ELEC 3300 – Tutorial for LAB6

Department of Electronic and Computer Engineering
HKUST
by WU Chi Hang 

Serial and Parallel Communication

- What devices in the computer uses serial/parallel communication ?
- How do you define whether the communication between devices is called
  - Serial ? Parallel ?
- Most important is that it relates to time…
- Let's take an example …
  1. Google "two-time" … Chinese Phrase is…
     - Serial ? Parallel ? At the same time ?
  2. Eat, Vomit ? …
     - Serial ? Parallel ? At the same time ?

Go back to your LAB4 Task 5

If both K1 AND K2 are pressed together, servo will stay at the middle.

```
GPIO_PinState K1, K2;
      K1 = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1);
      K2 = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);

      if ((K1==GPIO_PIN_SET) && (K2==GPIO_PIN_SET)) {
            // Set servo to middle
      }
```

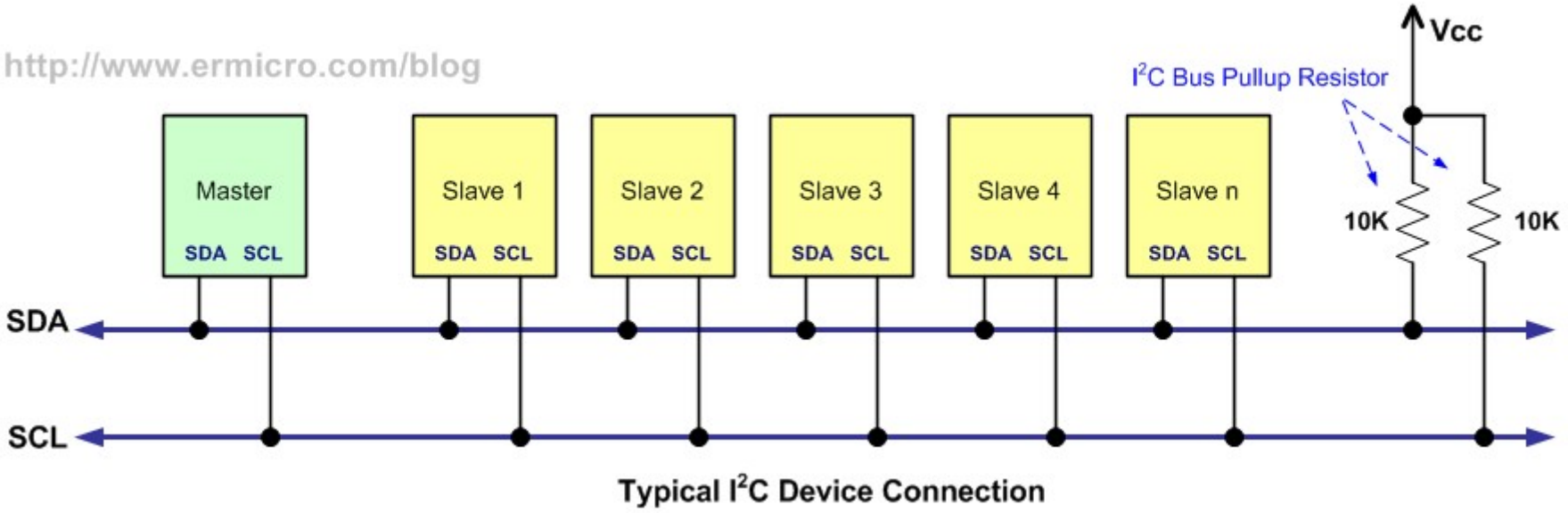
- Are the states of K1, K2 sampled AT THE SAME TIME ?

Inter-integrated Circuit (I^2C)

- I^2C is an interface invented by Philips that used as a communication protocol between microcontroller to its peripherals.
- Common bus speed is 100kbits/s
- Most recent is I^2C can work up to 5Mbits/s
- The main point of I^2C is to use 2-wire for communication, as a result, it sometimes called 2-wire interface, the 2 wires are
  - SCL – Clock
  - SDA – Data
    - Use to synchronize all data transfers over the I^2C bus.
  - Both SCL and SDA lines are "open drain" drivers.
- As you can see, data is sending bit by bit via SDA, it is a serial communication.
- There is another serial interface called SPI which use 4-wire in total.

Inter-integrated Circuit (I^2C)

- Typical Connection



Typical I^2C Device Connection

Inter-integrated Circuit (I^2C)

- The architecture includes
  - Master device
    - Initiates a transaction on the I^2C bus
    - Controls the clock signal
    - Possible to have multiple masters, but most system designs have only one.
  - Slave devices
    - Addressed by the master device
  - Both masters and slaves can receive and transmit data bytes.
- Full specification can be found in Canvas.
  - THE I^2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000

Inter-integrated Circuit (I^2C)

- Bit Transfer

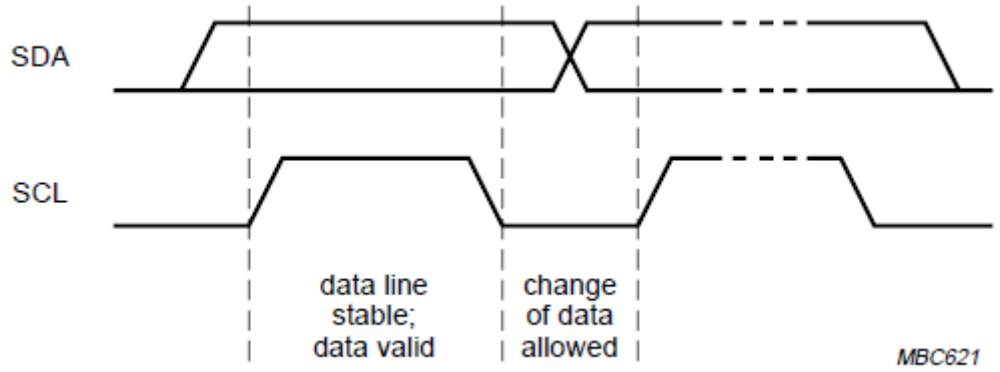


Figure from THE I 2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 document order number 9398 393 40011 by Philips Semiconductors

Inter-integrated Circuit (I^2C)

- Start and Stop Condition

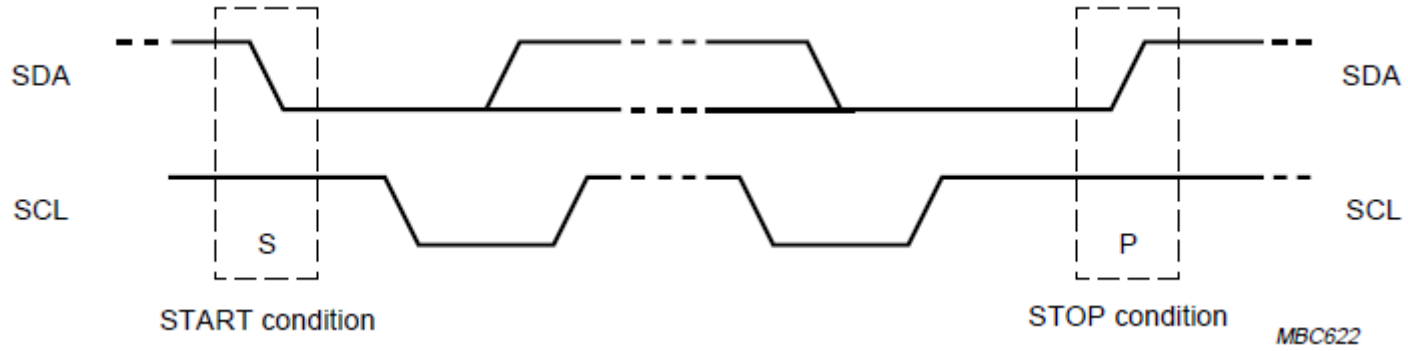


Figure from THE I 2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 document order number 9398 393 40011 by Philips Semiconductors

Inter-integrated Circuit (I^2C)

- Data Transfer

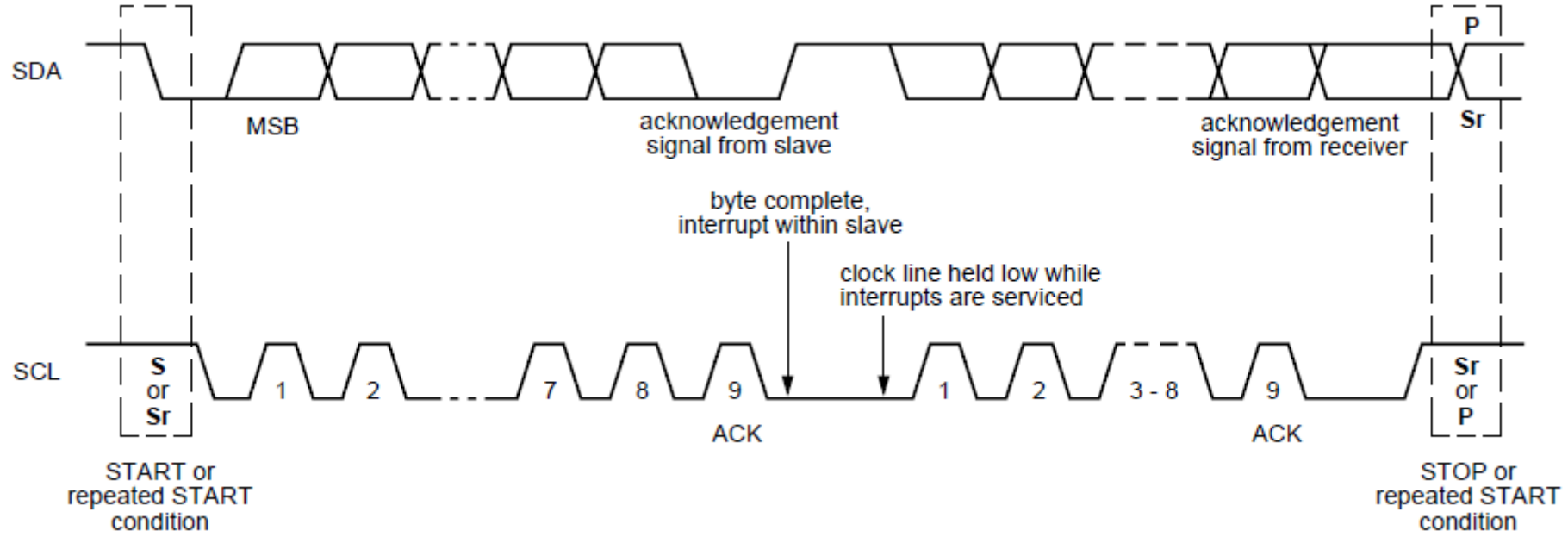


Figure from THE I 2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 document order number 9398 393 40011 by Philips Semiconductors

Inter-integrated Circuit (I^2C)

- A Complete Data Transfer

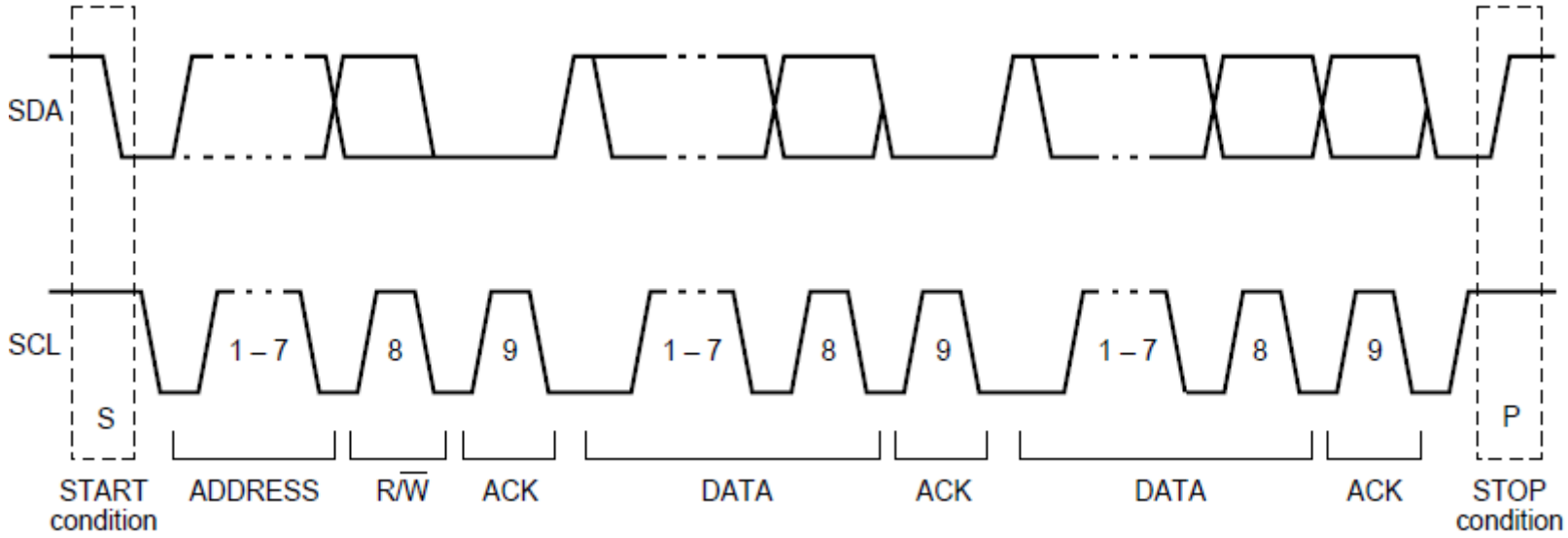


Figure from THE I 2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 document order number 9398 393 40011 by Philips Semiconductors

Inter-integrated Circuit (I^2C)

- A Complete Data Transfer

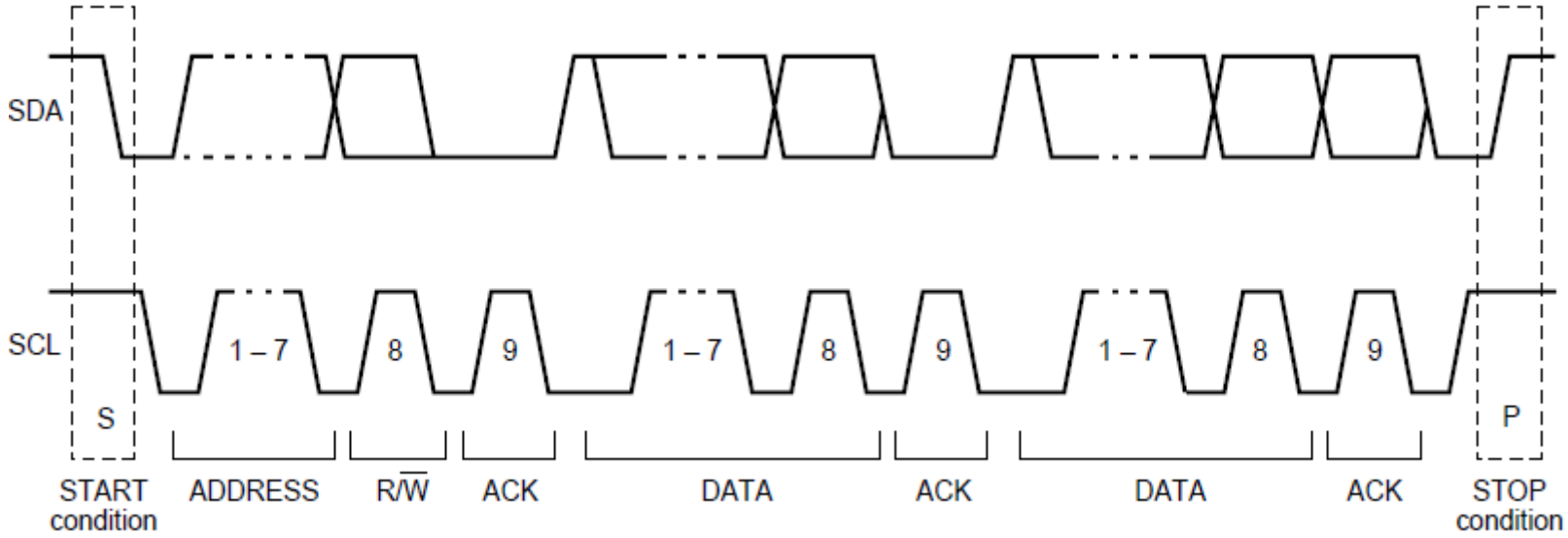
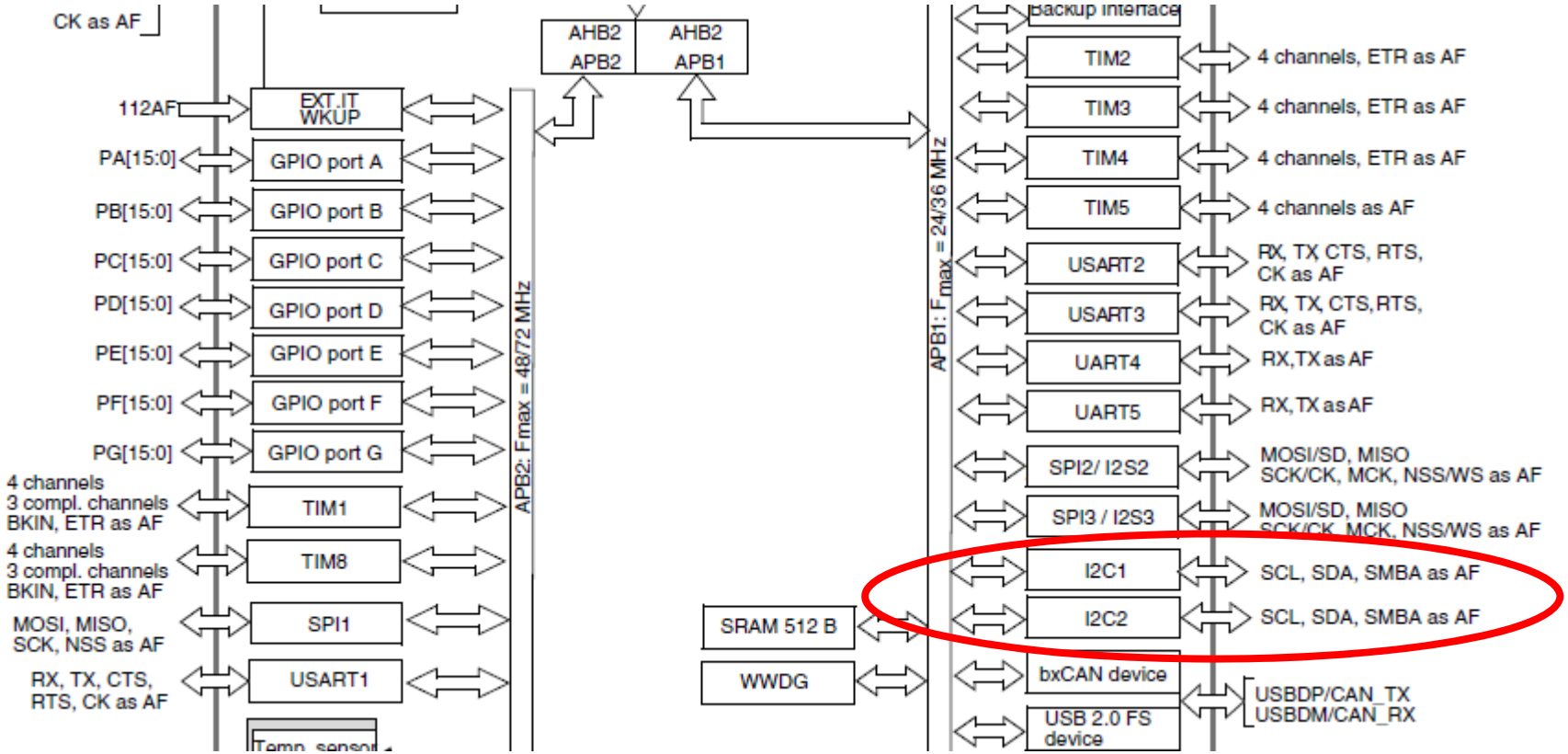


Figure from THE I 2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000 document order number 9398 393 40011 by Philips Semiconductors

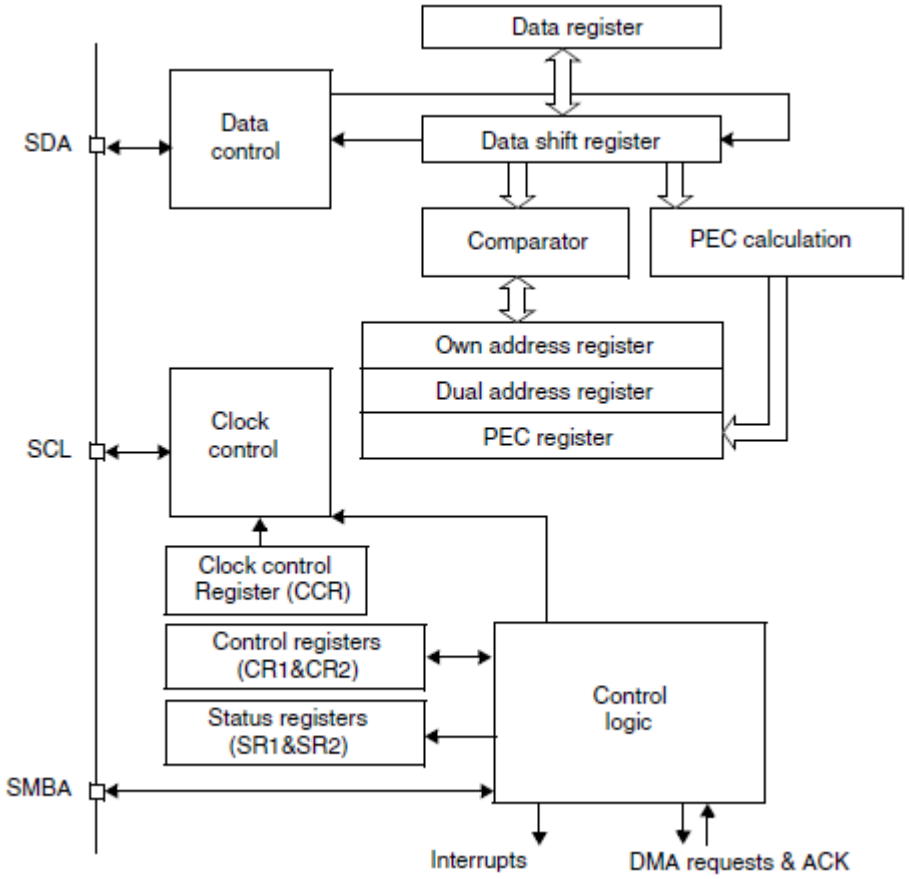
I²C in STM32

- In the STM32F103, there are two I²C Interfaces.



I²C in STM32

- The block diagram of each interface.



I²C in STM32

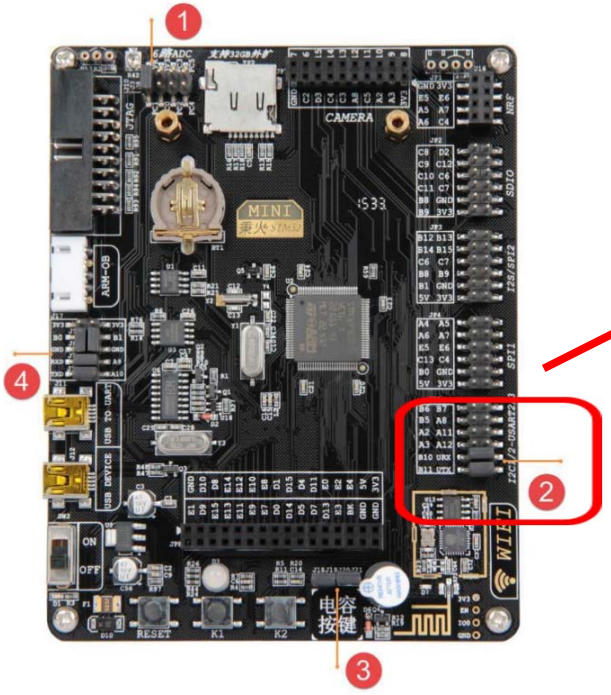
- One of the interface I2C2 is mapped to PB.10 and PB.11

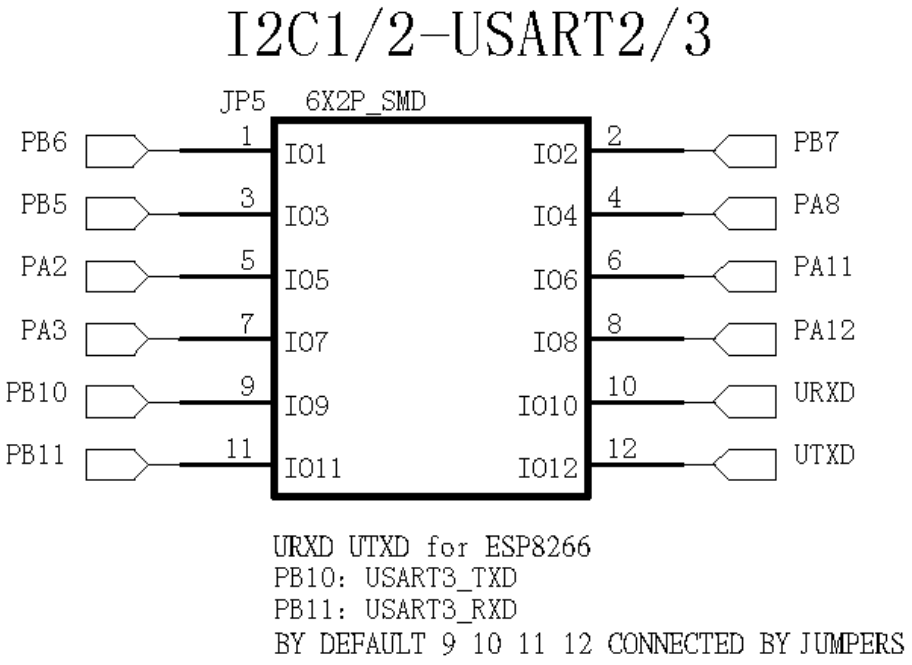
Table 5. High-density STM32F103xx pin definitions (continued)

Pins						Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
BGA144	BGA100	WLCSP64	LQFP64	LQFP100	LQFP144					Default	Remap
M8	H7	-	-	46	68	PE15	I/O	FT	PE15	FSMC_D12	TIM1_BKIN
M9	J7	G3	29	47	69	PB10	I/O	FT	PB10	I2C2_SCL/USART3_TX(8)	TIM2_CH3
M10	K7	F3	30	48	70	PB11	I/O	FT	PB11	I2C2_SDA/USART3_RX(8)	TIM2_CH4
H7	E7	H2	31	49	71	V_{SS_1}	S		V_{SS_1}		

I²C in MINI-V3

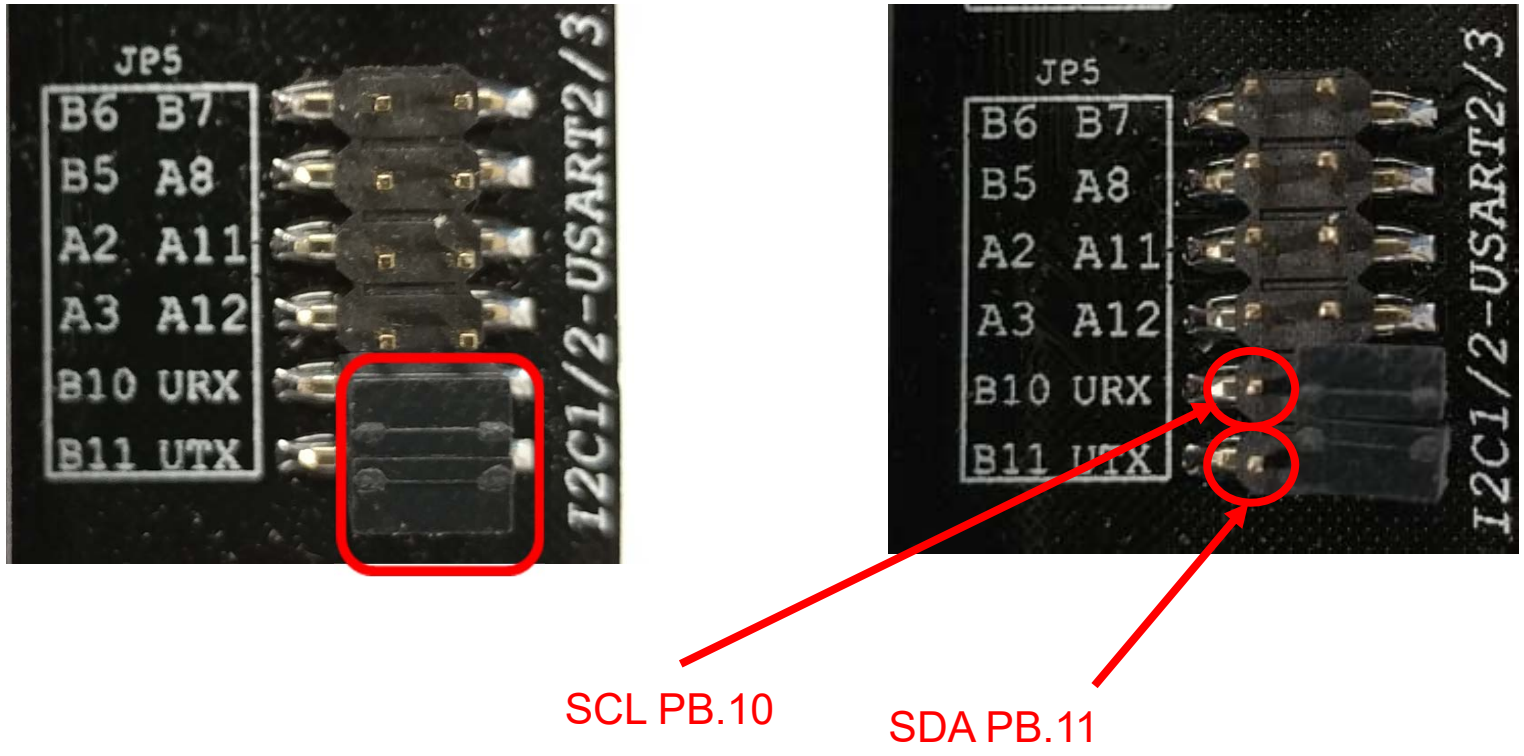
- In MINI-V3 development board, Locate PB.10 and PB.11





I²C in MINI-V3

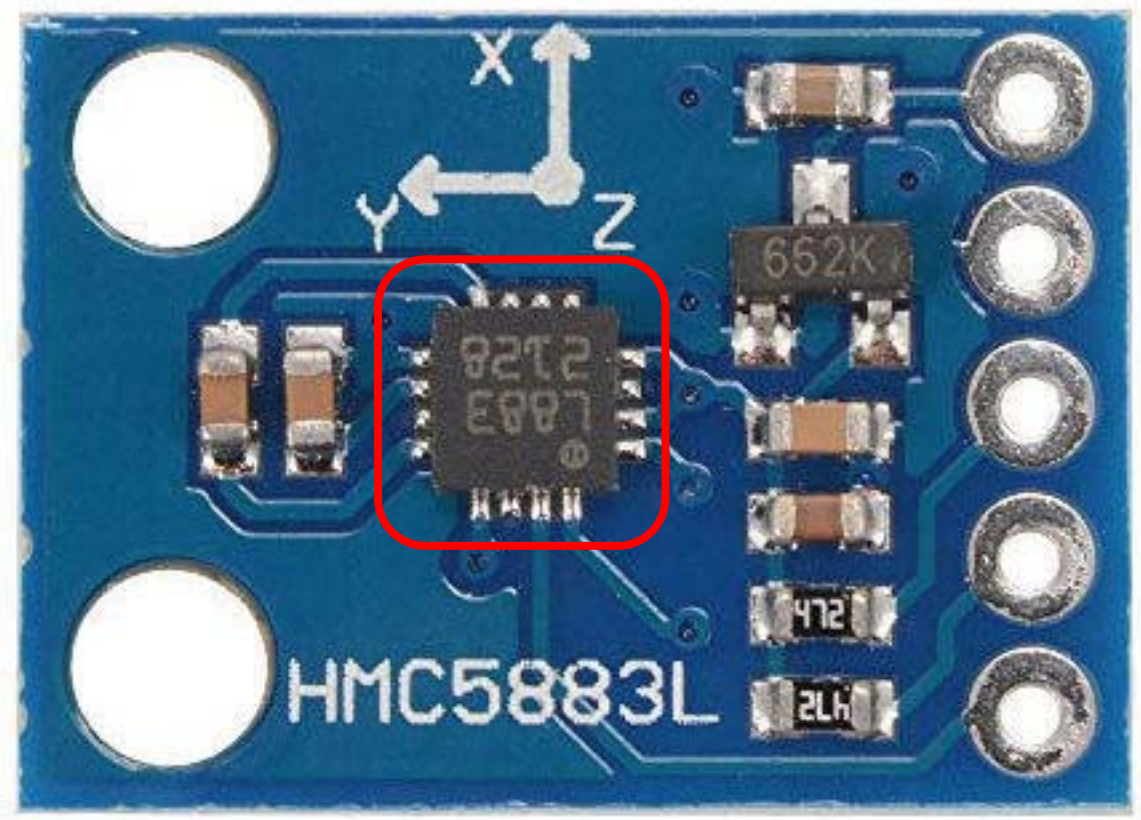
- Remove the Jumpers to access PB.10 and PB.11



HMC5883L IC Module GY273

- HMC5883L is a digital 3-Axis Digital Compass IC

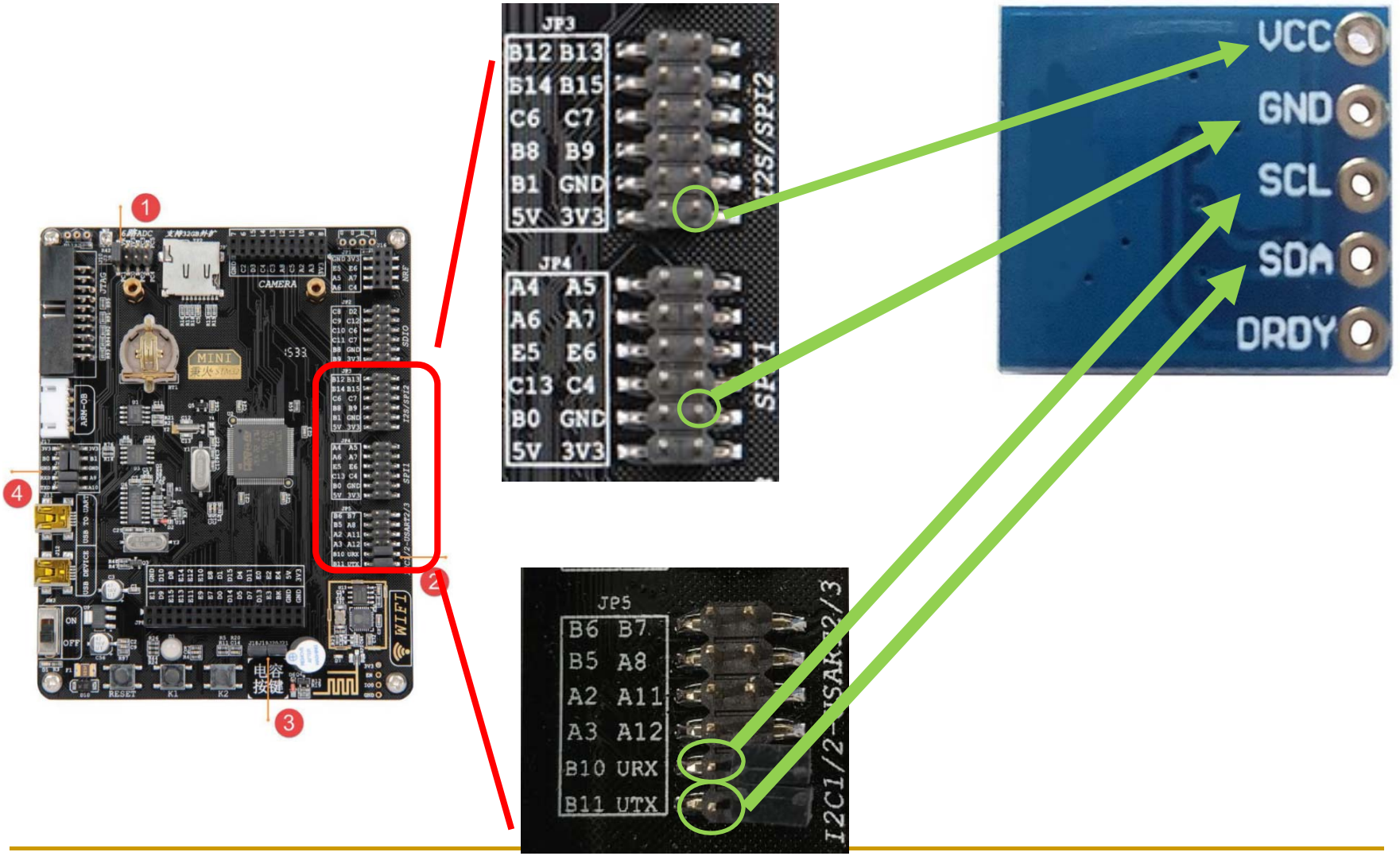
HMC5883L



HMC5883L IC Module GY273

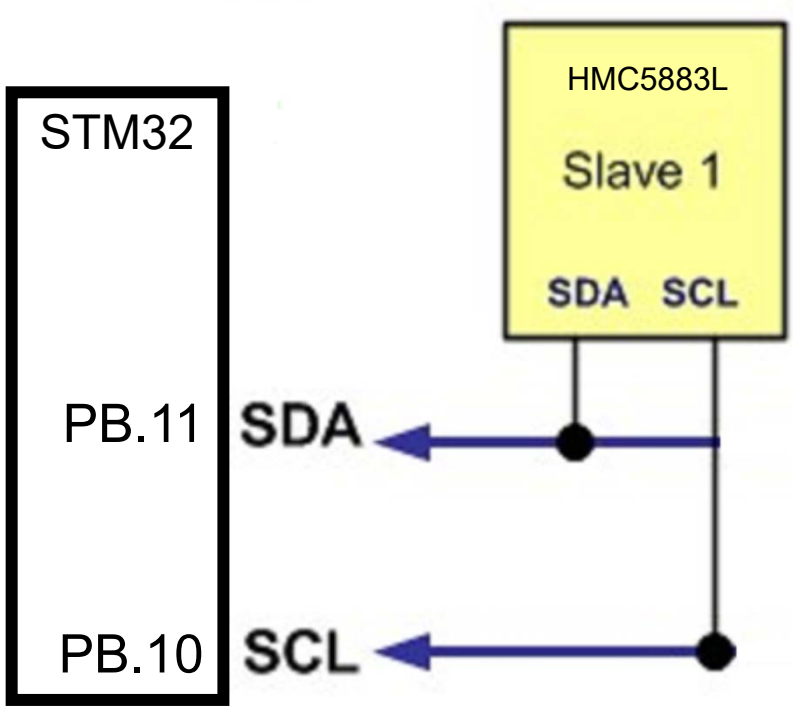
- HMC5883L back side

LAB6 Connection



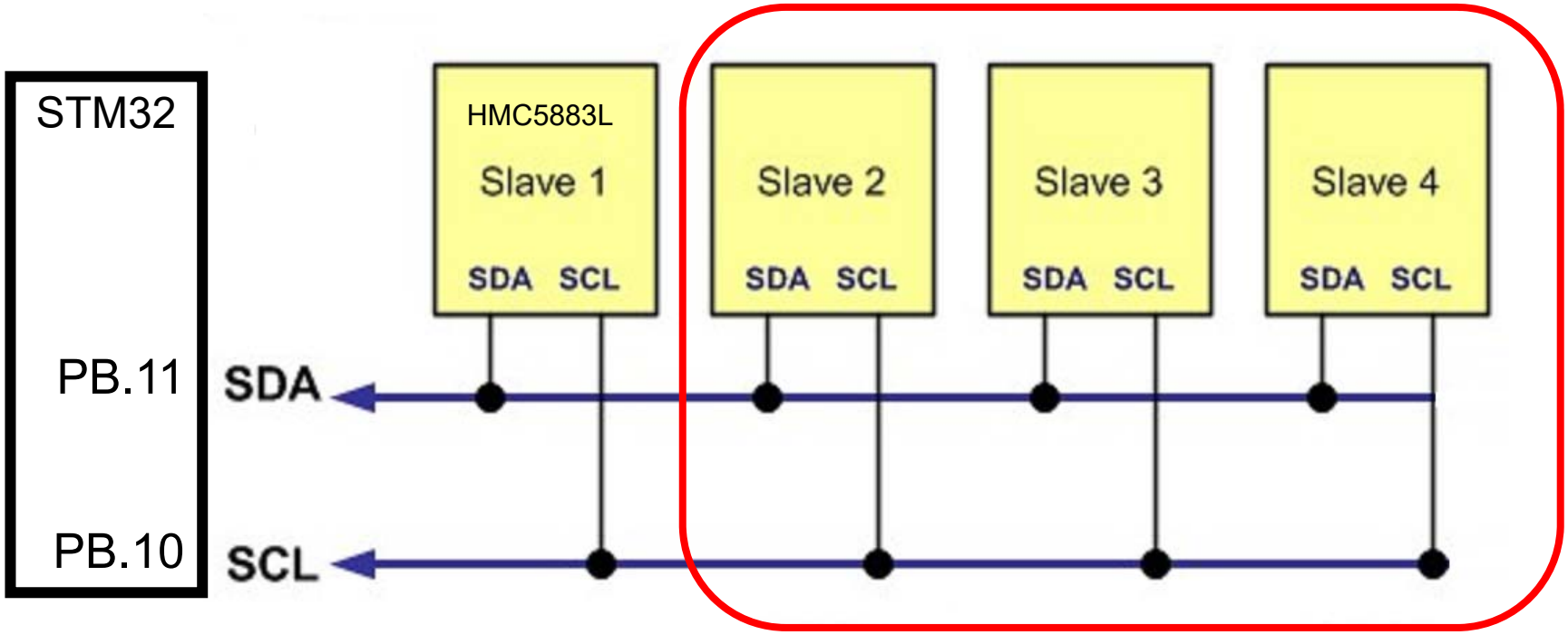
Connection to STM32

- You can consider the connection of module like this



Connecting different I2C devices

- Please note that you can connect more devices like this

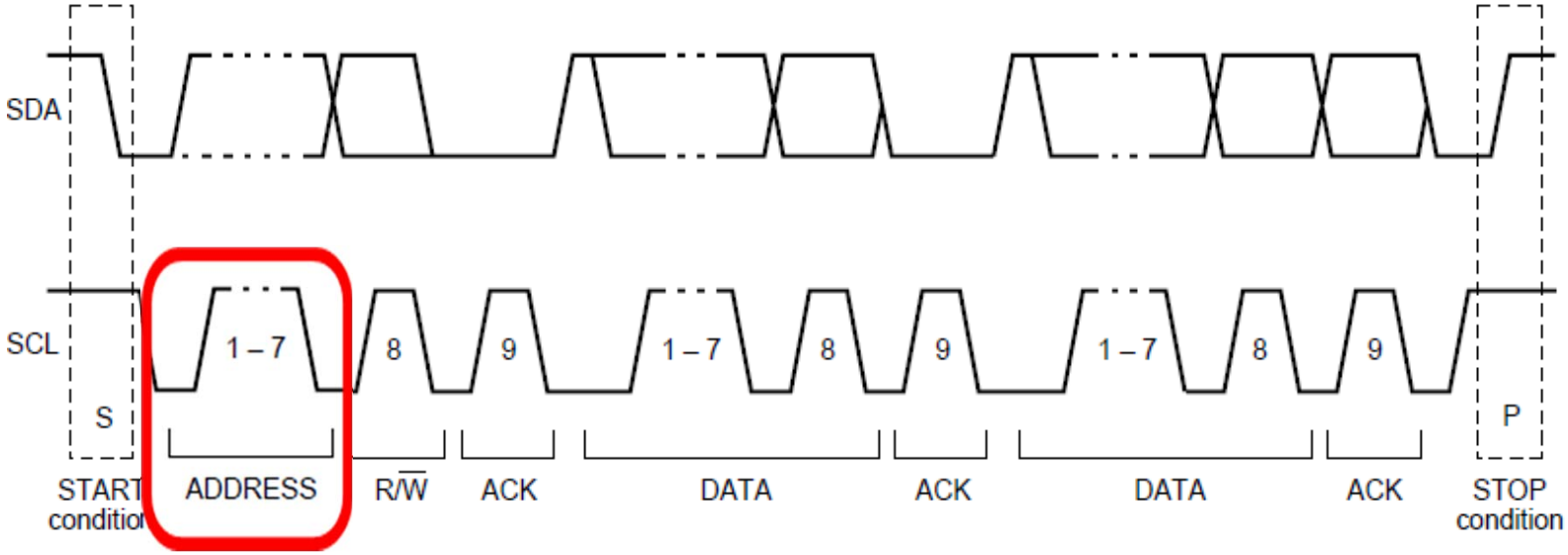


How to communicate ?

- Let's think about if the Master wants to write a byte to one device, what information should the Master gives out ?

Device/Slave Address

- Each device communicating with I²C should have a device/slave address, so that to know who is talking to.



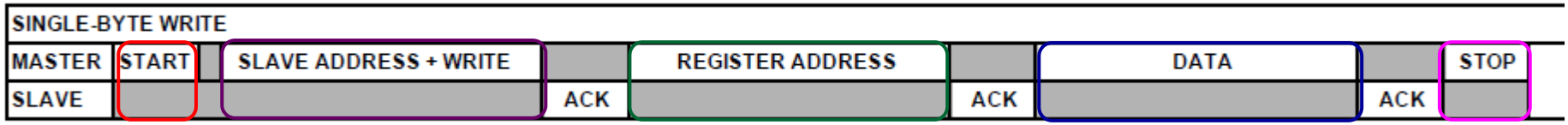
- From specification, this address should be 7-bit

Device/Slave Address

- Below shows the 7-bit Device/slave address of the HMC5883L
  - HMC5883L – 0x1E Shifted Left one bit = 0x3C
- Note: Sometimes datasheets will present in a shifted left function (it actually depends on the code, you should pay attention on this)

Write Timing Diagram

- In order to write a byte to an I^2C device, you need to do the following steps.



Step 1 Step 2 Step 3 Step 4 Step 6

The shaded areas represent when the device is listening

Write function in I^2C

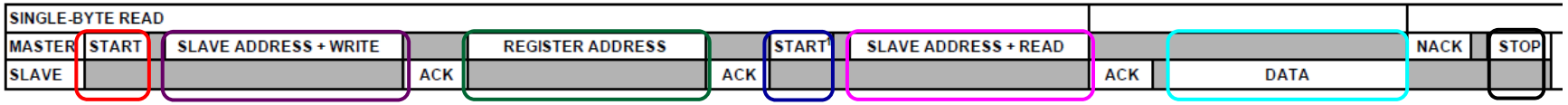
- Steps for a typical I^2C Write function

1. Send a start sequence
2. Send the I2C address of the slave with the R/W bit low (even address)
3. Send the internal register number you want to write to
4. Send the data byte
5. [Optionally, send any further data bytes]
6. Send the stop sequence.

Note: Actually, you need to check the status after each step.

Read Timing Diagram

- In order to read a byte form an I^2C device, you need to do the following steps.



Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Step 7

The shaded areas represent when the device is listening

Read function in I^2C

- A typical read function by I^2C as follows

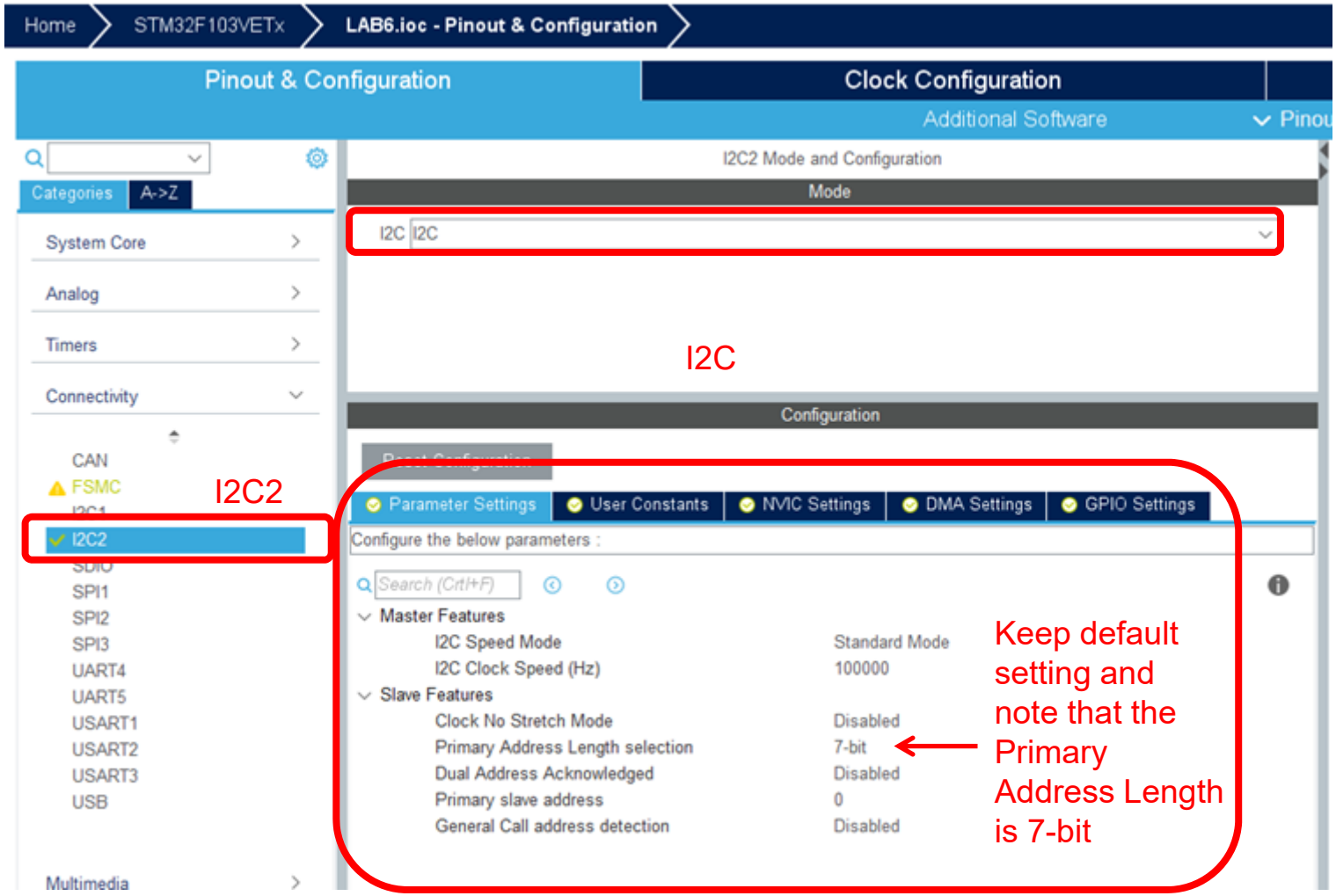
1. Send a start sequence
2. Send the read address with the R/W bit low (even address)
3. Send the lower read address
4. Send a start sequence again (repeated start)
5. Send the read address with the R/W bit high (odd address)
6. Read data byte
7. Send the stop sequence.

Note: Actually, you need to check the status after each step.

Configuration of I2C in CubeIDE

- In this LAB, we need to use the LCD to display the value.
- Please refer to the Tutorial for CubeIDE and Tutorial for LAB3 to create a project that allows you to use the LCD Display.
- Or you may start your LAB6 by using the LAB3 as a starting point.

Configuration of I2C2 in CubeIDE



You can then Generate the Project Template

I²C Initialization

- After code generation, in main.c you will notice that there is a I2C interface

```
I2C_HandleTypeDef hi2c2;
```

- The STM32 will perform the necessary timing for shifting out and shifting in the bits by using the I2C HAL library functions.

I²C HAL Functions available to you

- In I²C HAL Library, there is a Mem_Read function

```
HAL_StatusTypeDef HAL_I2C_Mem_Read(I2C_HandleTypeDef *hi2c, uint16_t
DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData,
uint16_t Size, uint32_t Timeout)
/**
  * @brief  Read an amount of data in blocking mode from a specific memory address
  * @param  hi2c Pointer to a I2C_HandleTypeDef structure that contains
  *                the configuration information for the specified I2C.
  * @param  DevAddress Target device address: The device 7 bits address value
  *         in datasheet must be shifted to the left before calling the interface
  * @param  MemAddress Internal memory address
  * @param  MemAddSize Size of internal memory address
  * @param  pData Pointer to data buffer
  * @param  Size Amount of data to be sent
  * @param  Timeout Timeout duration
  * @retval HAL status
  */
```

I²C HAL Functions available to you

- In I²C HAL Library, there is a Mem_Write function

```
HAL_StatusTypeDef HAL_I2C_Mem_Write(I2C_HandleTypeDef *hi2c, uint16_t
DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t *pData,
uint16_t Size, uint32_t Timeout)
/**
  * @brief  Write an amount of data in blocking mode to a specific memory address
  * @param  hi2c Pointer to a I2C_HandleTypeDef structure that contains
  *                the configuration information for the specified I2C.
  * @param  DevAddress Target device address: The device 7 bits address value
  *         in datasheet must be shifted to the left before calling the interface
  * @param  MemAddress Internal memory address
  * @param  MemAddSize Size of internal memory address
  * @param  pData Pointer to data buffer
  * @param  Size Amount of data to be sent
  * @param  Timeout Timeout duration
  * @retval HAL status
  */
```

I²C HAL Functions available to you

- Example

  If you want to read one byte stored in address 0x6C at HMC5883L (DevAddress is 0x1E), using I2C2

```
HAL_I2C_Mem_Read(&hi2c2,0x1E<<1,0x6C,1,&data,1,100);
```

  where data is a uint8_t variable.

  So, if you want to read one byte stored in address 0x00 at HMC5883L, what should you write ?

I²C HAL Functions available to you

- Example

  If you want to write one byte (say, 0xFF) to address 0x30 on HMC5883L (DevAddress is 0x1E), using I2C2

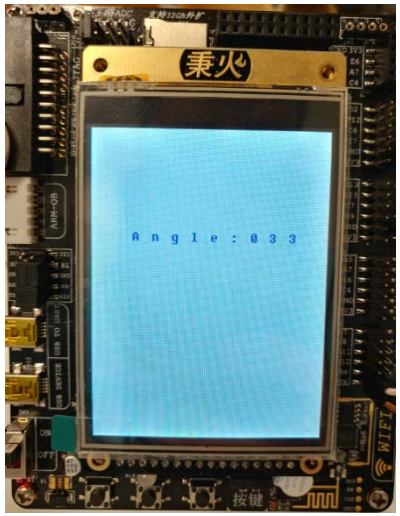
```
HAL_I2C_Mem_Write(&hi2c2,0x1E<<1,0x30,1,&data,1,100);
```

```
where       uint8_t data = 0xff
```

  So, if you want to write one byte (say 0xAB) to address 0x20 on HMC5883L, what should you write ?

LAB6 – Task 1

- In this LAB, you are required to get the information from the Digital Compass and display the information on the LCD Screen. 0 – 359 degree. Below is just an example, you can display even more information



- As it is the last LAB, you need to think how to fill the code in the while(1) loop to complete the task.

HMC5883L – On-chip Registers

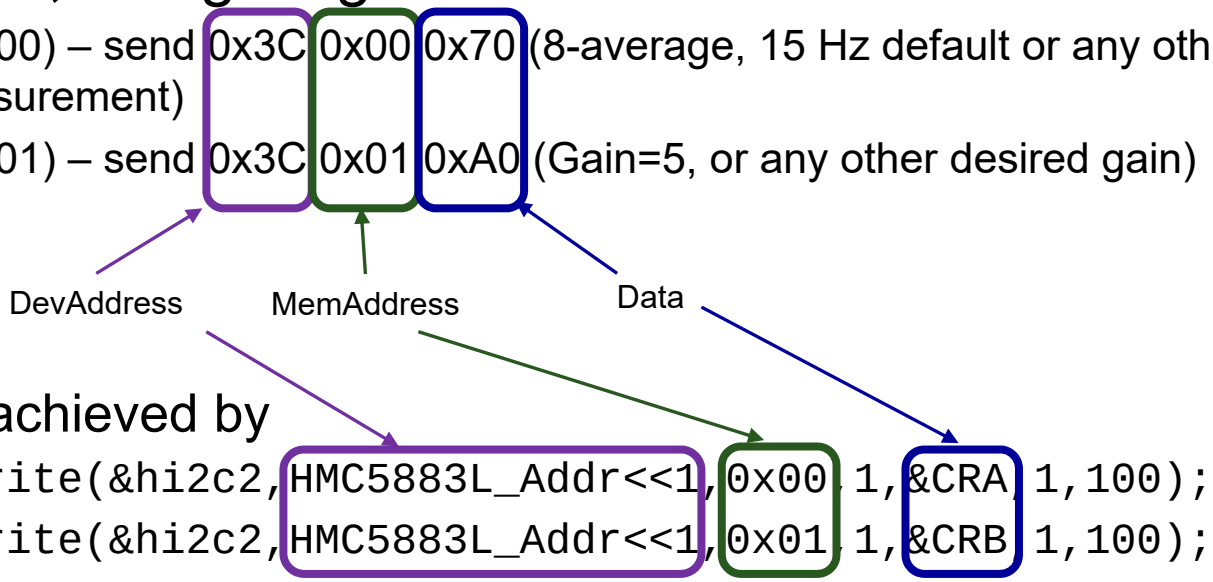
- HMC5883L is controlled and configured via a number of on-chip registers. The table below lists the registers and their access. All address locations are 8 bits.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

HMC5883L – Initialization

- The initialization for the HMC is written on page 18 of the datasheet, using single measurement mode.

1. Write CRA (00) – send 0x3C 0x00 0x70 (8-average, 15 Hz default or any other rate, normal measurement)
2. Write CRB (01) – send 0x3C 0x01 0xA0 (Gain=5, or any other desired gain)



- It can be achieved by

```
HAL_I2C_Mem_Write(&hi2c2,HMC5883L_Addr<<1,0x00,1,&CRA,1,100);
HAL_I2C_Mem_Write(&hi2c2,HMC5883L_Addr<<1,0x01,1,&CRB,1,100);
```

- where

```
HMC5883L_Addr = 0x1E; uint8_t CRA = 0x70; uint8_t CRB = 0xA0;
```

HMC5883L – Getting Data

- For each measurement query:

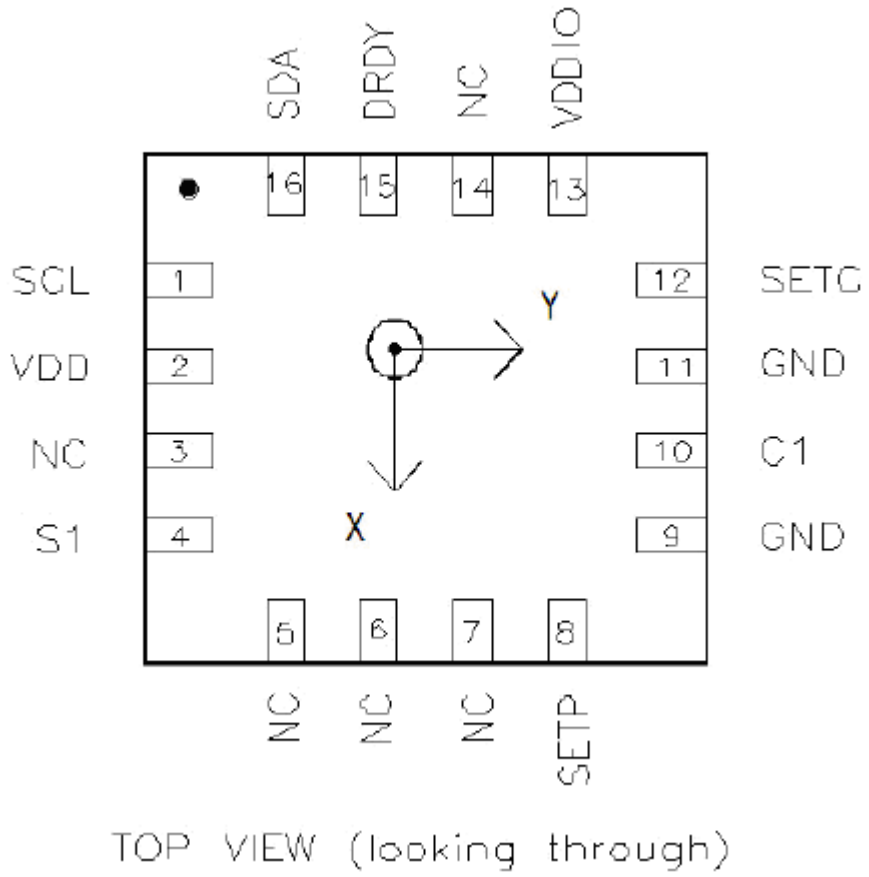
1. Write Mode (02) – send 0x3C 0x02 0x01
   - Write the Mode Register (02) with value 01.
2. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
3. Read X, Z, Y values from registers
4. Convert three 16-bit 2's compliment hex values to decimal values and assign to X, Z, Y respectively.

HMC5883L – X, Y, Z

- Refer to Page 15, 16 of the Datasheet, X, Y, Z are three values stored in 6 registers namely
  - X – (MSB) DXRA (LSB) DXRB
  - Y – (MSB) DYRA (LSB) DYRB
  - Z – (MSB) DZRA (LSB) DZRB
- The value stored in the corresponding registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF.
- Actually, you need X and Y values only.

HMC5883L – X, Y

- Arrow indicates direction of magnetic field that generates a positive output reading in Normal Measurement configuration.

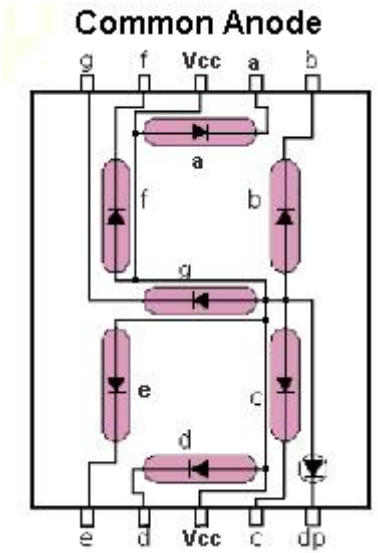


Presenting the output

- Things to consider
- You got X and Y, how to get an angle out of them ?
  - Which trigonometry function you need ? → Refer to math.h
  - Remember that values of X and Y are signed. How does it relates to the angle ?
  - You need to pay attention to the data type for the variables, it might help you in simplifying your calculation.
- Once you got the angle, how can you use the LCD functions you developed before to display your result to LCD ?

LAB6 – Task 2

- In order to let you familiar with the board.
- I want you to display the last digit using a 7-segment LED externally. (e.g. if LCD is displaying 236, the 7-segment LED should display 6)
- You will be given a Common Anode 7-segment LED



LAB6 – Task 2 Connection

- The connection of the 7-segment LED will depends on your student ID.
- Example, if your student ID is 21234567, you need to check accordingly to the Pin Set below.

Pin Set	Actual Pin Number on STM32	Default Function of the pin on 100pin STM32F103VET6
A	67	PA8
B	56	PD9
C	45	PE14
D	34	PC5
E	23	PA0
F	12	OSC_IN
G	21	VREF+

LAB6 – Task 2

- Differs from LAB2, as this LAB we used the LCD and I^2C, some pins that originally available for you might not be available now.
- You are strongly suggested to check the datasheets and the schematics available on Canvas.
- You are **REQUIRED** to check all the pins as shown on next page.
- Please note that the student ID 21234567 as an example, you are **REQUIRED** to use your student ID for Task 2.

LAB6 – Task 2 Connection

Pin Set	Actual Pin Number on STM32	Default Function of the pin on 100pin STM32F103VET6	I/O function ?	Alternate Functions	Function on the MINI V3 Development Board	Can use for 7-segment LED ?
A	67	PA8	Yes			
B	56	PD9	Yes			
C	45	PE14	Yes			
D	34	PC5	Yes			
E	23	PA0	Yes			
F	12	OSC_IN	No			
G	21	VREF+	No			

LAB6 – Task 2 Connection

Pin Set	Actual Pin Number on STM32	Default Function of the pin on 100pin STM32F103V ET6	I/O Function	Alternate Functions	Function on the MINI V3 Development Board	Can use for 7-segment LED ?
A	67	PA8	Yes	USART1_CK/ TIM1_CH1(8)/MCO	Speaker	Yes, if we remove the jumper for the Speaker No, if we do not remove the jumper
B	56	PD9	Yes	FSMC_D14	LCD Data bus 14	No, as this LAB used LCD
C	45	PE14	Yes	FSMC_D11	LCD Data bus 11	No, as this LAB used LCD
D	34	PC5	Yes	ADC12_IN15	ADC12_IN15 Camera FIFO RCLK	Yes, as this LAB not used ADC and Camera
E	23	PA0	Yes	WKUP/USART2_CTS(8) ADC123_IN0 TIM2_CH1_ETR TIM5_CH1/TIM8_ETR	K1 on the Development Board	No, as there is no external connector for PA0 on the Development Board
F	12	OSC_IN	No			
G	21	VREF+	No			

LAB6 – Task 2 Connection

- For example, if the student ID is 21234567, refer to the last page, I can only use pin set D.

Pin Set	Actual Pin Number on STM32	Default Function of the pin on 100pin STM32F103V ET6	I/O function ?	Alternate Functions	Function on the MINI V3 Development Board	Can use for 7-segment LED ?
D	34	PC5	Yes	ADC12_IN15	ADC12_IN15 Camera FIFO RCLK	Yes, as this LAB not used ADC and Camera

- You are **REQUIRED** to use *ALL* the pin sets available according to your student ID to connect to the 7-segment LED.
- Other then the pin sets required, you are then free to choose the unused I/O pins of STM32 to connect to the 7-segment.
- You are **REQUIRED** to show to the TA, together with your student ID
  1. The filled table on your LAB Sheet
  2. Your corresponding program and hardware

LAB6 – Task 2 Procedure

- Basically you need to control the 7 pins on and off.
- However, you need to build your own decoding table. (i.e. how to display 1, 2, 3, 4 … 0)
- Please connect the V_{cc} of the 7-segment to 3.3V with a resistor.
- Enjoy ☺

After finishing LAB6 you are expected to…

1. Use CubeIDE to initialize different I2C interface.
2. Understand the I2C protocol and how Slave address is being incorporated in the I2C data.
3. Understand how to get important information from the I2C devices based on the information/operating instructions from datasheet.
4. Translate the raw data from I2C devices to useful data for the system. (e.g. For LAB6, how to convert the X Y to angle.)
5. Integrate the knowledge of LAB2 (I/O), LAB3 (LCD) add to LAB6 to connect to 7-segment.

END