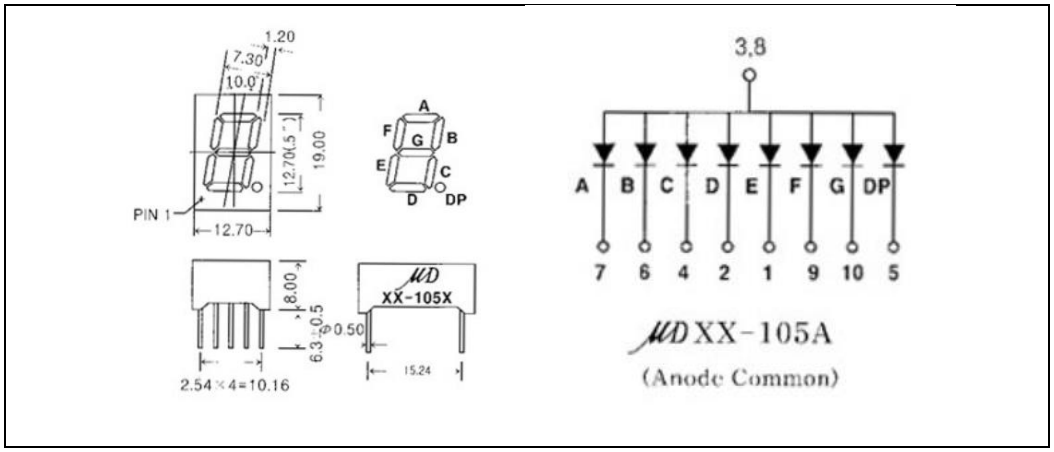
Ref. https://makeabilitylab.github.io/physcomp/arduino/rgb-led.html

As shown in the figure above, Common cathode refers to a method of applying GND(-) to a common pin And applying Vcc(+) to each pin. Common Anode refers to a method of applying Vcc(+) to a common pin And applying GND(-) to each pin.

3) This is common anode 7-segment. Does the LED turn on when output pin from MCU is 'HIGH'?

In the common anode, when 'HIGH' is applied to the common pin, the output pin does not light. Therefore, If i want to turn on the light, input sinal should be 'LOW'

4) Find out how to connect a 7-segment to MCU pins with current limiting resistors.



According to the data sheet above, eight output pins were selected, and each output pin was connected to match the location of the LED.

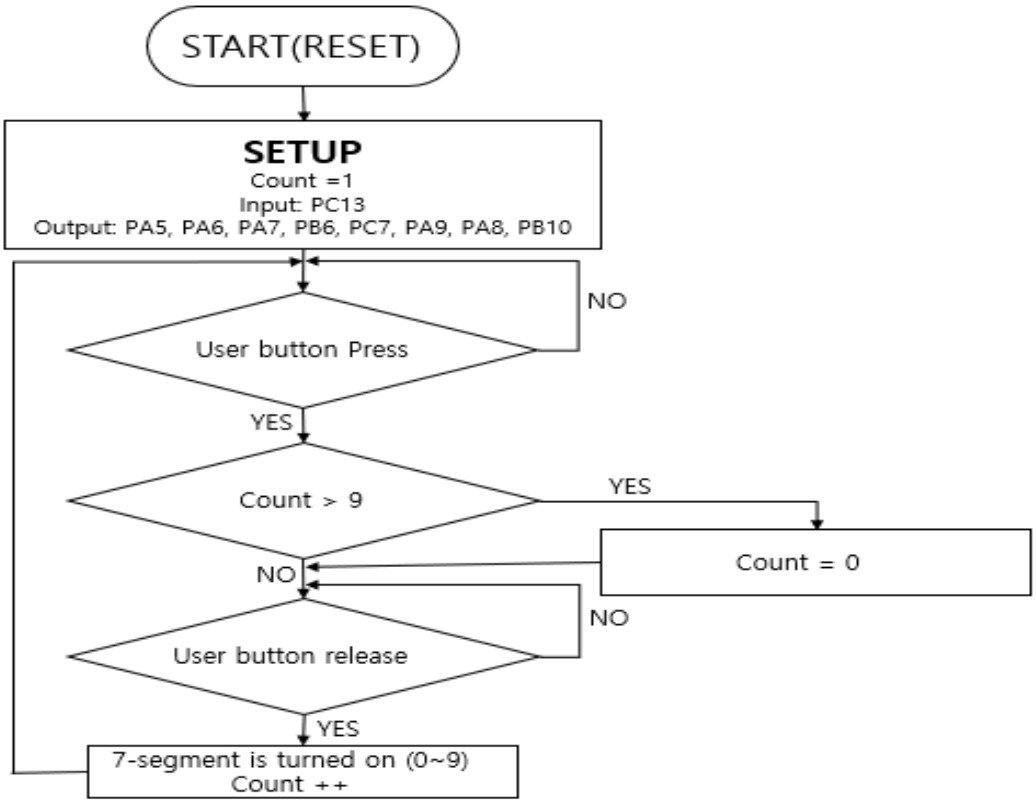
Resistance is needed to prevent overcurrent from flowing through the LED, but it is cumbersome to plug in the resistance one by one, so I use the array register.

B. Configuration – 7-Segment Display

- Observation of the Output

When the reset button is pressed, 0 is output on the 7-segement display and then if presse the user button, it increases sequentially from 1 to 9.

- Flow Chart



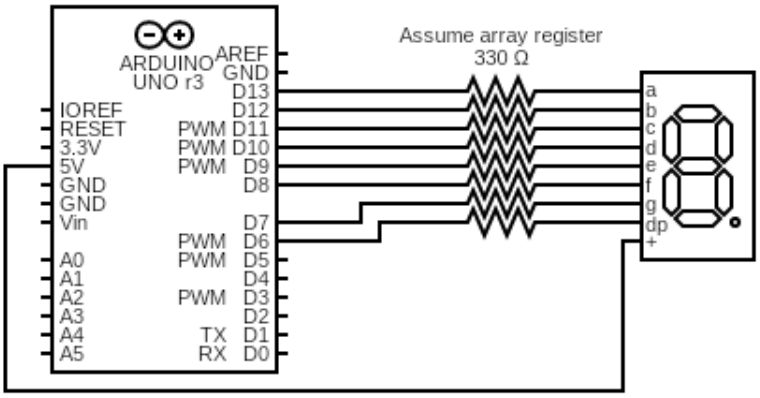
- Configure input and output pins

Digital In: Button	Digital Out: LED
GPIOC, Pin 13 Digital Input Set PULL-UP	PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10 Digital Output Push-pull No Pull-up Pull down Fast

- Fill in the table

Port/Pin	Description	Register setting
Port A pin 5	Clear Pin5 mode	GPIOA→MODER &= ~(3<<(5*2))
Port A pin 5	Set Pin5 mode = Output	GPIOA→MODER |= (1<<(5*2))
Port A pin 6	Clear Pin6 mode	GPIOA→MODER &= ~(3<<(6*2))
Port A pin 6	Set Pin6 mode = Output	GPIOA→MODER |= (1<<(6*2))
Port A pin Y	Clear PinY mode	GPIOA→MODER &= ~(3<<(Y*2))
Port A pin Y	Set PinY mode = Output	GPIOA→MODER |= (1<<(Y*2))
Port A pin 5~9	Clear Pin5~9 mode	GPIOA→MODER &= ~(0011 1111 1111(2)<<(5*2))
	Set Pin5~9 mode = Output	GPIOA→MODER |= (0001 0101 0101(2)<<(5*2))
Port X pin Y	Clear Pin Y mode	GPIOX→MODER &= ~(3<<(Y*2))
	Set Pin Y mode = Output	GPIOX→MODER |= (1<<(Y*2))
Port A pin 5	Set Pin5 otype=push-pull	GPIOA→OTYPER &= ~(1<<5)
Port A pin Y	Set PinY otype=push-pull	GPIOA→OTYPER &= ~(1<<Y)
Port A pin 5	Set Pin5 ospeed=Fast	GPIOA→OSPEEDR &= ~(3<<(5*2)) GPIOA→OSPEEDR |= (2<<(5*2))
Port A pin Y	Set PinY ospeed=Fast	GPIOA→OSPEEDR &= ~(3<<(Y*2)) GPIOA→OSPEEDR |= (2<<(Y*2))
Port A pin 5	Set Pin5 PUPD=no pullup/down	GPIOA→OTYPER &= ~(3<<(5*2))
Port A pin Y	Set PinY PUPD=no pullup/down	GPIOA→OTYPER &= ~(3<<(Y*2))

- Circuit Diagram



The 7-segment i used is S-5101ASR. But it is incompatible in the circuit drawing site. So I cannot display the pin to which Vcc is applied, so it is expressed as shown in the figure above.

- Source code

LAB_GPIO_Digitalinout_Sevensegment.cpp

```
/**********************************************************************************
* @author    JeongWoo Park
* @Mod       2021-10-07 by JeongWoo Park
* @brief   Embedded Controller:  LAB Digital In/Out  - 7-segment Display
*                - 7-segment display to show a decimal number (0~9)
*
**********************************************************************************
*/

#include "stm32f4xx.h"
#include "ecRCC.h"
#include "ecGPIO.h"

void setup(void);


int main(void) {

  uint32_t count =1;

  // Initialiization --------------------------------------------------------
   setup();

  // Inifinite Loop ----------------------------------------------------------
   while(1){

    if(GPIO_read(GPIOC, BUTTON_PIN) == 0){

      if(count > 9) count = 0;

    while(GPIO_read(GPIOC, BUTTON_PIN) == 0) {;}
```

```
32
33          sevensegment_decode(count % 10);
34          count++;
35
36
37      }
38    }
39  }
40
41
42  // Initialiization
43  void setup(void)
44  {
45    RCC_HSI_init();
46    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
47    sevensegment_init();
48
49    sevensegment_decode(0);
50  }
```

C. Create EC_HAL functions

Include File	Function
ecGPIO.h,c	Void sevensegment_init(void);
	Void sevensegment_decoder(uint8 num);

- ecGPIO.h

```
// pin
#define     PA5             5
#define     PA6             6
#define     PA7             7
#define     PB6             6
#define     PC7             7
#define     PA9             9
#define     PA8             8
#define     PB10            10
```

```
63  void sevensegment_init(void);
64
65  void sevensegment_decode(int number);
```

ecGPIO.c : See Appendix

- Documenation of Library

sevensegment_init()

Initializes 7-segment 8pins

```
void sevensegment_init();
```

This function includes others functions

ex) GPIO_init, GPIO_pudr, GPIO_otype, GPIO_ospeed

so, it define register's state

Example code

```
void sevensegment_init(); // setting registers
```

sevensegment_decode()

According to Input signal, change the 7-segment display

```
void sevensegment_decode(int number);
```

Parameters

- number: the number shown on the display (0~9)

Example code

```
void sevensegment_init(5); // Appear to number 5 on the 7-segment
```

D. Conclusion & Trouble Shooting

1) Conclusion

This experiment is more complex than the previous experiment of turning on three LEDs. I created Decoder in software. the Decoder is created according to digital logic to control 7-segment. I changed the number of 7-segment each time a button was pressed. I learned the principle and operation of the 7 segments used in the experiment.

2) Trouble Shooting

Q. If I define the output of the 7-segment one by one, the readability is poor.

A. Use a two-dimensional array to define the output value in rows

E. Appendix

- Demo_Link

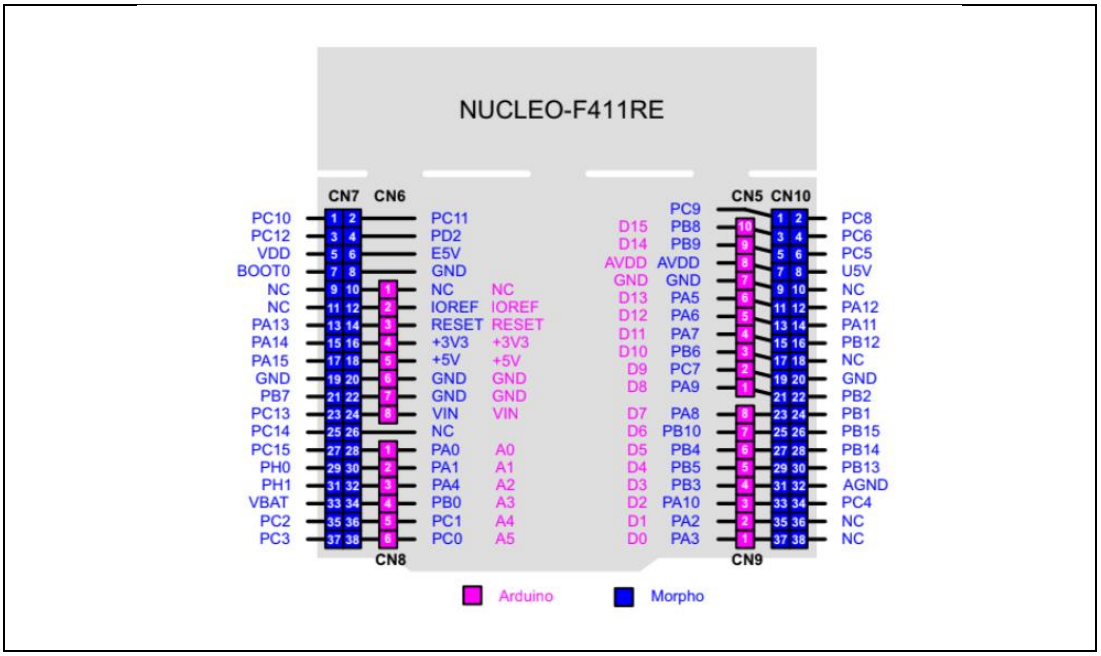
  https://youtu.be/hIsWKrHrX1o

- ecGPIO.c

```
66  void sevensegment_init()
67  {
68    GPIO_init(GPIOA, PA5, OUTPUT);
69    GPIO_init(GPIOA, PA6, OUTPUT);
70    GPIO_init(GPIOA, PA7, OUTPUT);
71    GPIO_init(GPIOB, PB6, OUTPUT);
72    GPIO_init(GPIOC, PC7, OUTPUT);
73    GPIO_init(GPIOA, PA9, OUTPUT);
74    GPIO_init(GPIOA, PA8, OUTPUT);
75    GPIO_init(GPIOB, PB10, OUTPUT);
76
77
78
79    // Digital in ------------------------------------------------------------
80    GPIO_pudr(GPIOC, BUTTON_PIN, PULL_UP);
81
82    // Digital out -----------------------------------------------------------
83    GPIO_setting(GPIOA, PA6, FAST_SPEED, PUSH_PULL, NO_PUPD);
84    GPIO_setting(GPIOA, PA7, FAST_SPEED, PUSH_PULL, NO_PUPD);
85    GPIO_setting(GPIOB, PB6, FAST_SPEED, PUSH_PULL, NO_PUPD);
86    GPIO_setting(GPIOC, PC7, FAST_SPEED, PUSH_PULL, NO_PUPD);
87    GPIO_setting(GPIOA, PA9, FAST_SPEED, PUSH_PULL, NO_PUPD);
88    GPIO_setting(GPIOA, PA8, FAST_SPEED, PUSH_PULL, NO_PUPD);
89    GPIO_setting(GPIOB, PB10, FAST_SPEED, PUSH_PULL, NO_PUPD);
90
91  }
```

```
93  void sevensegment_decode(int number)
94  {
95     uint32_t segment_value [11][8]={
96                        {0,0,0,0,0,0,1,1},          //zero
97                        {1,0,0,1,1,1,1,1},          //one
98                        {0,0,1,0,0,1,0,1},          //two
99                        {0,0,0,0,1,1,0,1},          //three
100                       {1,0,0,1,1,0,0,1},          //four
101                       {0,1,0,0,1,0,0,1},          //five
102                       {0,1,0,0,0,0,0,1},          //six
103                       {0,0,0,1,1,1,1,1},          //seven
104                       {0,0,0,0,0,0,0,1},          //eight
105                       {0,0,0,0,1,0,0,1},          //nine
106                       {1,1,1,1,1,1,1,0}           //dot
107                       };
108
109    GPIO_write(GPIOA, PA5, segment_value  [number][0]);
110    GPIO_write(GPIOA, PA6, segment_value  [number][1]);
111    GPIO_write(GPIOA, PA7, segment_value  [number][2]);
112    GPIO_write(GPIOB, PB6, segment_value  [number][3]);
113    GPIO_write(GPIOC, PC7, segment_value  [number][4]);
114    GPIO_write(GPIOA, PA9, segment_value  [number][5]);
115    GPIO_write(GPIOA, PA8, segment_value  [number][6]);
116    GPIO_write(GPIOB, PB10, segment_value [number][7]);
117
118
119 }
120
```

- PIN-MAP



- STM32F411xC/E Reference

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..E and H)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..E and H)

Address offset: 0x04

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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
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 **OTy**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..E and H)

Address offset: 0x08

Reset values:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **OSPEEDRy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: Fast speed

11: High speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..E and H)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved