Embedded System

Training 1

August 30, 2022,

Overview

In our first training section, we are going to cover the following stuff:

1. The basic grammar of the C language
    a. types/struct/pointer/;/if-else/for/while/switch/memcpy/strcpy…
2. The basic knowledge of stm32
    a. what is stm32?
    b. stm32f427iih6 & schematics
    c. registers
    d. what is HAL
        i. Hal functions
    e. how can we program it?
    f. GPIO
        i. push pull / open drain
        ii. pull up/ pull down

- Example 1: light up led on the board
- homework 1: light up each of the led one by one in a sequence
- Example2: use a button to control the LEDs

The basic knowledge of the C language

Types:

Keyword Used	Data Type
int	Integer
float	Floating-point
void	Void
char	Character
double	Double

Arrays:

Declaring Arrays

```
type arrayName [ arraySize ];
double balance[10];
```

Initializing Arrays

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
double balance[ ] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

accessing Arrays

```
double salary = balance[9];
balance[4] = 50.0;
```

if-else

```
if (test expression1) {
   // statement(s)
}
else if(test expression2) {
   // statement(s)
}
else if (test expression3) {
   // statement(s)
}
.
.
else {
   // statement(s)
}
```

for loop

```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}

for (int i = 0, i<10,i++){

}
```

while loop

```
while (testExpression) {
  // the body of the loop
}
```

pointer

```
#include <stdio.h>
int main()
{
```

```
   int* pc, c; //pc is a pointer and c is a normal int variable

   c = 22;
   printf("Address of c: %p\n", &c);
   printf("Value of c: %d\n\n", c);  // 22

   pc = &c;
   printf("Address of pointer pc: %p\n", pc);
   printf("Content of pointer pc: %d\n\n", *pc); // 22

   c = 11;
   printf("Address of pointer pc: %p\n", pc);
   printf("Content of pointer pc: %d\n\n", *pc); // 11

   *pc = 2;
   printf("Address of c: %p\n", &c);
   printf("Value of c: %d\n\n", c); // 2
   return 0;
}
```

struct

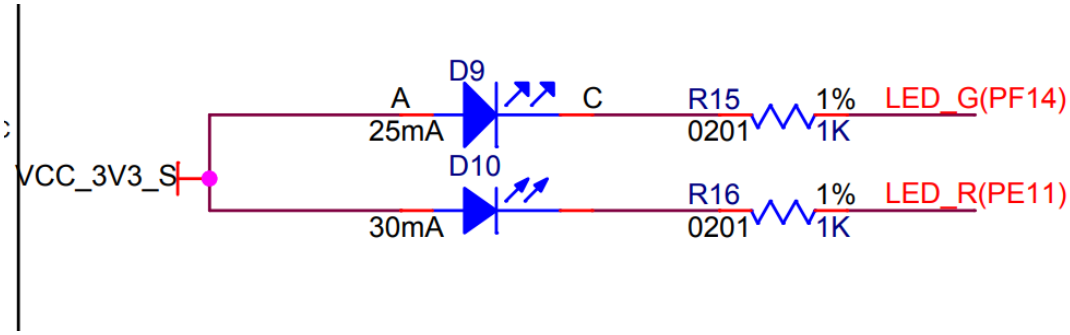
```
struct structureName {
  dataType member1;
  dataType member2;
  ...
};

struct Person {
  char name[50];
  int citNo;
  float salary;
};
```

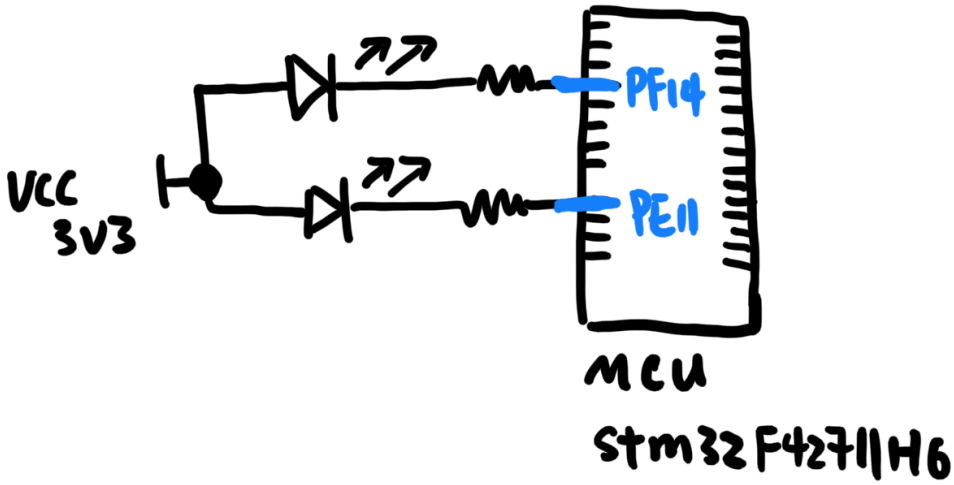
How to light up a LED?

Basic Idea:

Schematics



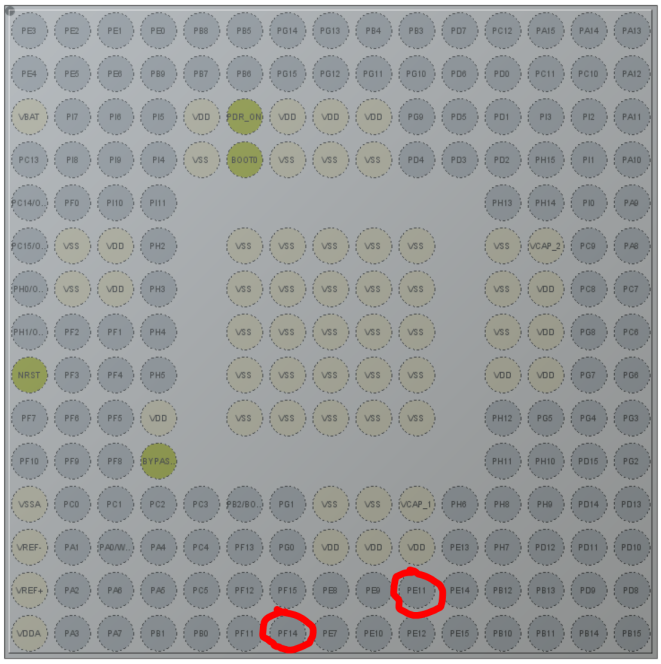
This is a LED schematic from A board. We can see the LED has a 3.3V so if we set the voltage of the pin PF14 and PE11 low, there will be a voltage drop in the circuit, and LED_G and LED_R will light up.

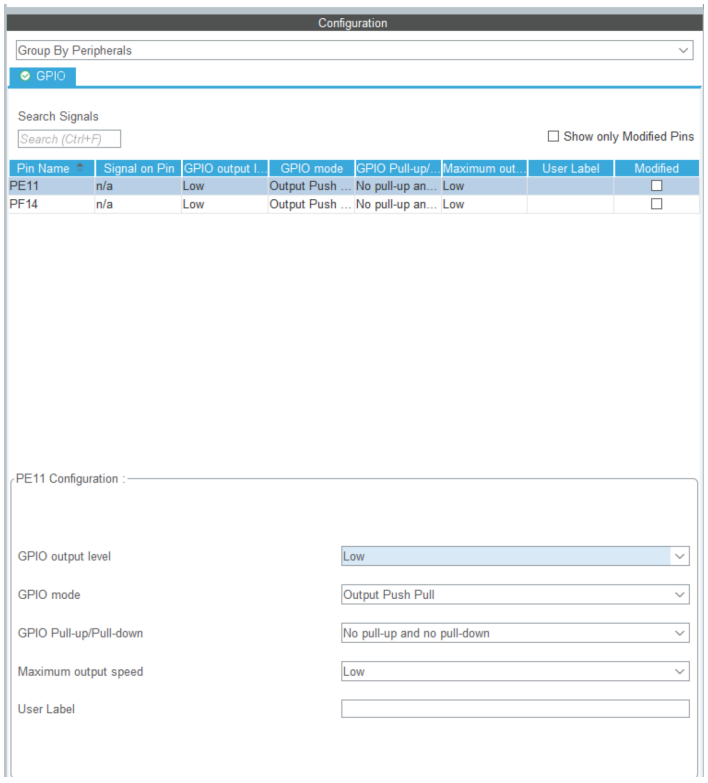


How can we do it?

CubeMX:

6


PE11
PF14

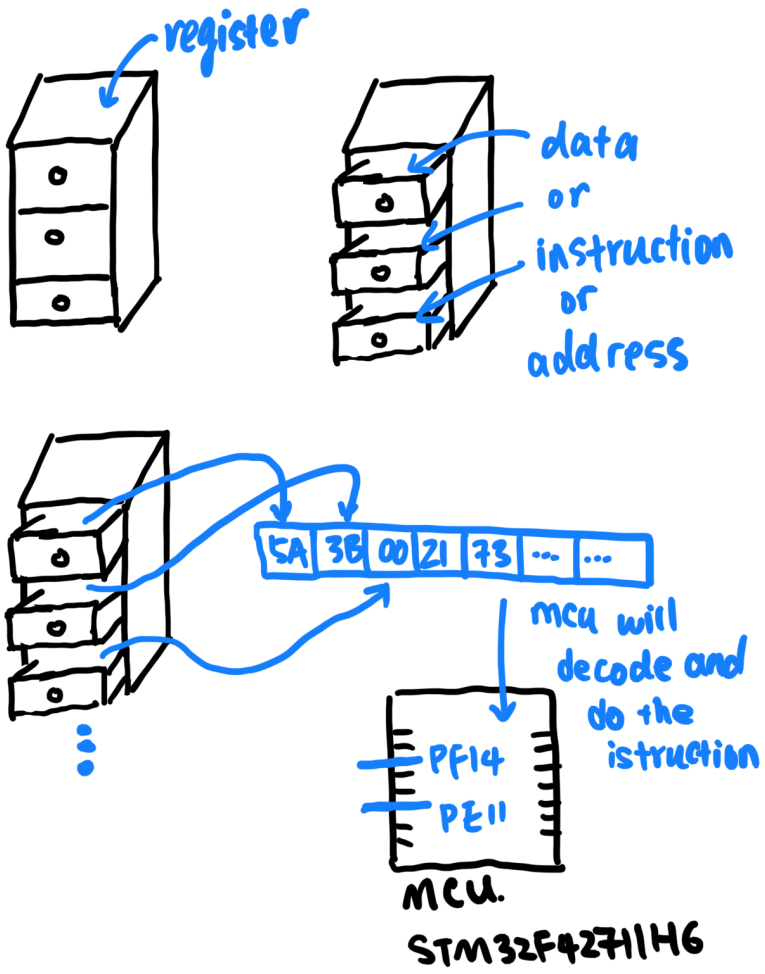

Configuration
Group By Peripherals
GPIO
Search Signals
Search (Ctrl+F)
Show only Modified Pins
Pin Name
Signal on Pin
GPIO output l...
GPIO mode
GPIO Pull-up/...
Maximum out...
User Label
Modified
PE11 n/a Low Output Push ... No pull-up an... Low
PF14 n/a Low Output Push ... No pull-up an... Low
PE11 Configuration :
GPIO output level
Low
GPIO mode
Output Push Pull
GPIO Pull-up/Pull-down
No pull-up and no pull-down
Maximum output speed
Low
User Label

Output Level: set it to low

mode: push pull / open drain

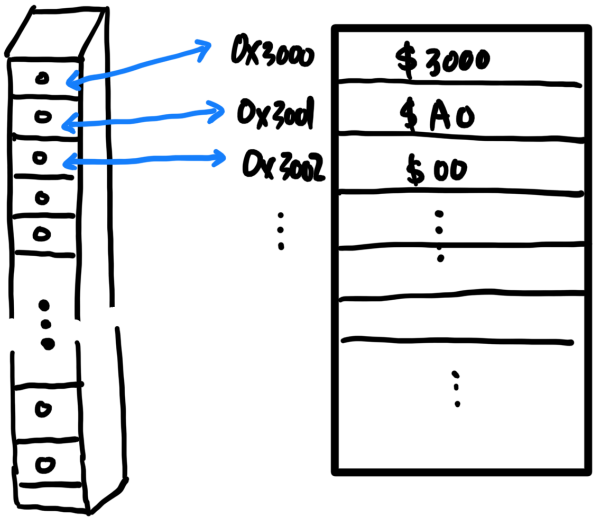
What is a Register:

A processor register (CPU register) is one of a small set of data holding places that are part of the computer processor. A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters).



The Address of the Register:

Each register has its own address. The reference manual has information about the register map.



Basically, if you put specific data in the register, the microprocessor will do all the work for you.

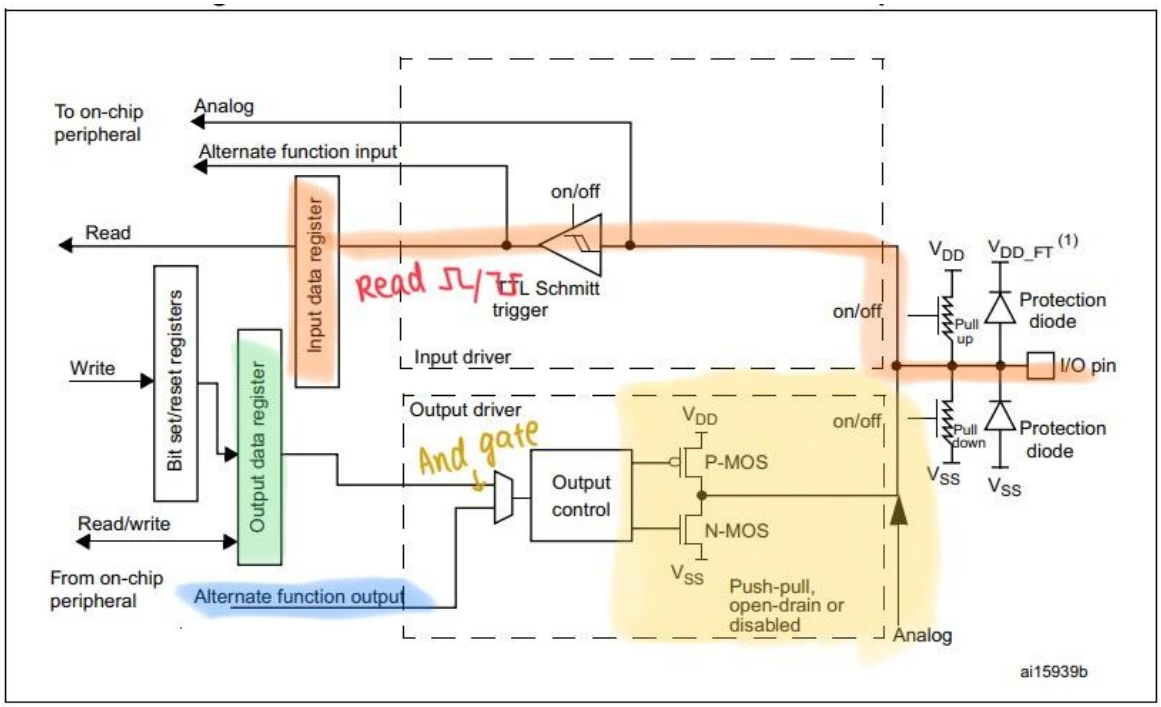
GPIO (General-purpose input/output):

It is the digital signal pin on an integrated circuit or electronic circuit board which may be used as an input or output.

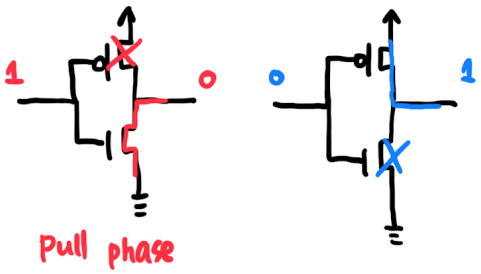
0x4002 3000 - 0x4002 33FF	CRC	AHB1	Section 4.4.4: CRC register map on page 115
0x4002 2800 - 0x4002 2BFF	GPIOK		Section 8.4.11: GPIO register map on page 287
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		Section 8.4.11: GPIO register map on page 287
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		

Push-pull output:

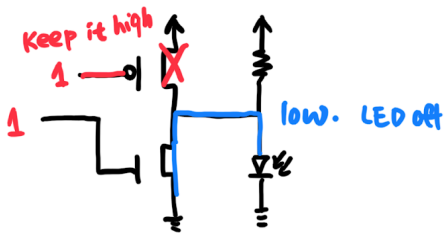
1. find its register address. Reference Manual page 65.

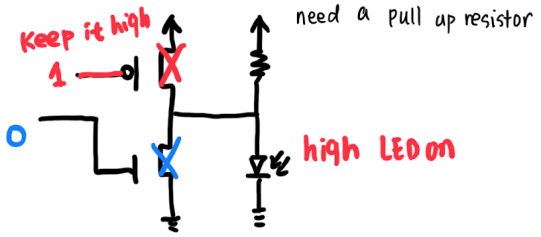


Push-Pull



Open Drain

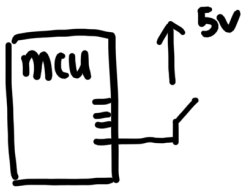




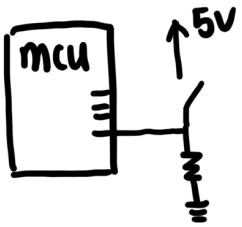
	Push - Pull	Open Drain
Advantage	• High Operating Speed • Stable • low power	• can switch higher or lower voltage than Vdd supply • tie together for multiple-OR functionality.
Disadvantage:	• not design to drive external loads, • can't combine Vout for multiple sensors onto a common bus.	• High power consumption than push-pull • Slower switching

pull up / pull down.

pull down resistor

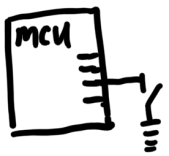


Switch	voltage	
close	5v	not good
open	?	

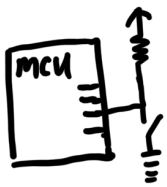


Switch	voltage
close	5v
open	0v

Pull up Resistor



Switch	voltage	
close	0v	not good
open	?	



Switch	voltage
close	0v
open	5v

Write to Register

0x4002 4000 - 0x4002 4FFF	BKPSRAM		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		Sec
0x4002 3800 - 0x4002 3BFF	RCC		Sec
0x4002 3000 - 0x4002 33FF	CRC	AHB1	Sec
0x4002 2800 - 0x4002 2BFF	GPIOK		Sec
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		

1. Enable AHB1 clock
    a. base address (section2): 0x40023800
    b. shift address: 0x30
    c. address: 0x40023800 + 0x30 = 0x40023830
2. Configure GPIO registers. PE11

Table 35. Port bit configuration table(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

PE11: GPIO port register 10

GPIOE base address: 0x4002 1000

mode offset: 0x00

output data offset: 0x14