I2C AM2320 & LCD Control

Embedded System 2561, KU CSC

Adapted by Sorayut Glomglome

I2C

- I2C เป็นโปรโตคอลสื่อสารอนุกรม ประกอบด้วยขาสัญญาณ
  - SDA: สัญญาณข้อมูล
  - SCL: สัญญาณนาฬิกา
  - https://www.youtube.com/watch?v=6IAkYpmA1DQ

STM32F411 I2C

- I^2C Master features:
  - Clock generation
  - Start and Stop generation

- I^2C Slave features:
  - Programmable I^2C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection

- Supports different communication speeds:
  - Standard Speed (up to 100 kHz)
  - Fast Speed (up to 400 kHz)
  - The I2C bus frequency can be increased up to 1 MHz. For more details about the complete solution, please contact your local ST sales representative

Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

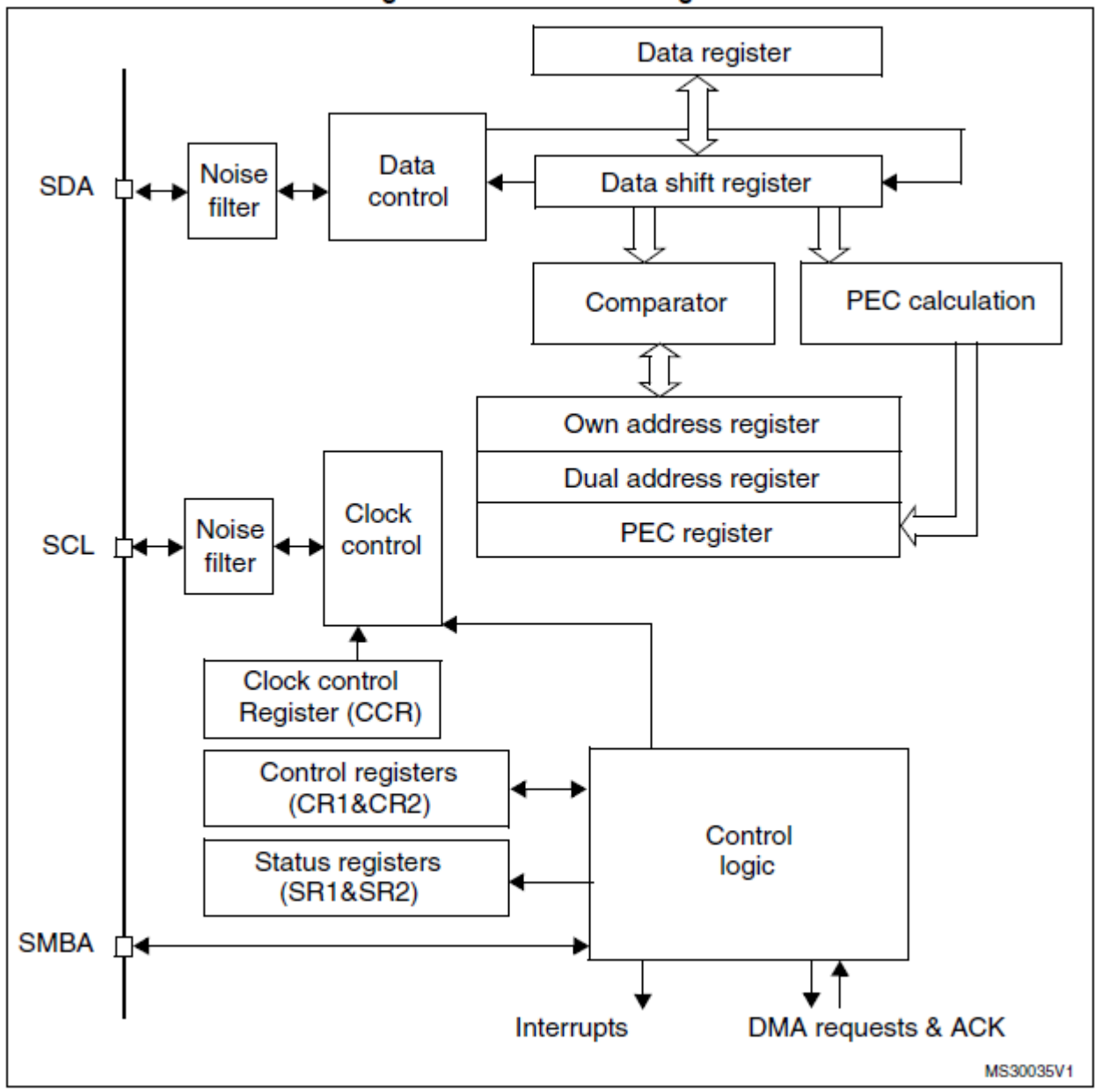
By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.

STM32F411 I2C

- I^2C Master features:
  - Clock generation
  - Start and Stop generation

- I^2C Slave features:
  - Programmable I^2C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection

Figure 161. I^2C block diagram



1. SMBA is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.

Table 9. Alternate function mapping

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	SPI4_MOSI /I2S4_SD	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	I2S2_CKIN	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	I2S2_MCK	-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT
	PA4	-	-	-	-	-	SPI1_NSS/I 2S1_WS	SPI3_NSS/I2 S3_WS	USART2_ CK	-	-	-	-	-	-	-	EVENT OUT
	PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK/I 2S1_CK	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_MISO	I2S2_MCK	-	-	-	-	-	SDIO_ CMD	-	-	EVENT OUT
	PA7	-	TIM1_CH1N	TIM3_CH2	-	-	SPI1_MOSI /I2S1_SD	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA8	MCO_1	TIM1_CH1	-	-	I2C3_ SCL	-	-	USART1_ CK	-	-	USB_FS_ SOF	-	SDIO_ D1	-	-	EVENT OUT
	PA9	-	TIM1_CH2	-	-	I2C3_ SMBA	-	-	USART1_ TX	-	-	USB_FS_ VBUS	-	SDIO_ D2	-	-	EVENT OUT
	PA10	-	TIM1_CH3	-	-	-	-	SPI5_MOSI/I 2S5_SD	USART1_ RX	-	-	USB_FS_ ID	-	-	-	-	EVENT OUT
	PA11	-	TIM1_CH4	-	-	-	-	SPI4_MISO	USART1_ CTS	USART6_ TX	-	USB_FS_ DM	-	-	-	-	EVENT OUT
	PA12	-	TIM1_ETR	-	-	-	-	SPI5_MISO	USART1_ RTS	USART6_ RX	-	USB_FS_ DP	-	-	-	-	EVENT OUT
	PA13	JTMS-SWDIO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA14	JTCK-SWCLK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA15	JTDI	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_NSS/I 2S1_WS	SPI3_NSS/I2 S3_WS	USART1_ TX	-	-	-	-	-	-	-	EVENT OUT

Table 9. Alternate function mapping (continued)

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	-	-	-	SPI5_SCK /I2S5_CK		-	-	-	-	-	-	-	EVENT OUT
	PB1	-	TIM1_CH3N	TIM3_CH4	-	-	-	SPI5_NSS /I2S5_WS		-	-	-	-	-	-	-	EVENT OUT
	PB2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB3	JTDO-SWO	TIM2_CH2	-	-	-	SPI1_SCK/I2S1_CK	SPI3_SCK /I2S3_CK	USART1_RX	-	I2C2_SDA	-	-	-	-	-	EVENT OUT
	PB4	JTRST		TIM3_CH1	-	-	SPI1_MISO	SPI3_MISO	I2S3ext_SD	-	I2C3_SDA			SDIO_D0	-	-	EVENT OUT
	PB5	-	-	TIM3_CH2	-	I2C1_SMBA	SPI1_MOSI /I2S1_SD	SPI3_MOSI/ I2S3_SD		-	-	-	-	SDIO_D3	-	-	EVENT OUT
	PB6	-	-	TIM4_CH1	-	I2C1_SCL	-	-	USART1_TX	-	-	-	-		-	-	EVENT OUT
	PB7	-	-	TIM4_CH2	-	I2C1_SDA	-	-	USART1_RX	-	-	-	-	SDIO_D0	-	-	EVENT OUT
	PB8	-	-	TIM4_CH3	TIM10_CH1	I2C1_SCL	-	SPI5_MOSI/ I2S5_SD	-	-	I2C3_SDA	-	-	SDIO_D4	-	-	EVENT OUT
	PB9	-	-	TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2_NSS/I2S2_WS	-	-	-	I2C2_SDA	-	-	SDIO_D5	-	-	EVENT OUT
	PB10	-	TIM2_CH3	-	-	I2C2_SCL	SPI2_SCK/I2S2_CK	I2S3_MCK	-	-	-	-	-	SDIO_D7	-	-	EVENT OUT
	PB11	-	TIM2_CH4	-	-	I2C2_SDA	I2S2_CKIN	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB12	-	TIM1_BKIN	-	-	I2C2_SMBA	SPI2_NSS/I2S2_WS	SPI4_NSS /I2S4_WS	SPI3_SCK /I2S3_CK	-	-	-	-	-	-	-	EVENT OUT
	PB13	-	TIM1_CH1N	-	-	-	SPI2_SCK/I2S2_CK	SPI4_SCK/ I2S4_CK	-	-	-	-	-	-	-	-	EVENT OUT
	PB14	-	TIM1_CH2N	-	-	-	SPI2_MISO	I2S2ext_SD	-	-	-	-	-	SDIO_D6	-	-	EVENT OUT
	PB15	RTC_50Hz	TIM1_CH3N	-	-	-	SPI2_MOSI /I2S2_SD	-	-	-	-	-	-	SDIO_CK	-	-	EVENT OUT

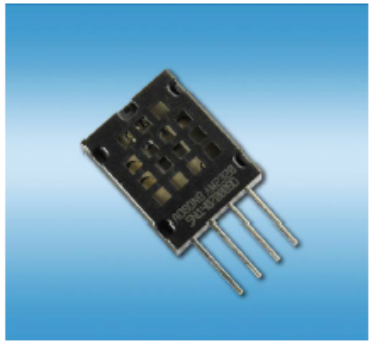
Table 9. Alternate function mapping (continued)

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port C	PC0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PC1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PC2	-	-	-	-	-	SPI2_MISO	I2S2ext_SD	-	-	-	-	-	-	-	-	EVENT OUT
	PC3	-	-	-	-	-	SPI2_MOSI /I2S2_SD	-	-	-	-	-	-	-	-	-	EVENT OUT
	PC4	-	-	-	-	-		-	-	-	-	-	-	-	-	-	EVENT OUT
	PC5	-	-	-	-	-		-	-	-	-	-	-	-	-	-	EVENT OUT
	PC6	-	-	TIM3_CH1	-	-	I2S2_MCK	-	-	USART6_ TX	-	-	-	SDIO_ D6	-	-	EVENT OUT
	PC7	-	-	TIM3_CH2	-	-	SPI2_SCK/I 2S2_CK	I2S3_MCK	-	USART6_ RX	-	-	-	SDIO_ D7	-	-	EVENT OUT
	PC8	-	-	TIM3_CH3	-	-	-	-	-	USART6_ CK	-	-	-	SDIO_ D0	-	-	EVENT OUT
	PC9	MCO_2	-	TIM3_CH4	-	I2C3_SDA	I2S2_CKIN	-	-		-	-	-	SDIO_ D1	-	-	EVENT OUT
	PC10	-	-	-	-	-	-	SPI3_SCK/I2 S3_CK	-	-	-	-	-	SDIO_ D2	-	-	EVENT OUT
	PC11	-	-	-	-	-	I2S3ext_SD	SPI3_MISO	-	-	-	-	-	SDIO_ D3	-	-	EVENT OUT
	PC12	-	-	-	-	-	-	SPI3_MOSI/I 2S3_SD	-	-	-	-	-	SDIO_ CK	-	-	EVENT OUT
	PC13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	PC14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	PC15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

AOSONG

Digital Temperature and Humidity Sensor

AM2320 Product Manual



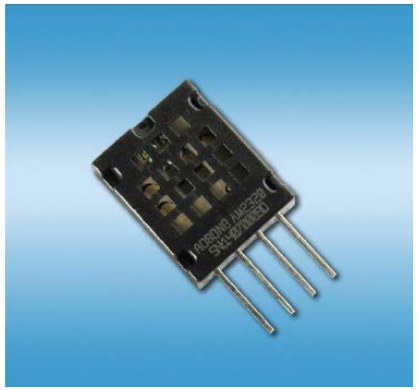
Product Features:

- Ultra-small size
- Super cost-effective
- Ultra-low voltage operation
- Excellent long-term stability
- Standard I2C and single-bus output

For more information, please visit: www. aosong .com

AM2320

- **Digital** Temperature & Humidity Sensor
- Standard I2C and single-bus output (1Wire)
- CRC checksum



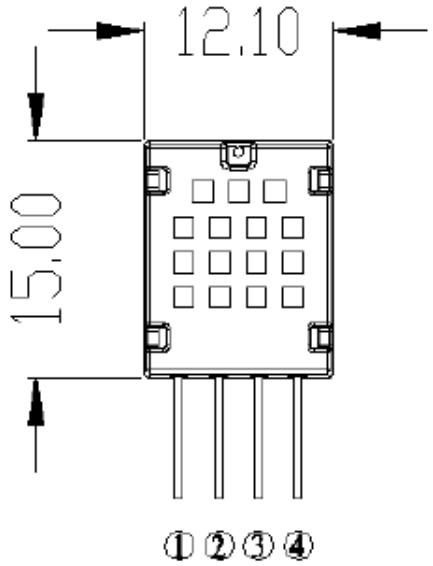


Table 4: AM2320 pin assignment

Pin	Name	Description
1	VDD	Power supply(3.1-5.5V)
2	SDA	Serial data, bidirectional port
3	GND	Ground
4	SCL	Serial clock input port (single bus ground)

Sensor performance : Relative Humidity

Table 1: AM2320 relative humidity performance table

parameter	condition	min	typ	max	unit
resolution			0.1		%RH
Range		0		99.9	%RH
Accuracy	25℃		±3		%RH
Repeatability			±0. 1		%RH
Interchangeability		Completely interchangeable			
Response time	1/e(63%)		<5		S
Sluggish			±0.3		%RH
Drift	Typical values		<0.5		%RH /yr

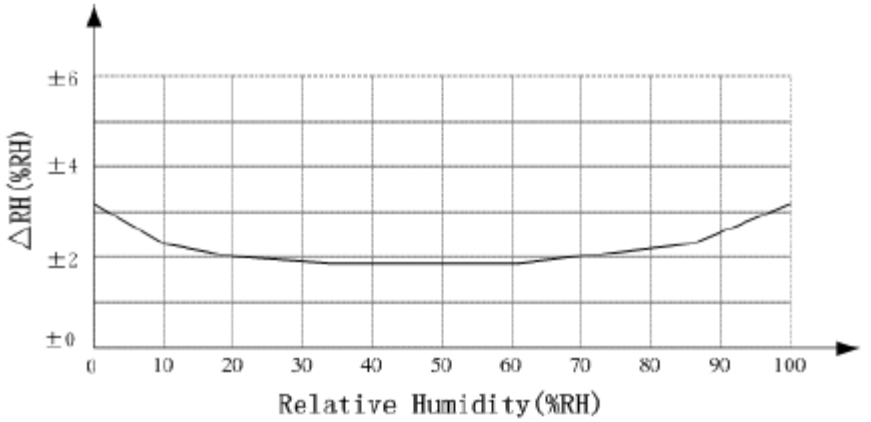


Figure 2: 25 ℃ relative humidity of maximum error AM2320

Sensor performance : Temperature

Table 2: AM2320 relative temperature performance table

parameter	condition	min	typ	max	unit
resolution			0.1		℃
			16		bit
Accuracy			±0.5		℃
Range		-40		80	℃
Repeatability			±0.2		℃
Interchangeability					
Response time	1/e(63%)		<5		S
Drift			±0.1		℃/yr

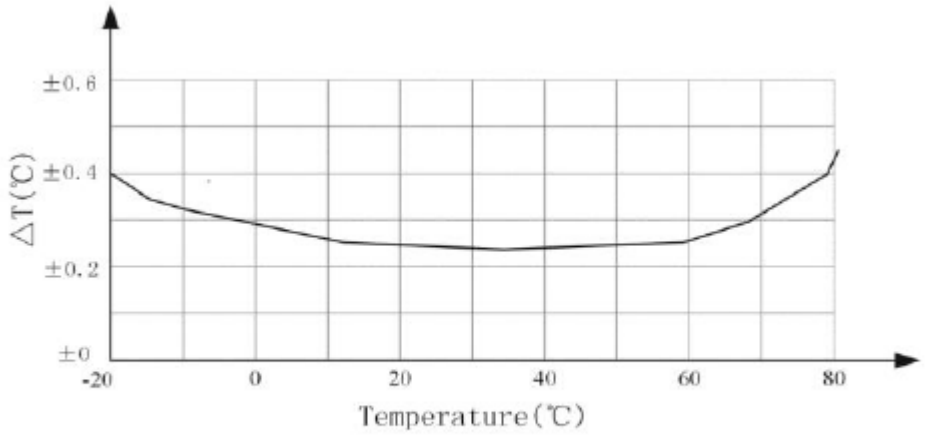


Figure 3: The maximum error of the temperature sensor

AM2320 Wiring

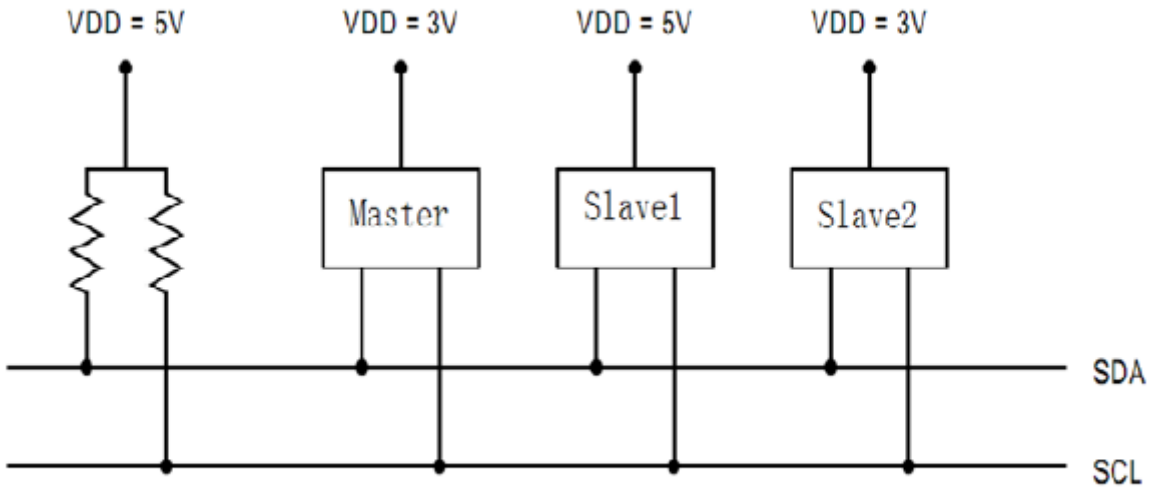
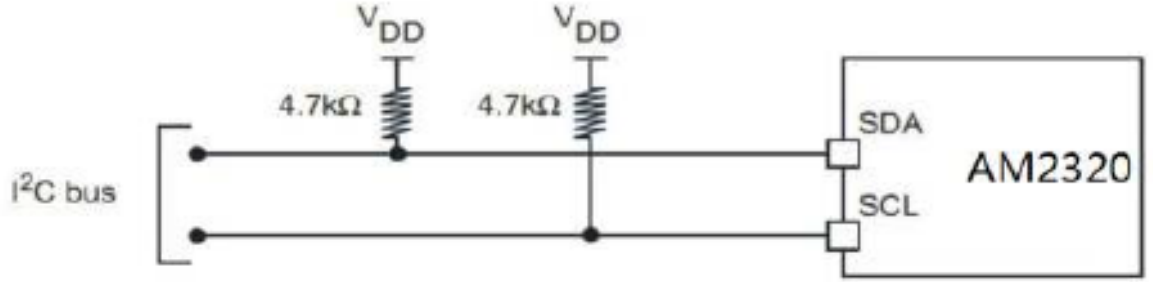


Figure 5: I ^{2} C typical configuration



Data Packet Format

◎**Communication data (information frame) format**

Data formats:

Data length:

◎

I^2C data+W/R	Function Code	Data Area	CRC
1byte	1 byte	N-byte	16-bit CRC (cyclic redundancy code)

Table 7:C Mod Bus part of the function code

Function Code	Definitions	Operation (binary)
0x03	Reading Register Data	Read one or more data registers
0x10	Write Multiple Registers	Multiple sets of binary data to write multiple registers

AM2320 Registers

Table 8: AM2320 Data Register Table

Register information	Address	Register information	Address	Register information	Address	Register information	Address
High humidity	0x00	Model High	0x08	Users register a high	0x10	Retention	0x18
Low humidity	0x01	Model Low	0x09	Users register a low	0x11	Retention	0x19
High temperature	0x02	The version number	0x0A	Users register 2 high	0x12	Retention	0x1A
Low temperature	0x03	Device ID (24-31) Bit	0x0B	Users register 2 low	0x13	Retention	0x1B
Retention	0x04	Device ID (24-31) Bit	0x0C	Retention	0x14	Retention	0x1C
Retention	0x05	Device ID (24-31) Bit	0x0D	Retention	0x15	Retention	0x1D
Retention	0x06	Device ID (24-31) Bit	0x0E	Retention	0x16	Retention	0x1E
Retention	0x07	Status Register	0x0F	Retention	0x17	Retention	0x1F

Sending Command to AM2320 to Read Data

1. Function code "03": Read registers multiplexed sensor

The host sends reading frame format:

START + (I^2C address + W) + function code (0x03) + start address + number of registers + STOP

Host read return data:

START + (I^2C address + R) + sequential read sensor data returned + STOP

Sensor response frame format:

Function code (0x03) + number + data +CRC [1]

For example: Host sequential read sensor data: the starting address for the register data of four sensors 0x00.

Sensor data register address and data:

Register Address	Register data	Data Description	Register Address	Register data	Data Description
0x00	0x01	High humidity	0x02	0x00	High temperature
0x01	0XF4	Low humidity	0x03	0xFA	Low temperature

Host message format sent:

The host sends	Byte count	Transmitting information	Remarks
Sensor address	1	0xB8	Sensor C address (0xB8) + W (0)
Function Code	1	0x03	Read register
Starting address	1	0x00	Register start address is 0x00
Number of registers	1	0x04	Read the number of register

AM2320 Respond with Data

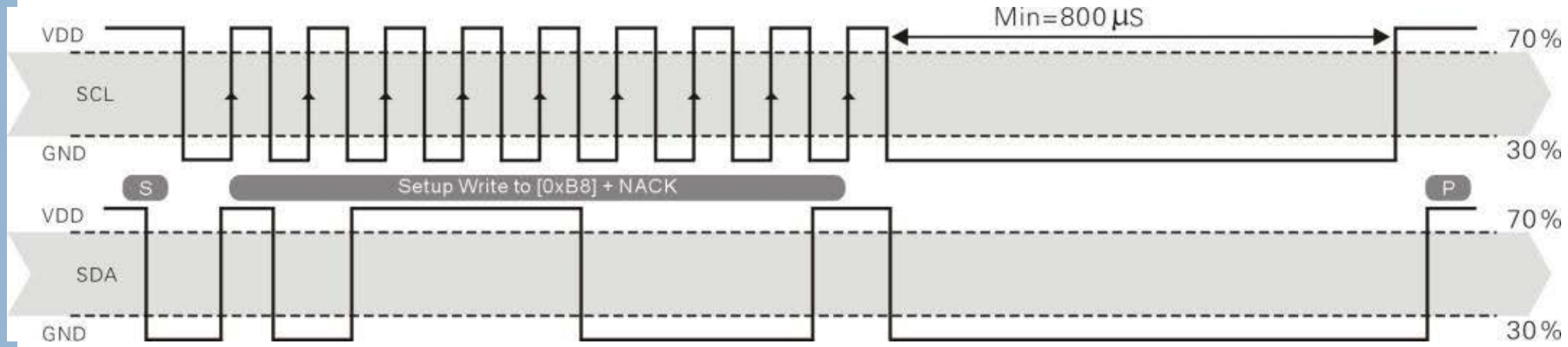
Slave response	Byte count	Transmitting information	Remarks
Function Code	1	0x03	Read register
Returns the number of bytes	1	0x04	Returns 4 of 4 byte register
Register 1	1	0x01	Address for the content of 0x00 (high humidity bytes)
Register 2	1	0XF4	Address for the content of 0x01 (low humidity bytes)
Register 3	1	0x00	Address for the content of 0x01 (low humidity bytes)
Register 4	1	0XFA	Address for the content 0x03 (temperature low byte)
CRC code	2	31A5	Sensors calculate the CRC code returned, low byte first;

Humidity: 01F4 = 1×256+15×16+4 =500 => humidity = 500÷10=50.0%RH;

Temperature: 00FA= 15×16+10 = 250 => temperature = 250÷10 = 25.0℃

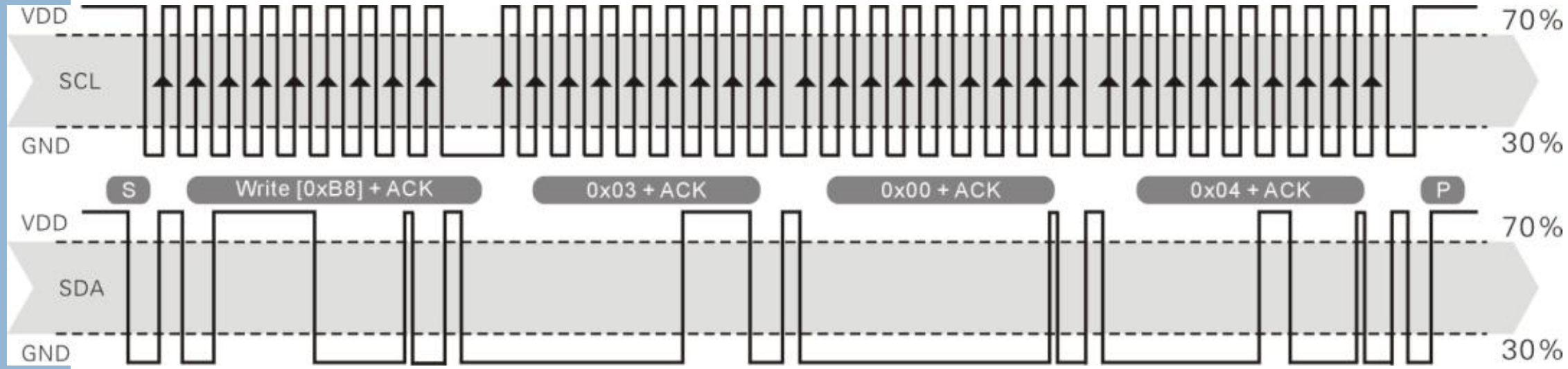
Timing Diagram : Step 1 Wake Sensor

- the starting signal + 0xB8 + wait (> 800us) + stop signal

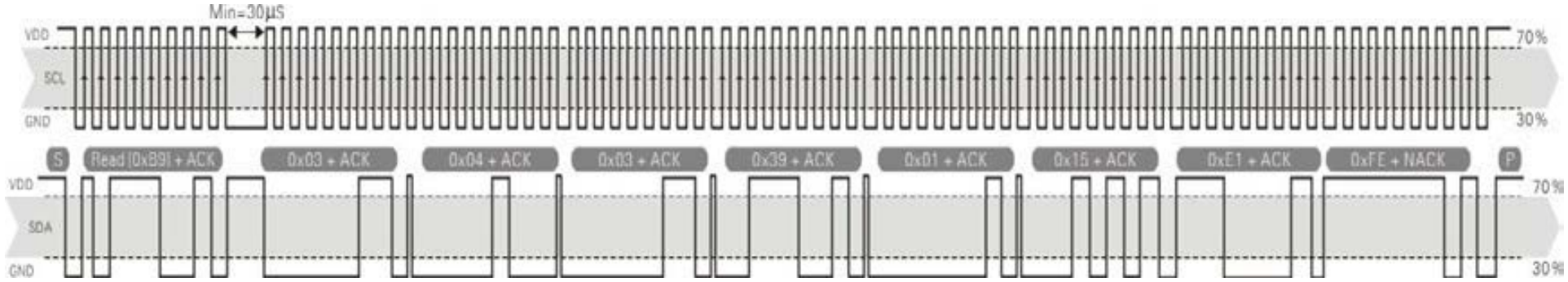


Timing Diagram : Step 2 Send Read Command

- The host sends commands to: START + 0xB8 (SLA) + 0x03 (function code) + 0x00 (starting address) +0x04 (register length) + STOP



Timing Diagram : Step 3 Receive Sensor Data

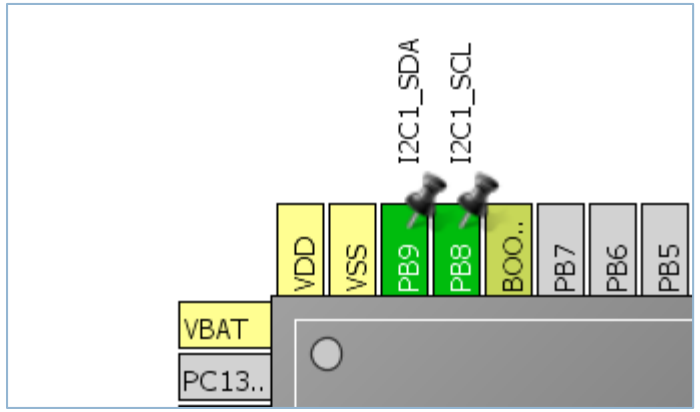


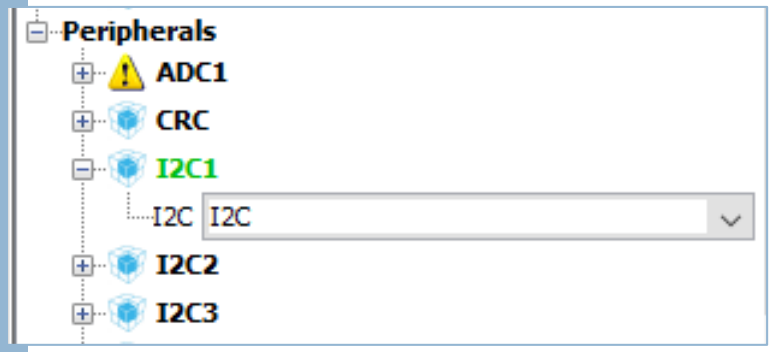
Host read back the data as follows:

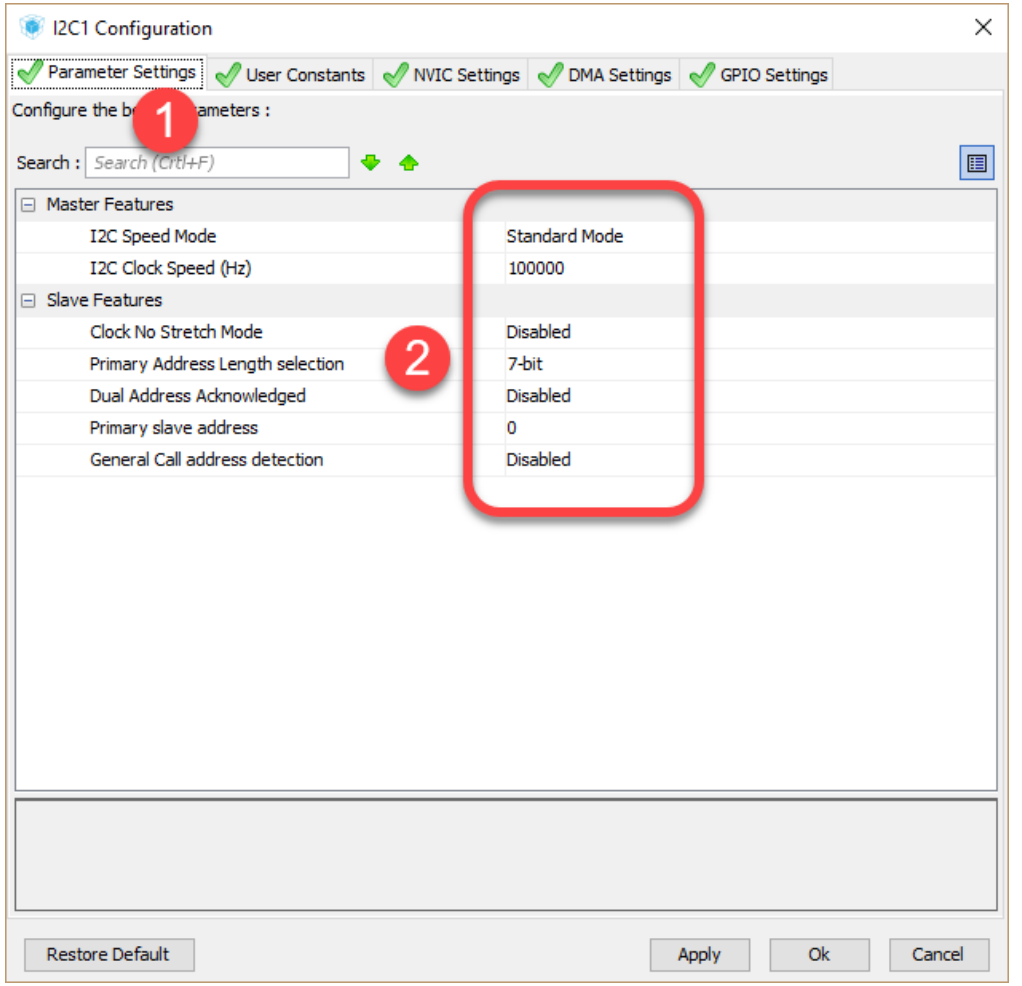
- **0x03** (Function Code) + **0x04** (data length) +
  0x03 (high humidity) + **0x39** (low humidity) +
  0x01 (high temperature) + **0x15** (low temperature) +
  0xE1 (CRC checksum low byte) + **0xFE** (CRC checksum high byte)
- Therefore: **0339H** = 825_{10} => humidity = 825÷10 = 82.5% RH

  0115H = 277_{10} => temperature = 277÷10 = 27.7 ℃

Setting STM32Cube for I2C (AM2320)







I2C HAL Lib

Polling mode IO operation

- Transmit in master mode an amount of data in blocking mode using HAL_I2C_Master_Transmit()
- Receive in master mode an amount of data in blocking mode using HAL_I2C_Master_Receive()
- Transmit in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Transmit()
- Receive in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Receive()

Interrupt mode IO operation

- Transmit in master mode an amount of data in non blocking mode using HAL_I2C_Master_Transmit_IT()
- At transmission end of transfer HAL_I2C_MasterTxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2C_MasterTxCpltCallback
- Receive in master mode an amount of data in non blocking mode using HAL_I2C_Master_Receive_IT()
- At reception end of transfer HAL_I2C_MasterRxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2C_MasterRxCpltCallback
- Transmit in slave mode an amount of data in non blocking mode using HAL_I2C_Slave_Transmit_IT()
- At transmission end of transfer HAL_I2C_SlaveTxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2C_SlaveTxCpltCallback
- Receive in slave mode an amount of data in non blocking mode using HAL_I2C_Slave_Receive_IT()
- At reception end of transfer HAL_I2C_SlaveRxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2C_SlaveRxCpltCallback
- In case of transfer Error, HAL_I2C_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_I2C_ErrorCallback
- Abort a master I2C process communication with Interrupt using HAL_I2C_Master_Abort_IT()
- End of abort process, HAL_I2C_AbortCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_AbortCpltCallback()

Declaration main.c

```
39  /* Includes ------------------------------------------------------------------*/
40  #include "main.h"
41  #include "stm32f4xx_hal.h"
42
43  /* USER CODE BEGIN Includes */
44  #include "AM2320.h"
45  #include "string.h"
46  /* USER CODE END Includes */
47
48  /* Private variables ---------------------------------------------------------*/
49  I2C_HandleTypeDef hi2c1;
50
51  UART_HandleTypeDef huart2;
52
53  /* USER CODE BEGIN PV */
54  /* Private variables ---------------------------------------------------------*/
55  float h=30.0,t=40.0;
56  uint8_t step = 0;
57  HAL_StatusTypeDef status;
58  /* USER CODE END PV */
59
60  /* Private function prototypes -----------------------------------------------*/
61  void SystemClock_Config(void);
62  static void MX_GPIO_Init(void);
63  static void MX_USART2_UART_Init(void);
64  static void MX_I2C1_Init(void);
65
66  /* USER CODE BEGIN PFP */
67  /* Private function prototypes -----------------------------------------------*/
68  uint16_t CRC16_2(uint8_t *, uint8_t );
69  /* USER CODE END PFP */
70
```

Declare local variables

```
80  int main(void)
81  {
82    /* USER CODE BEGIN 1 */
83    char str[50];
84    uint8_t cmdBuffer[3];
85    uint8_t dataBuffer[8];
86
87    /* USER CODE END 1 */
```

Setting before while loop

```
105   /* Initialize all configured peripherals */
106   MX_GPIO_Init();
107   MX_USART2_UART_Init();
108   MX_I2C1_Init();
109   /* USER CODE BEGIN 2 */
110
111
112     sprintf(str, "\n\rAM2320 I2C DEMO Starting . . .\n\r");
113     //while(__HAL_UART_GET_FLAG(&huart2,UART_FLAG_TC)==RESET){}
114     HAL_UART_Transmit(&huart2, (uint8_t*) str, strlen(str),200);
115
116     cmdBuffer[0] = 0x03;
117     cmdBuffer[1] = 0x00;
118     cmdBuffer[2] = 0x04;
119
120   /* USER CODE END 2 */
121
```

```
126   while (1)
127   {
128
129   /* USER CODE END WHILE */
130
131   /* USER CODE BEGIN 3 */
132
133     //Send Temp & Humid via UART2
134     sprintf(str, "Temperature = %4.1f\tHumidity = %4.1f\n\r", t, h);
135     while(__HAL_UART_GET_FLAG(&huart2,UART_FLAG_TC)==RESET){}
136     HAL_UART_Transmit(&huart2, (uint8_t*) str, strlen(str),200);
137
138     HAL_Delay(5000); //>3000 ms
139     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
140
141     //Wake up sensor
142     HAL_I2C_Master_Transmit(&hi2c1, 0x5c<<1, cmdBuffer, 3, 200);
143     //Send reading command
144     HAL_I2C_Master_Transmit(&hi2c1, 0x5c<<1, cmdBuffer, 3, 200);
145
146     HAL_Delay(1);
147
148     //Receive sensor data
149     HAL_I2C_Master_Receive(&hi2c1, 0x5c<<1, dataBuffer, 8, 200);
150
151     uint16_t Rcrc = dataBuffer[7] << 8;
152     Rcrc += dataBuffer[6];
153     if (Rcrc == CRC16_2(dataBuffer, 6)) {
154       uint16_t temperature = ((dataBuffer[4] & 0x7F) << 8) + dataBuffer[5];
155       t = temperature / 10.0;
156       t = (((dataBuffer[4] & 0x80) >> 7)== 1) ? (t * (-1)) : t ; // the temperature can be negative
157
158       uint16_t humidity = (dataBuffer[2] << 8) + dataBuffer[3];
159       h = humidity / 10.0;
160     }
161   }
162   /* USER CODE END 3 */
```

While loop

HAL_I2C_Master_Transmit

HAL_I2C_Master_Transmit

Function name	**HAL_StatusTypeDef HAL_I2C_Master_Transmit (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)**
Function description	Transmits in master mode an amount of data in blocking mode.
Parameters	• **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • **DevAddress:** Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • **pData:** Pointer to data buffer • **Size:** Amount of data to be sent • **Timeout:** Timeout duration
Return values	• **HAL:** status

HAL_I2C_Master_Receive

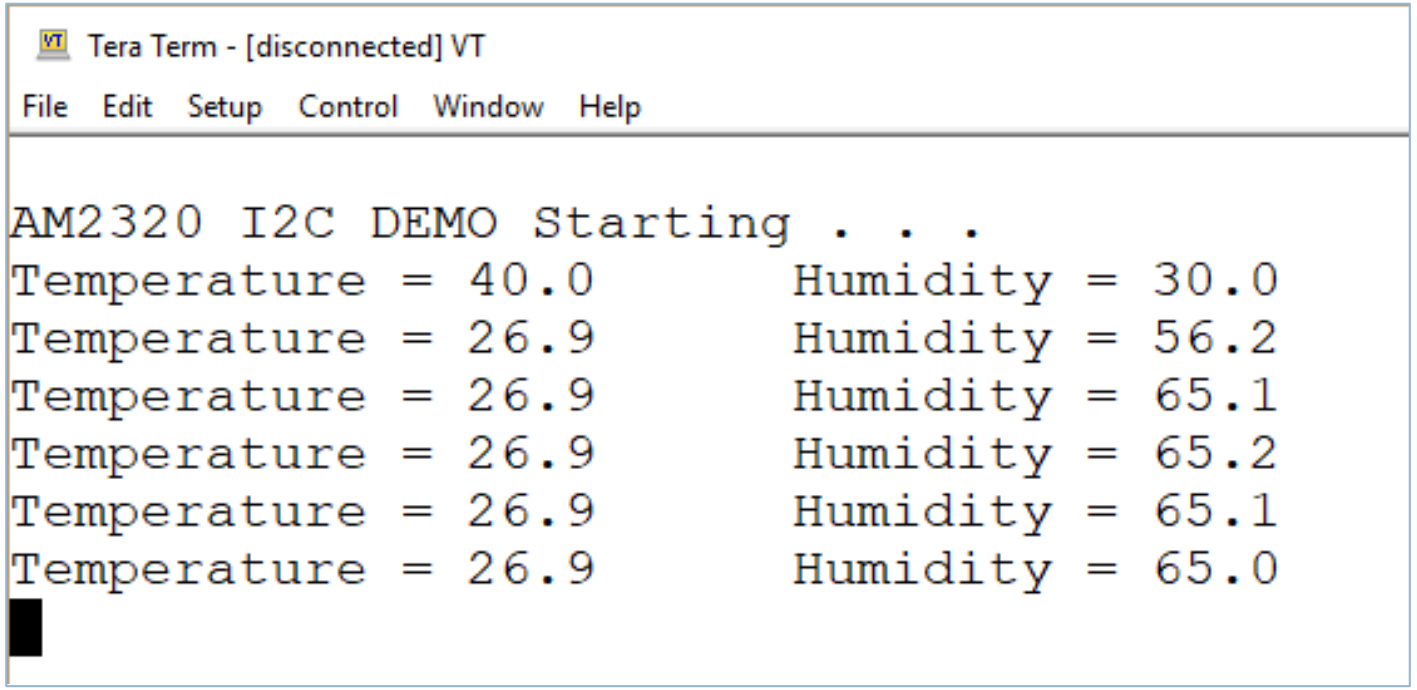
HAL_I2C_Master_Receive

Function name	**HAL_StatusTypeDef HAL_I2C_Master_Receive (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)**
Function description	Receives in master mode an amount of data in blocking mode.
Parameters	• **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • **DevAddress:** Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • **pData:** Pointer to data buffer • **Size:** Amount of data to be sent • **Timeout:** Timeout duration
Return values	• **HAL:** status

CRC Checksum

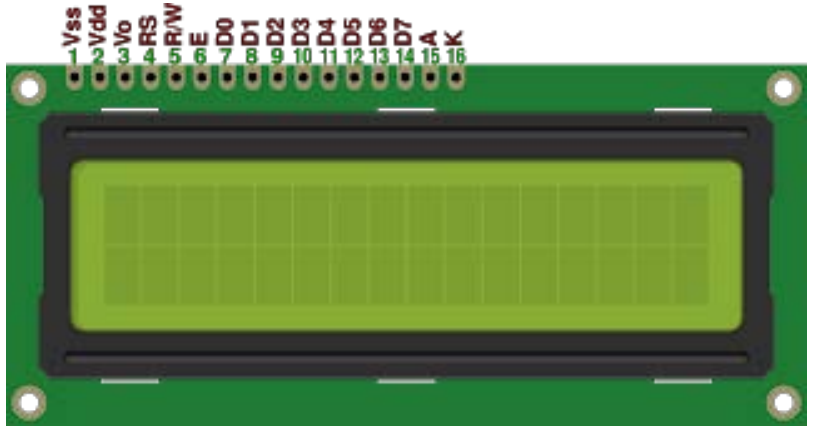
```
291  /* USER CODE BEGIN 4 */
292  uint16_t CRC16_2(uint8_t *ptr, uint8_t length)
293  {
294        uint16_t  crc = 0xFFFF;
295        uint8_t   s   = 0x00;
296
297        while(length--) {
298          crc ^= *ptr++;
299          for(s = 0; s < 8; s++) {
300            if((crc & 0x01) != 0) {
301              crc >>= 1;
302              crc ^= 0xA001;
303            } else crc >>= 1;
304          }
305        }
306        return crc;
307  }
308  /* USER CODE END 4 */
```

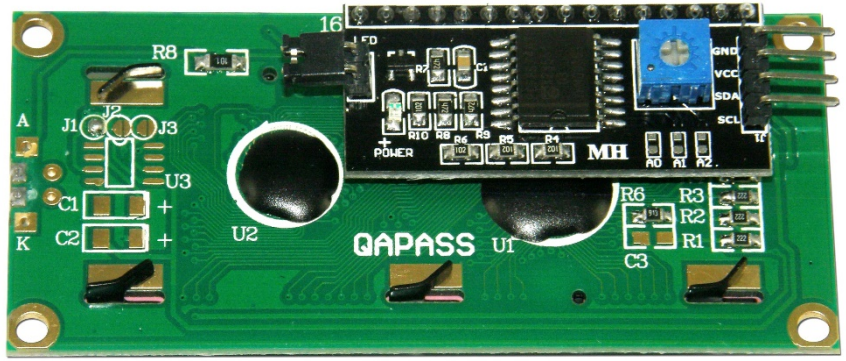
Show result in Tera Term via UART2



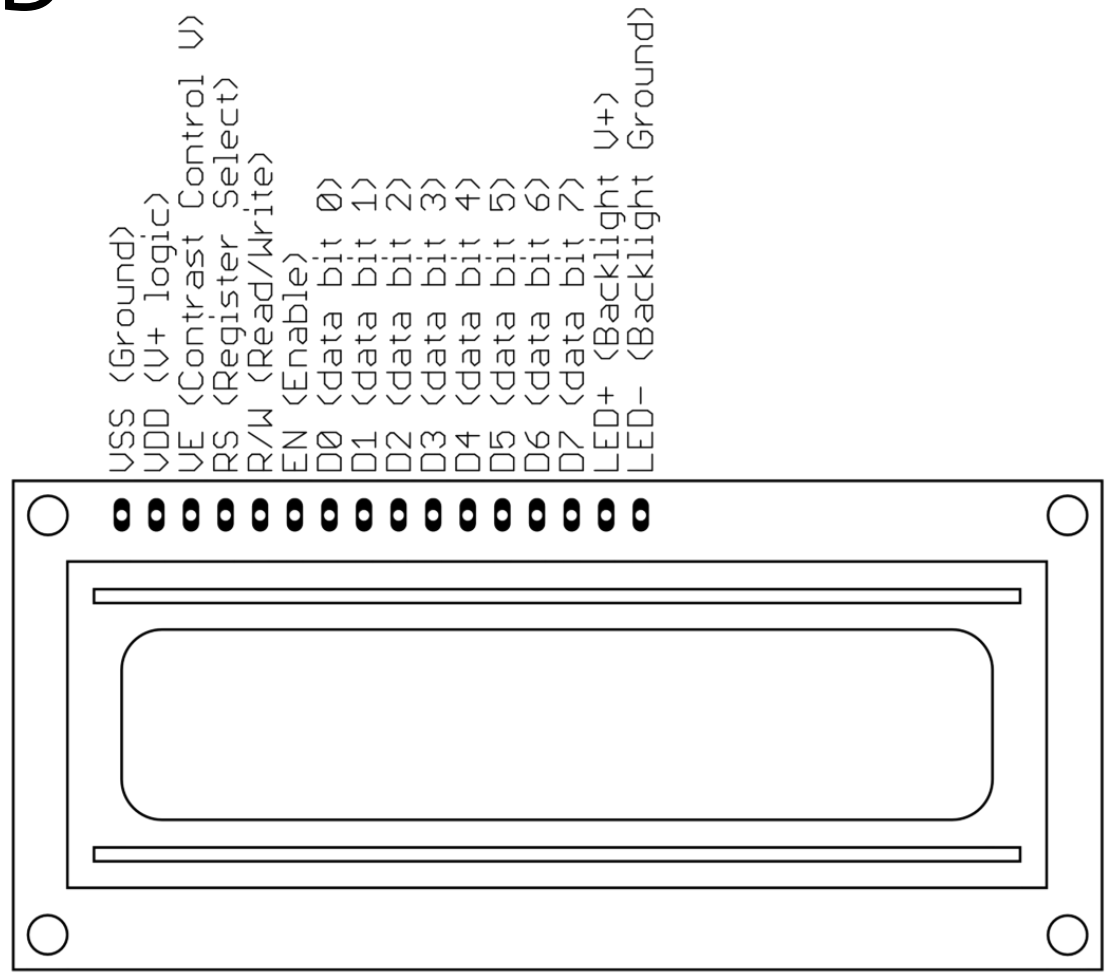
I2C LCD

- Character LCD
  - 16x2 characters
  - Parallel Connection
  - HD44780U LCD Controller
- I/O Expander
  - I2C
  - PCF8574





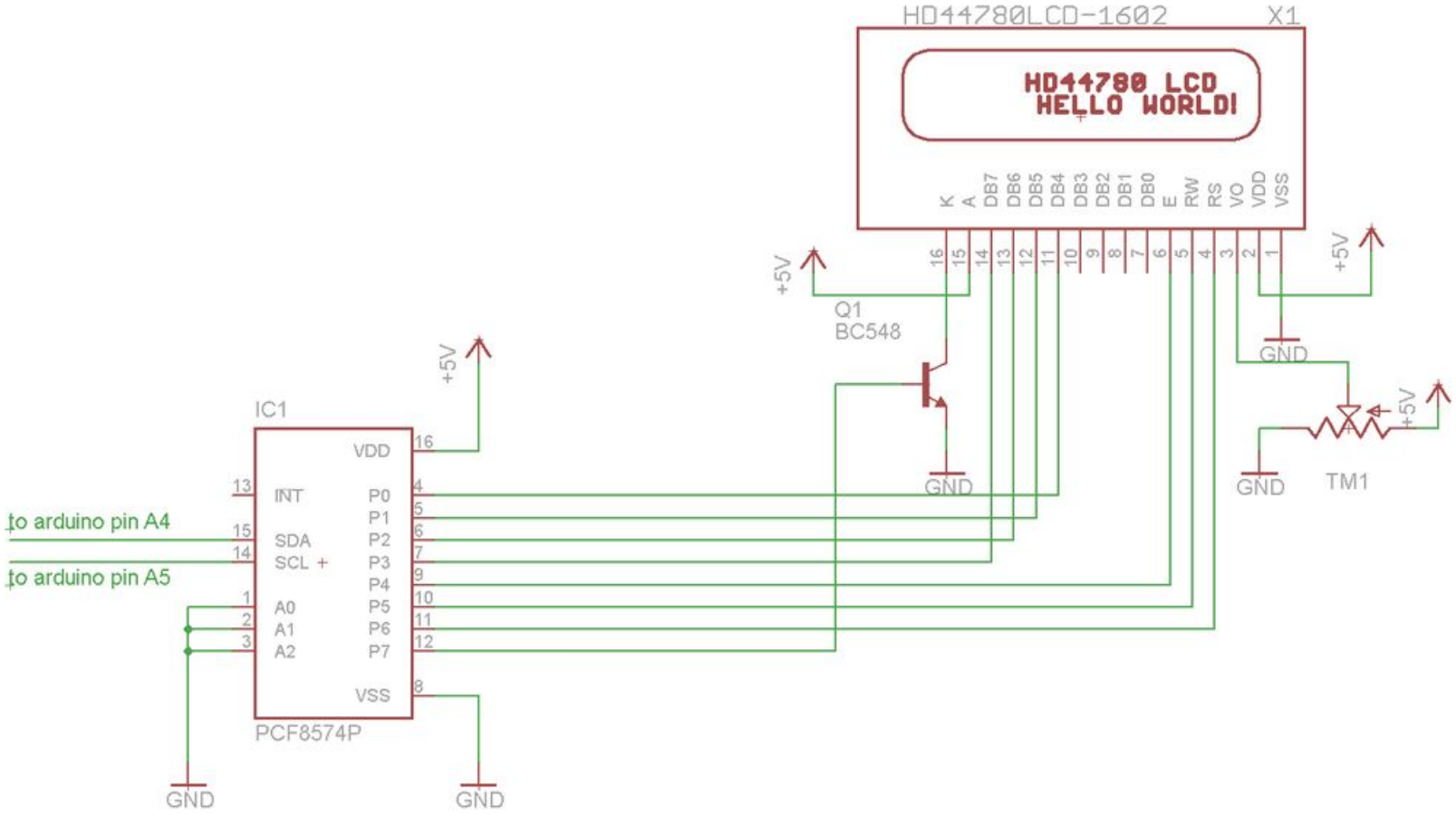
LCD



HD44780 EagleCad part by Adafruit Industries.

VSS (Ground) — POWER
VDD (V+ logic) — POWER
VE (Contrast Control V) — CONTRAST
RS (Register Select) — REGISTER CONTROL
R/W (Read/Write) — REGISTER CONTROL
EN (Enable) — REGISTER CONTROL
D0 (data bit 0) — NOT USED
D1 (data bit 1) — NOT USED
D2 (data bit 2) — NOT USED
D3 (data bit 3) — NOT USED
D4 (data bit 4) — DATA CTRL (4 bits)
D5 (data bit 5) — DATA CTRL (4 bits)
D6 (data bit 6) — DATA CTRL (4 bits)
D7 (data bit 7) — DATA CTRL (4 bits)
LED+ (Backlight V+) — LED POWER
LED- (Backlight Ground) — LED POWER

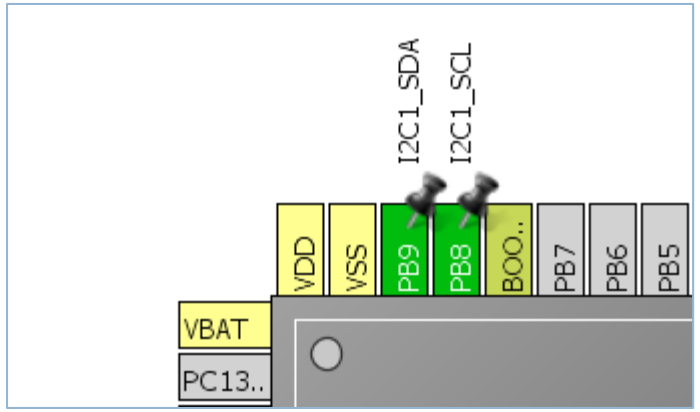
LCD Schematic

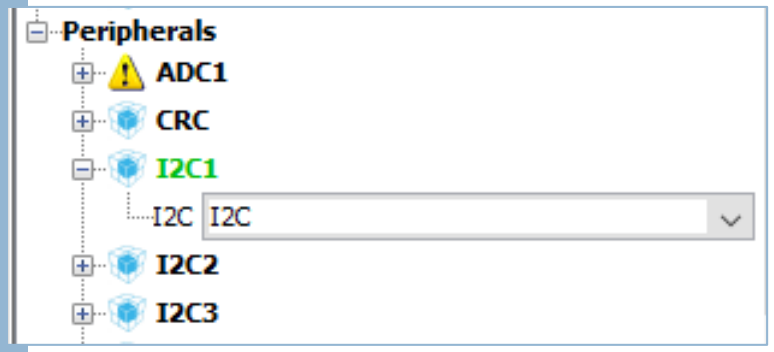


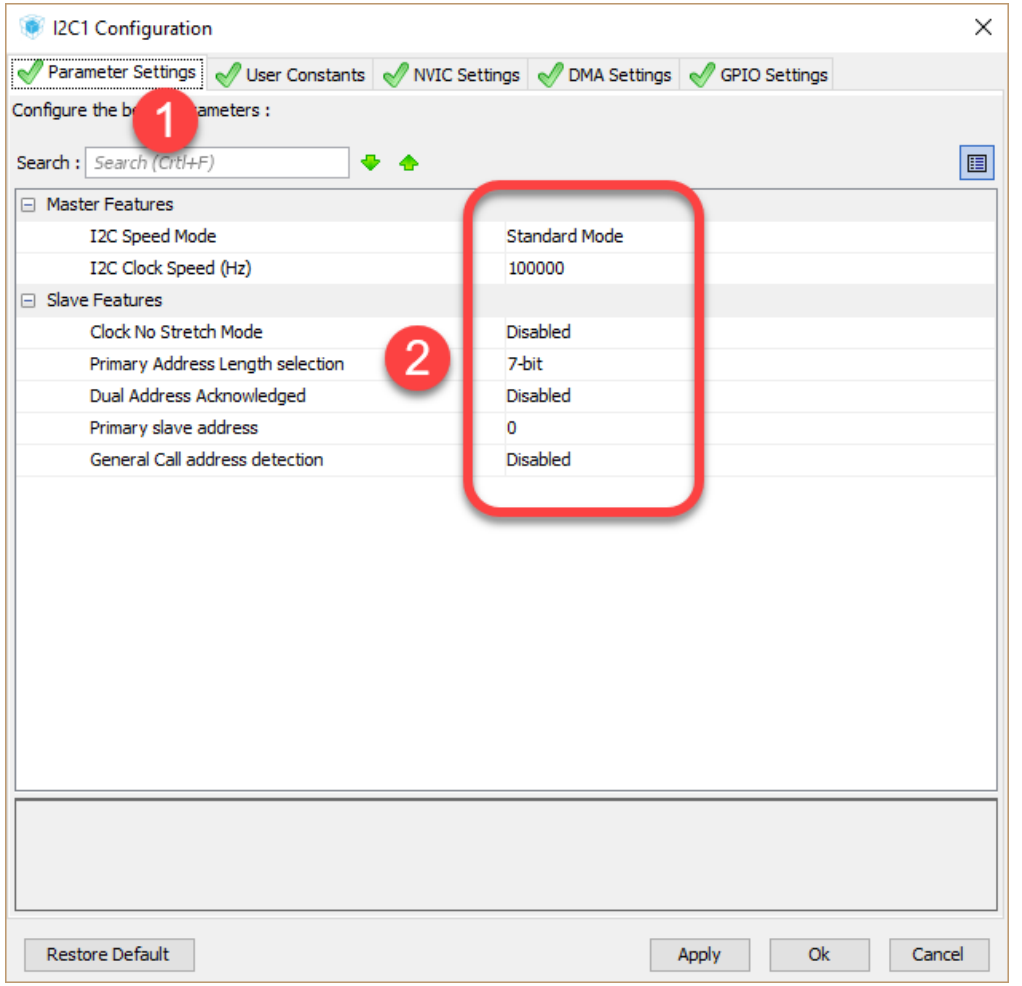
HD44780U LCD Controller

- Open Datasheet

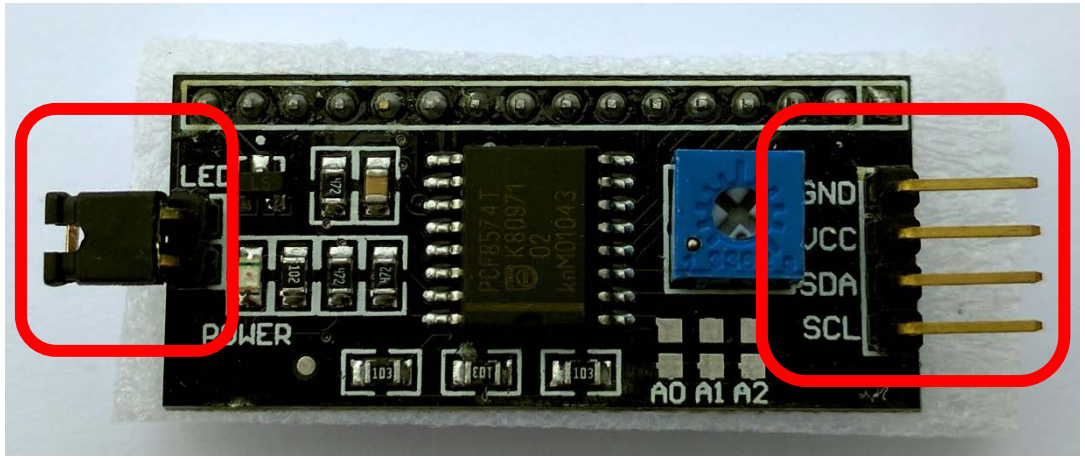
Setting STM32Cube for I2C







การเชื่อมต่อกับ I2C ของ LCD กับ STM32

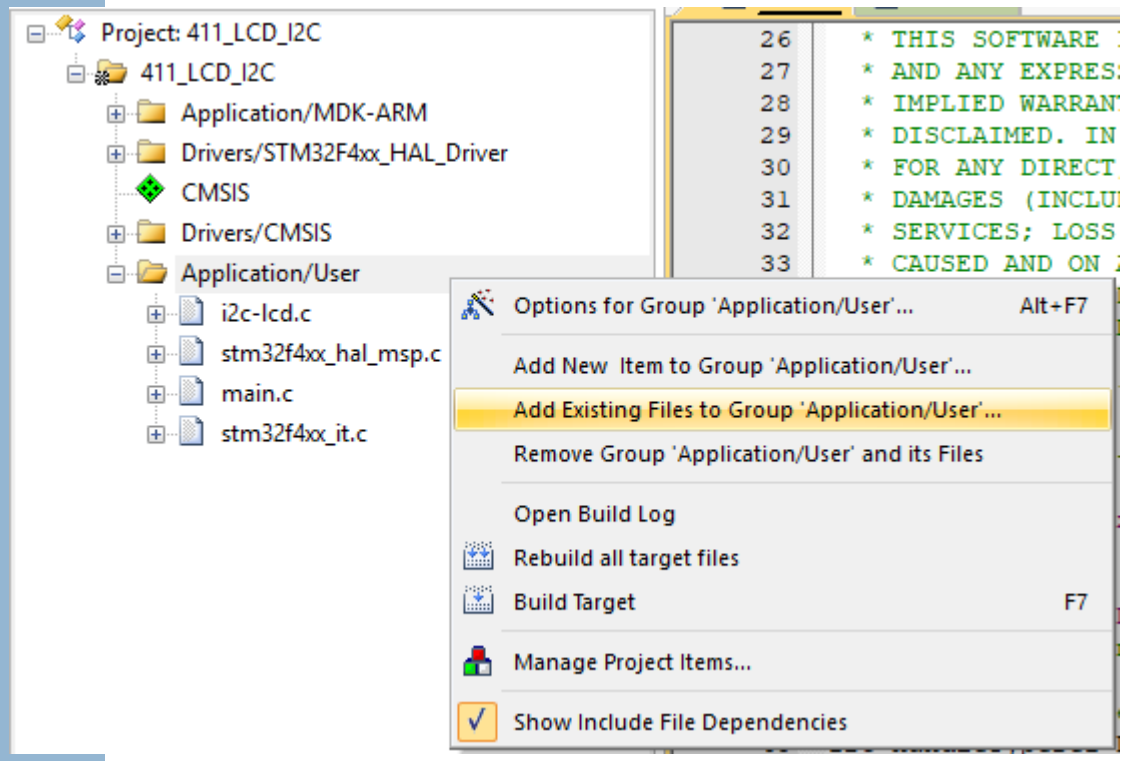


I2C LCD	STM32
GND	GND
Vcc	5V
SDA	PB8 (D14, I2C1_SDA)
SCL	PB9 (D15, I2C1_SCL)

คง Jumper : Backlight ติด

ถอด Jumper : Backlight ดับ

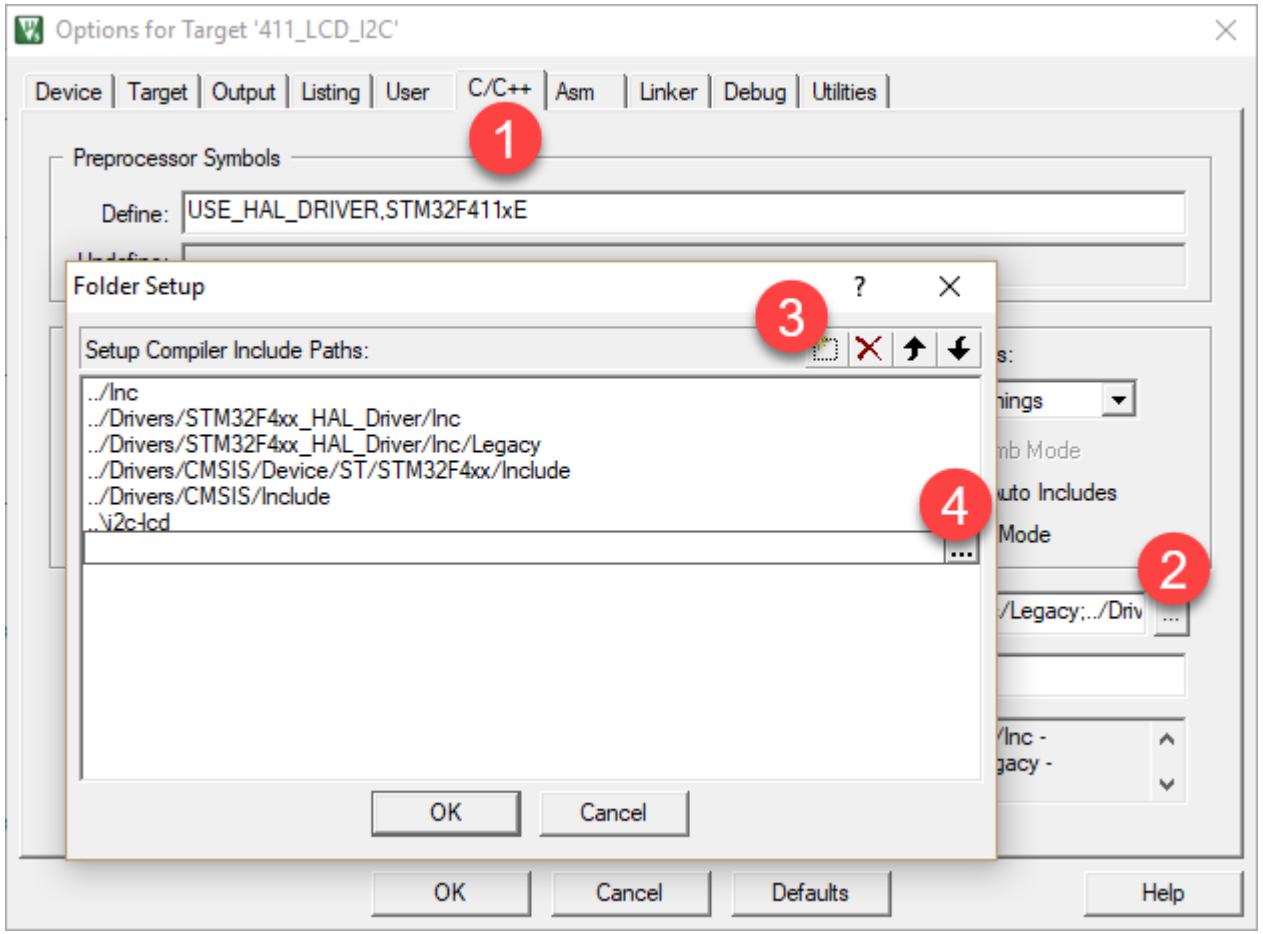
เพิ่ม i2c-lcd.c เข้าไปใน Project

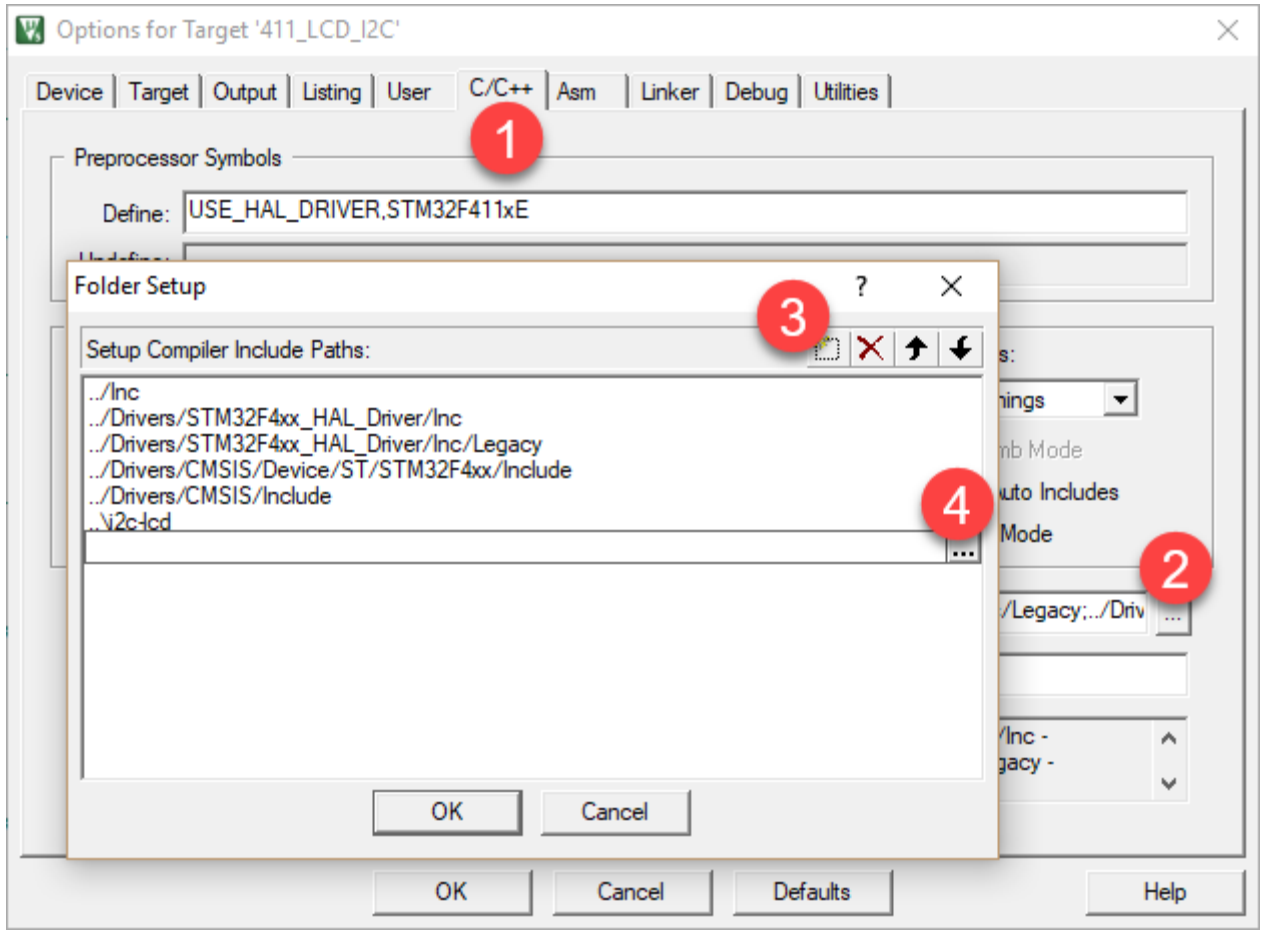


- คลิกขวาที่โฟลเดอร์ Application/User
- เลือก Add Existing Files
- เลือกไฟล์ i2c-lcd.c
- กด Add

เพิ่ม Path สำหรับ i2c-lcd.h

- เมนู Project -> Options for Target
- ทำขั้นตอน 1-4
- เลือกโฟลเดอร์ที่บรรจุไฟล์ i2c-lcd.h

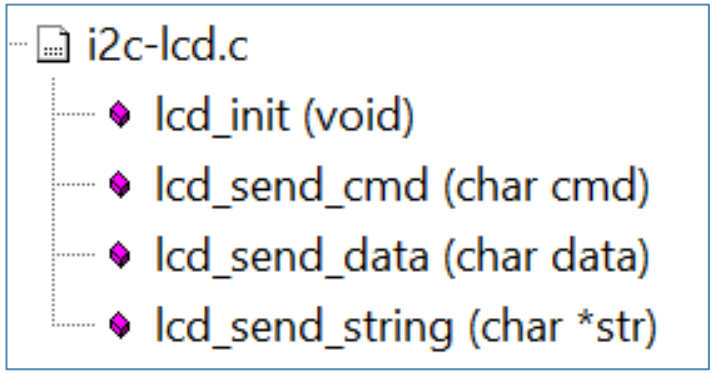



Options for Target '411_LCD_I2C'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
1
Preprocessor Symbols
Define:
USE_HAL_DRIVER,STM32F411xE
Folder Setup
3
Setup Compiler Include Paths:
../Inc
../Drivers/STM32F4xx_HAL_Driver/Inc
../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
../Drivers/CMSIS/Device/ST/STM32F4xx/Include
../Drivers/CMSIS/Include
../i2c-lcd
4
OK
Cancel
2
OK
Cancel
Defaults
Help

i2c-lcd.c

```
#include "i2c-lcd.h"
extern I2C_HandleTypeDef hi2c1;

//#define SLAVE_ADDRESS_LCD 0x3F<<1
#define SLAVE_ADDRESS_LCD 0x27<<1
```




```
void lcd_send_cmd (char cmd)
{
  char data_u, data_l;
  uint8_t data_t[4];
  data_u = (cmd&0xf0);
  data_l = ((cmd<<4)&0xf0);
  data_t[0] = data_u|0x0C;  //en=1, rs=0
  data_t[1] = data_u|0x08;  //en=0, rs=0
  data_t[2] = data_l|0x0C;  //en=1, rs=0
  data_t[3] = data_l|0x08;  //en=0, rs=0
  HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);
}
```

เพิ่ม Code ก่อนเข้า while loop

```
/* Private includes ----------
/* USER CODE BEGIN Includes */
#include "i2c-lcd.h"
#include <string.h>
#include <stdio.h>
/* USER CODE END Includes */
```

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_I2C1_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
I2C1_ADDR_Chk();
lcd_init ();
HAL_Delay(500);
lcd_send_cmd (0x01);  // clear the display
HAL_Delay(500);

lcd_send_string ("HELLO WORLD !!");
HAL_Delay(500);

lcd_send_cmd (0x01);  // clear the display
HAL_Delay(500);
/* USER CODE END 2 */
```

while loop code

```
112   while (1)
113   {
114
115   /* USER CODE END WHILE */
116
117   /* USER CODE BEGIN 3 */
118     lcd_send_cmd (0x80);  // cursor goes to line:1 col:1
119
120     lcd_send_string ("subscribe"); //display string
121
122     lcd_send_cmd (0xc0); // cursor goes line:2 col:1
123
124     lcd_send_string ("to this channel"); //diaplay string
125
126     HAL_Delay (2000);  // wait for 2 sec
127
128     lcd_send_cmd (0x01);  // clear the display
129
130     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
131
132     HAL_Delay (1000);
133
134
135     //Display Line 1
136     for(int i=0; i<=15; i++)
137     {
138       lcd_send_cmd (0x80+i);
139       lcd_send_string ("1");
140       HAL_Delay(200);
141
142     }
```

```
144     //Display Line 2
145     for(int i=0; i<=15; i++)
146     {
147
148       lcd_send_cmd (0xc0+i);
149       lcd_send_string ("2");
150       HAL_Delay(200);
151
152     }
153
154     lcd_send_cmd (0x01);  // clear the display
155     HAL_Delay(500);
156
157     lcd_send_cmd (0xc0+0xd);
158     lcd_send_string ("XYZ");
159     HAL_Delay(500);
160
161     //Shift Left
162     lcd_send_cmd (0x18);
163     HAL_Delay(500);
164
165     //Shift Left
166     lcd_send_cmd (0x18);
167     HAL_Delay(500);
168
169     //Shift Left
170     lcd_send_cmd (0x18);
171     HAL_Delay(500);
172
173     //Shift Right
174     lcd_send_cmd (0x1c);
175     HAL_Delay(500);
176
177     //Shift Right
178     lcd_send_cmd (0x1c);
179     HAL_Delay(500);
180
181     //Shift Right
182     lcd_send_cmd (0x1c);
183     HAL_Delay(500);
184
185   }
```

Scan Addresses

main.c

```
void I2C1_ADDR_Chk(void)
{
  uint8_t testAddr;
  uint8_t testData;
  char str[30];

  for (testAddr=0; testAddr<=127; testAddr++)
  {
    if ( HAL_OK == HAL_I2C_Master_Transmit(&hi2c1, testAddr<<1, &testData, 1, 200))
    {
      sprintf(str, "=====> Address 0x%02X is found. <=====\r\n", testAddr);
      HAL_UART_Transmit(&huart2, (uint8_t *) str, strlen(str), 300);

    } else {
      sprintf(str, "Address 0x%02X is not found.\r\n", testAddr);
      HAL_UART_Transmit(&huart2, (uint8_t *) str, strlen(str), 300);

    }

    //if slave device is quite slow use HAL_Delay
    //HAL_Delay(200);

  }

}
```

```
Address 0x23 is not found.
Address 0x24 is not found.
Address 0x25 is not found.
Address 0x26 is not found.
=====> Address 0x27 is found. <=====
Address 0x28 is not found.
Address 0x29 is not found.
Address 0x2A is not found.
Address 0x2B is not found.
```