

Chapter 1 LCD1602

In this chapter, we will learn about the LCD1602 Display Screen.

Project 1.1 I2C LCD1602

In this section we learn how to use lcd1602 to display something.

GPIO

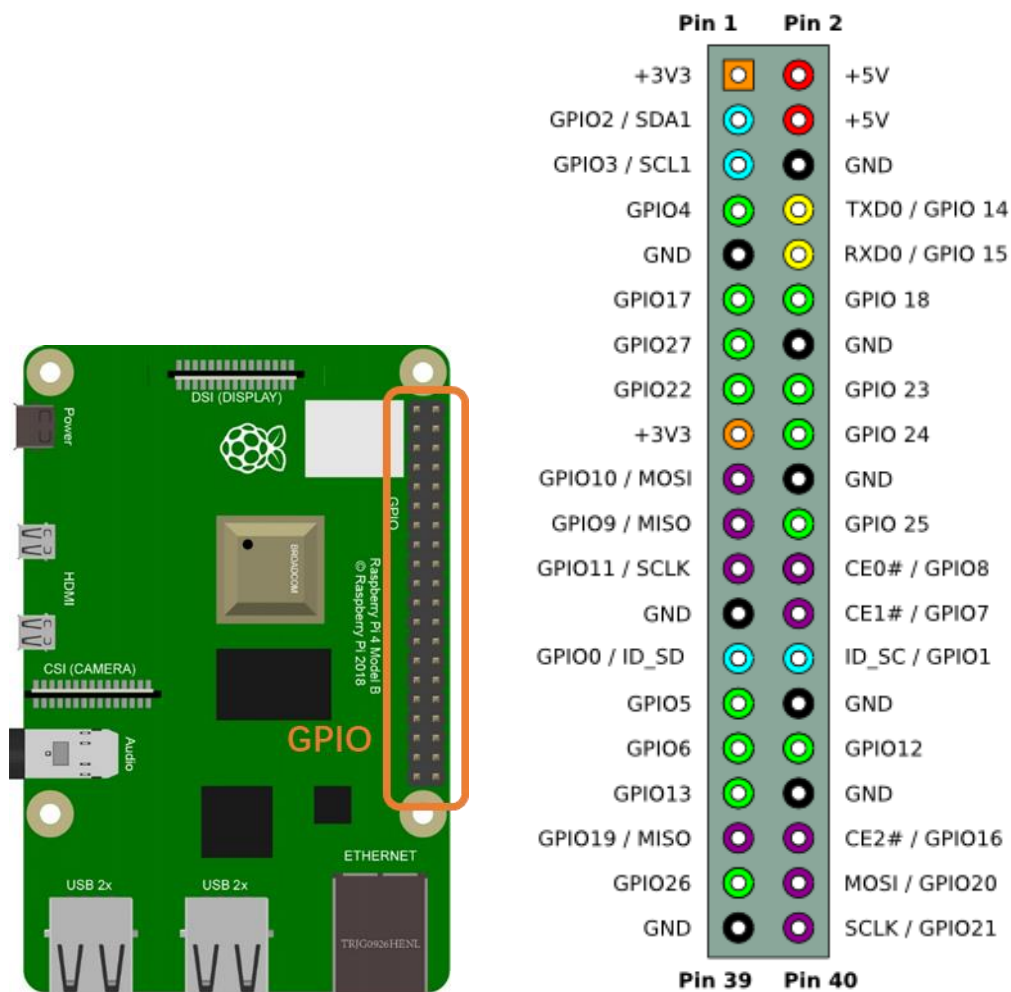
GPIO: General Purpose Input/Output. Here we will introduce the specific function of the pins on the Raspberry Pi and how you can utilize them in all sorts of ways in your projects. Most RPi Module pins can be used as either an input or output, depending on your program and its functions.

When programming GPIO pins there are 3 different ways to reference them: GPIO Numbering, Physical Numbering and WiringPi GPIO Numbering.

BCM GPIO Numbering

The Raspberry Pi CPU uses Broadcom (BCM) processing chips BCM2835, BCM2836 or BCM2837. GPIO pin numbers are assigned by the processing chip manufacturer and are how the computer recognizes each pin. The pin numbers themselves do not make sense or have meaning as they are only a form of identification. Since their numeric values and physical locations have no specific order, there is no way to remember them so you will need to have a printed reference or a reference board that fits over the pins.

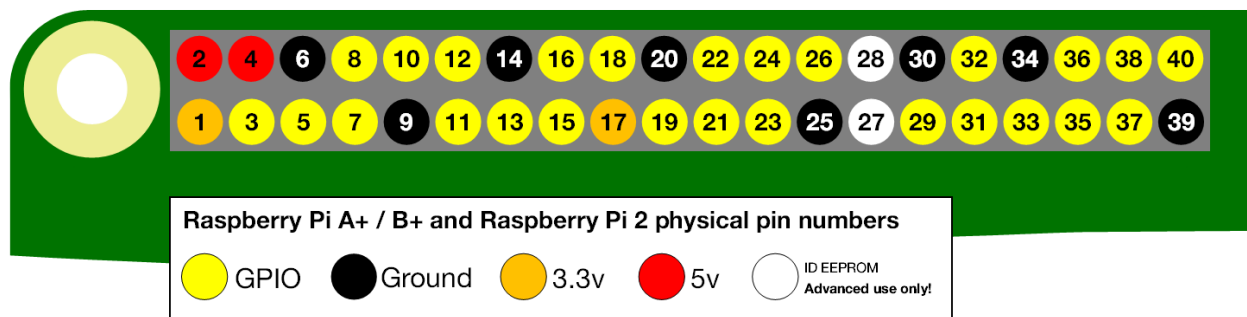
Each pin's functional assignment is defined in the image below:



For more details about pin definition of GPIO, please refer to <http://pinout.xyz/>

PHYSICAL Numbering

Another way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'Physical Numbering', as shown below:



WiringPi GPIO Numbering

Different from the previous two types of GPIO serial numbers, RPi GPIO serial number of the WiringPi are numbered according to the BCM chip use in RPi.

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	
—	—	3.3v	1 2	5v	—	—	For Pi B
8	R1:0/R2:2	SDA	3 4	5v	—	—	
9	R1:1/R2:3	SCL	5 6	0v	—	—	
7	4	GPIO7	7 8	TxD	14	15	
—	—	0v	9 10	RxD	15	16	
0	17	GPIO0	11 12	GPIO1	18	1	
2	R1:21/R2:27	GPIO2	13 14	0v	—	—	
3	22	GPIO3	15 16	GPIO4	23	4	
—	—	3.3v	17 18	GPIO5	24	5	
12	10	MOSI	19 20	0v	—	—	
13	9	MISO	21 22	GPIO6	25	6	
14	11	SCLK	23 24	CE0	8	10	
—	—	0v	25 26	CE1	7	11	
30	0	SDA.0	27 28	SCL.0	1	31	
21	5	GPIO.21	29 30	0V	—	—	
22	6	GPIO.22	31 32	GPIO.26	12	26	
23	13	GPIO.23	33 34	0V	—	—	
24	19	GPIO.24	35 36	GPIO.27	16	27	
25	26	GPIO.25	37 38	GPIO.28	20	28	
—	—	0V	39 40	GPIO.29	21	29	
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	

(For more details, please refer to <https://projects.drogon.net/raspberry-pi/wiringpi/pins/>)

You can also use the following command to view their correlation.

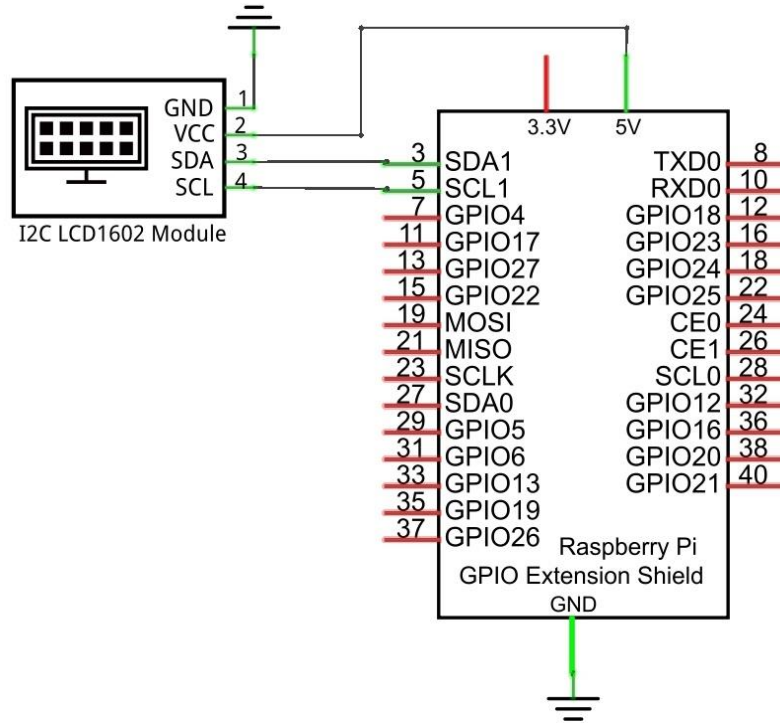
```
gpio readall
```

Pi 4B											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALT0	1	3	4		5v			
3	9	SCL.1	ALT0	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	

Circuit

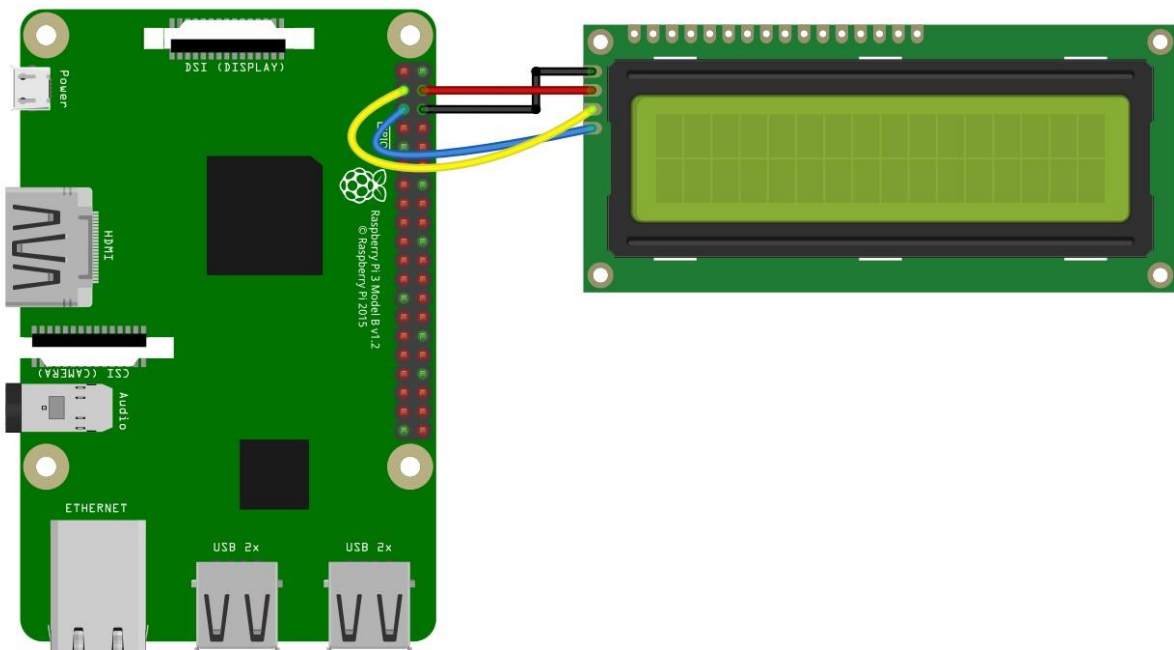
Note that the power supply for I2C LCD1602 in this circuit is 5V.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

NOTE: It is necessary to configure I2C and install Smbus first.



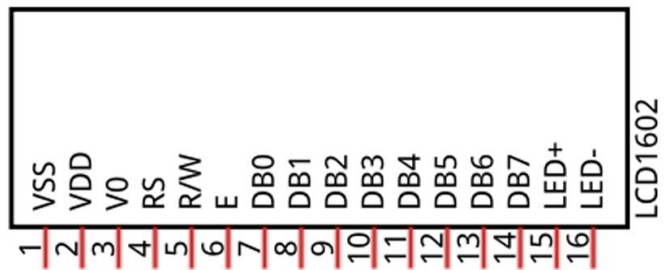
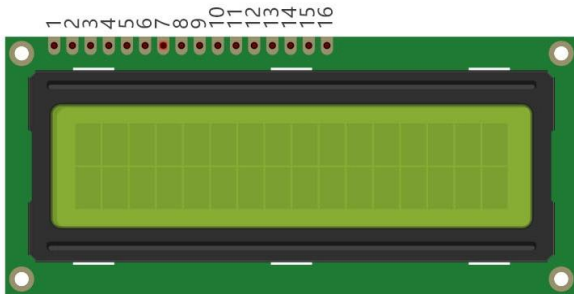
Component knowledge

I2C communication

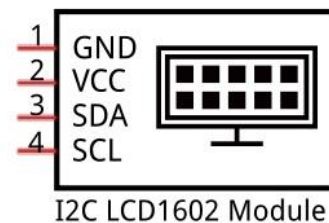
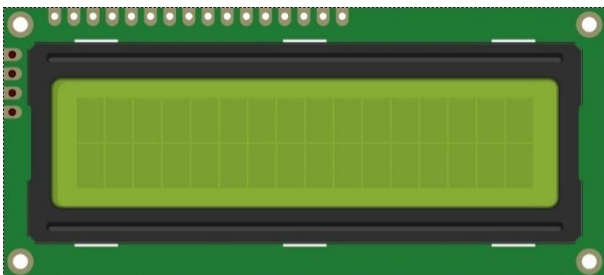
I2C (Inter-Integrated Circuit) has a two-wire serial communication mode, which can be used to connect a micro-controller and its peripheral equipment. Devices using I2C communications must be connected to the serial data line (SDA), and serial clock line (SCL) (called I2C bus). Each device has a unique address which can be used as a transmitter or receiver to communicate with devices connected via the bus.

LCD1602 communication

There are LCD1602 display screen and the I2C LCD. We will introduce both of them in this chapter. But what we use in this project is an I2C LCD1602 display screen. The LCD1602 Display Screen can display 2 lines of characters in 16 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 Display Screen along with its circuit pin diagram

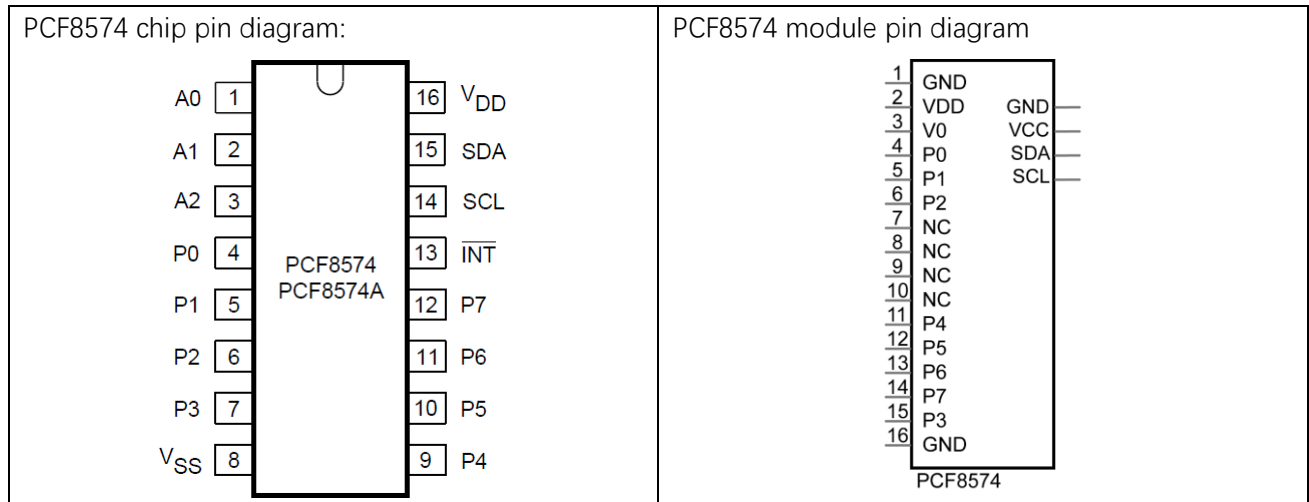


I2C LCD1602 Display Screen integrates a I2C interface, which connects the serial-input & parallel-output module to the LCD1602 Display Screen. This allows us to only use 4 lines to operate the LCD1602.

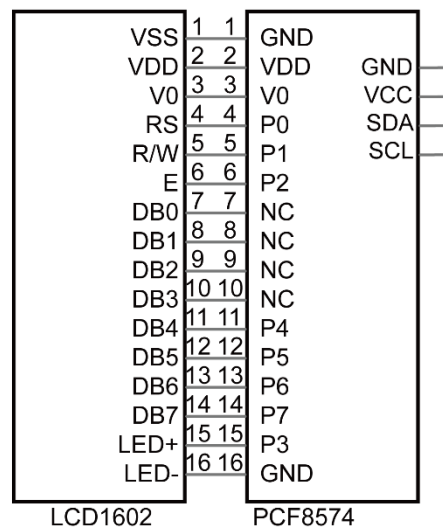


The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F). You can also view the RPI bus on your I2C device address through command "i2cdetect -y 1" (refer to the "configuration I2C" section below).

Below is the PCF8574 chip pin diagram and its module pin diagram:



PCF8574 module pins and LCD1602 pins correspond to each other and connected to each other:



Because of this, as stated earlier, we only need 4 pins to control the 16 pins of the LCD1602 Display Screen through the I2C interface.

In this project, we will use the I2C LCD1602 to display some static characters and dynamic variables.

Configure I2C and Install Smbus

If you have already configured I2C and installed Smbus, skip this section. If you have not, proceed with the following configuration.

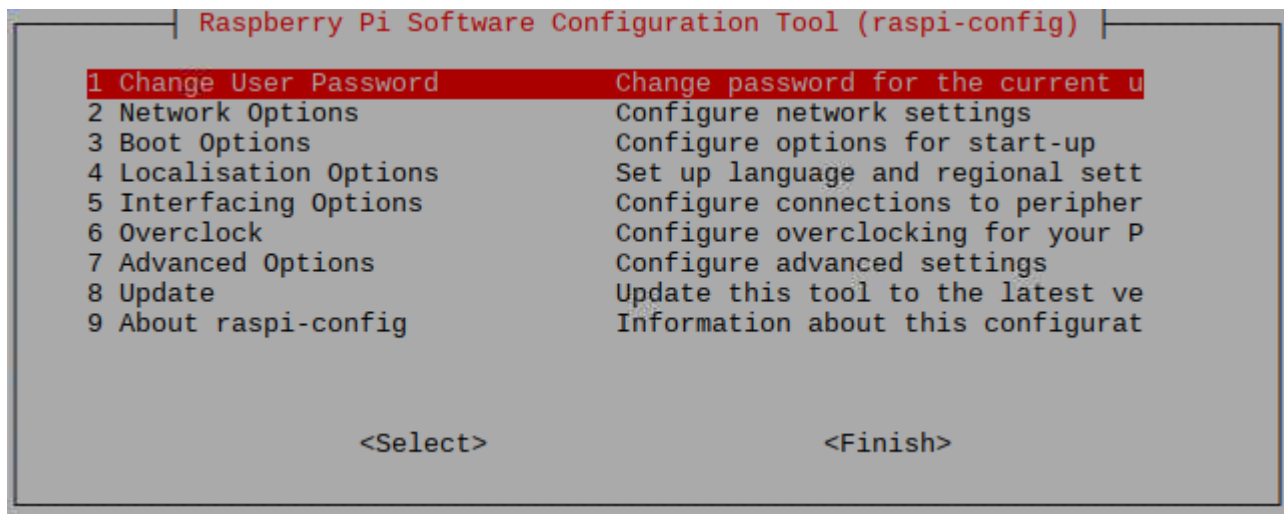
Enable I2C

The I2C interface in Raspberry Pi is disabled by default. You will need to open it manually and enable the I2C interface as follows:

Type command in the Terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose "5 Interfacing Options" then "P5 I2C" then "Yes" and then "Finish" in this order and restart your RPi. The I2C module will then be started.

Type a command to check whether the I2C module is started:

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown. "bcm2708" refers to the CPU model. Different models of Raspberry Pi display different contents depending on the CPU installed:

```
pi@raspberrypi:~$ lsmod | grep i2c
i2c_bcm2708      4770  0
i2c_dev         5859  0
pi@raspberrypi:~$
```


Install I2C-Tools

Next, type the command to install I2C-Tools. It is available with the Raspberry Pi OS by default.

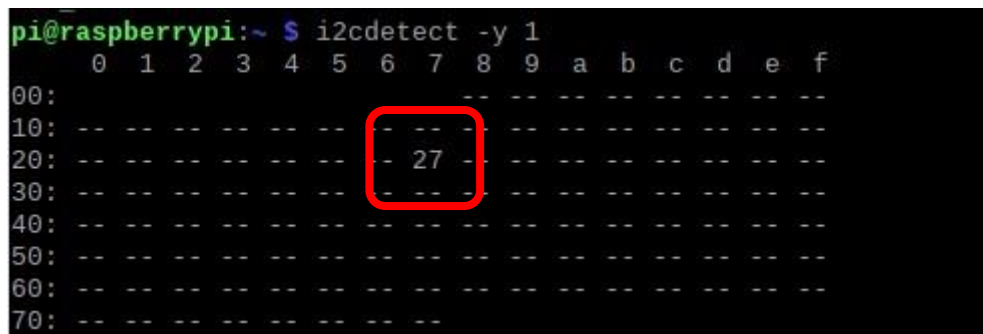
```
sudo apt-get install i2c-tools
```

I2C device address detection:

```
i2cdetect -y 1
```

When you use the serial-parallel IC chip PCF8574T, its I2C default address is 0x27. When the serial-parallel IC chip you use is PCF8574AT, its I2C default address is 0x3F.

When you use the serial-parallel IC chip PCF8574T, the result should look like this:



A terminal window on a Raspberry Pi showing the output of the command `i2cdetect -y 1`. The output is a grid where the first column lists hexadecimal addresses from 00 to 70 in increments of 10. The top row lists hexadecimal digits 0 through f. The grid contains dashes for most addresses, indicating no device was detected at those addresses. At address 20, the digit '2' is present, and at address 27, the digit '7' is present. A red square highlights the '27' in the output, indicating the detected I2C address of the PCF8574T chip.

```
pi@raspberrypi:~$ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  2  7  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Here, 27 (HEX) is the I2C address of LCD2004 Module (PCF8574T).

When you are using PCF8574AT, and its default I2C address is 0x3F.

Install Smbus Module

```
sudo apt-get install python-smbus
```

```
sudo apt-get install python3-smbus
```

Code

This code will have your RPi's CPU temperature and System Time Displayed on the LCD1602.

C Code 1.1 I2CLCD1602

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

If you have any concerns, please contact us via: support@freenove.com

1. Use cd command to enter 1.1_ I2CLCD1602 directory of C code.

```
cd ~/Freenove_LCD_Module/Freenove_LCD_Module_for_Raspberry_Pi/C/C_Code/1.1_I2CLCD1602
```

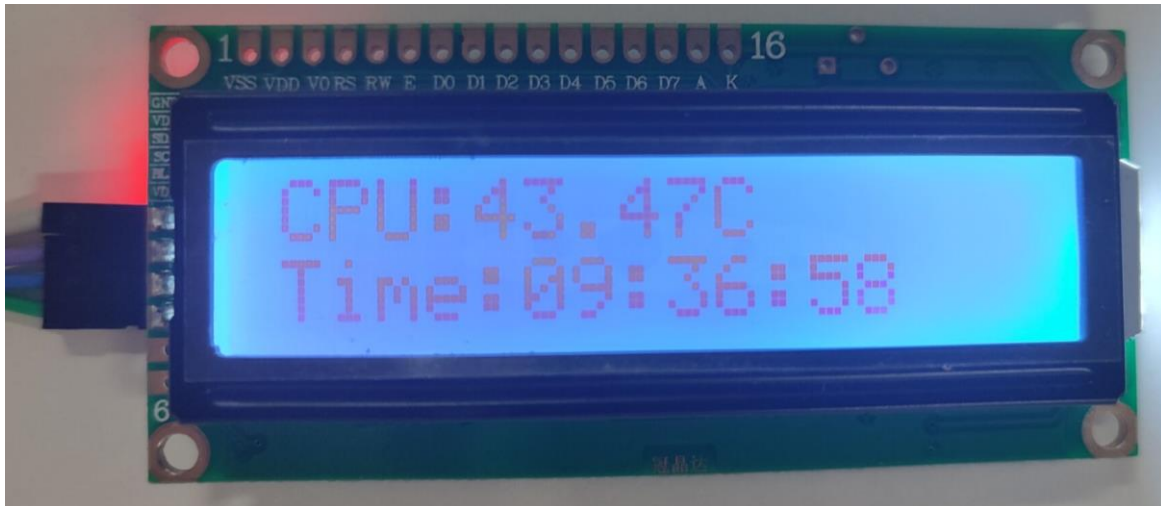
2. Use following command to compile "I2CLCD1602.c" and generate executable file "I2CLCD1602".

```
gcc I2CLCD1602.c -o I2CLCD1602 -lwiringPi -lwiringPiDev
```

3. Then run the generated file "I2CLCD1602".

```
sudo ./I2CLCD1602
```

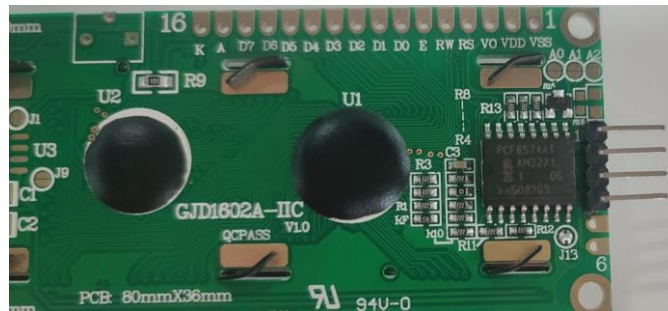
After the program is executed, the LCD1602 Screen will display your RPi's CPU Temperature and System Time.



NOTE: After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD1602 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.



In addition, if the back of your LCD module looks like the picture below, there is no need to adjust the contrast, this circuit has adjusted the contrast to a suitable value.



The following is the program code:

```

1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <wiringPi.h>
4  #include <wiringPiI2C.h>
5  #include <pcf8574.h>
6  #include <lcd.h>
7  #include <time.h>
8
9  int pcf8574_address = 0x27;          // PCF8574T:0x27, PCF8574AT:0x3F
10 #define BASE 64                      // BASE any number above 64
11 //Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.
12 #define RS      BASE+0
13 #define RW      BASE+1
14 #define EN      BASE+2
15 #define LED     BASE+3
16 #define D4      BASE+4
17 #define D5      BASE+5
18 #define D6      BASE+6
19 #define D7      BASE+7
20
21 int lcdhd; // used to handle LCD
22 void printCPUTemperature() { // sub function used to print CPU temperature
23     FILE *fp;
24     char str_temp[15];
25     float CPU_temp;
26     // CPU temperature data is stored in this directory.
27     fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
28     fgets(str_temp, 15, fp);          // read file temp
29     CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
30     printf("CPU's temperature : %.2f \n", CPU_temp);
31     lcdPosition(lcdhd, 0, 0);         // set the LCD cursor position to (0,0)
32     lcdPrintf(lcdhd, "CPU: %.2fC", CPU_temp); // Display CPU temperature on LCD

```

```

33     fclose(fp);
34 }
35 void printDateTime() { //used to print system time
36     time_t rawtime;
37     struct tm *timeinfo;
38     time(&rawtime); // get system time
39     timeinfo = localtime(&rawtime); //convert to local time
40     printf("%s \n", asctime(timeinfo));
41     lcdPosition(lcdhd, 0, 1); // set the LCD cursor position to (0,1)
42
43     lcdPrintf(lcdhd, "Time:%02d:%02d:%02d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
44     //Display system time on LCD
45 }
46 int detectI2C(int addr) { //Used to detect i2c address of LCD
47     int _fd = wiringPiI2CSetup (addr);
48     if (_fd < 0) {
49         printf("Error address : 0x%x \n", addr);
50         return 0 ;
51     }
52     else {
53         if(wiringPiI2CWrite(_fd, 0) < 0) {
54             printf("Not found device in address 0x%x \n", addr);
55             return 0;
56         }
57         else {
58             printf("Found device in address 0x%x \n", addr);
59             return 1 ;
60         }
61     }
62 }
63 int main(void) {
64     int i;
65     printf("Program is starting ... \n");
66     wiringPiSetup();
67     if(detectI2C(0x27)) {
68         pcf8574_address = 0x27;
69     } else if(detectI2C(0x3F)) {
70         pcf8574_address = 0x3F;
71     } else {
72         printf("No correct I2C address found, \n"
73             "Please use command 'i2cdetect -y 1' to check the I2C address! \n"
74             "Program Exit. \n");
75         return -1;
76     }

```

```

77  pcf8574Setup(BASE,pcf8574_address);//initialize PCF8574
78  for(i=0;i<8;i++) {
79      pinMode(BASE+i, OUTPUT);    //set PCF8574 port to output mode
80  }
81  digitalWrite(LED, HIGH);    //turn on LCD backlight
82  digitalWrite(RW, LOW);    //allow writing to LCD
83  lcdhd = lcdInit(2, 16, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return "handle"
used to handle LCD
84  if(lcdhd == -1) {
85      printf("lcdInit failed !");
86      return 1;
87  }
88  while(1) {
89      printCPUTemperature(); //print CPU temperature
90      printDateTime();    // print system time
91      delay(1000);
92  }
93  return 0;
94  }

```

First, define the I2C address of the PCF8574 and the Extension of the GPIO pin, which is connected to the GPIO pin of the LCD1602. LCD1602 has two different i2c addresses. Set 0x27 as default.

```

int pcf8574_address = 0x27;    // PCF8574T:0x27, PCF8574AT:0x3F
#define BASE 64    // BASE any number above 64
//Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.
#define RS    BASE+0
#define RW    BASE+1
#define EN    BASE+2
#define LED    BASE+3
#define D4    BASE+4
#define D5    BASE+5
#define D6    BASE+6
#define D7    BASE+7

```

Then, in main function, initialize the PCF8574, set all the pins to output mode, and turn ON the LCD1602 backlight (without the backlight the Display is difficult to read).

```

pcf8574Setup(BASE,pcf8574_address);// initialize PCF8574
for(i=0;i<8;i++) {
    pinMode(BASE+i, OUTPUT);    // set PCF8574 port to output mode
}
digitalWrite(LED, HIGH);    // turn on LCD backlight

```

Then use lcdInit() to initialize LCD1602 and set the RW pin of LCD1602 to 0 (can be written) according to requirements of this function. The return value of the function called "Handle" is used to handle LCD1602 ".

```

    lcdhd = lcdInit(2, 16, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return
    "handle" used to handle LCD

```

Details about lcdInit():

**int lcdInit (int rows, int cols, int bits, int rs, int strb,
int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7) ;**

This is the main initialization function and must be executed first before you use any other LCD functions. **Rows** and **cols** are the rows and columns of the Display (e.g. 2, 16 or 4, 20). **Bits** is the number of how wide the number of bits is on the interface (4 or 8). The **rs** and **strb** represent the pin numbers of the Display's RS pin and Strobe (E) pin. The parameters **d0** through **d7** are the pin numbers of the 8 data pins connected from the RPi to the display. Only the first 4 are used if you are running the display in 4-bit mode.

The return value is the 'handle' to be used for all subsequent calls to the lcd library when dealing with that LCD, or -1 to indicate a fault (usually incorrect parameter)

For more details about LCD Library, please refer to: <https://projects.drogon.net/raspberry-pi/wiringpi/lcd-library/>

In the next "while", two subfunctions are called to display the RPi's CPU Temperature and the SystemTime. First look at subfunction printCPUTemperature(). The CPU temperature data is stored in the `"/sys/class/thermal/thermal_zone0/temp"` file. We need to read the contents of this file, which converts it to temperature value stored in variable CPU_temp and uses lcdPrintf() to display it on LCD.

```
void printCPUTemperature() { //subfunction used to print CPU temperature

    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
    fgets(str_temp, 15, fp);    // read file temp
    CPU_temp = atof(str_temp)/1000.0;    // convert to Celsius degrees
    printf("CPU's temperature : %.2f \n", CPU_temp);
    lcdPosition(lcdhd, 0, 0);    // set the LCD cursor position to (0,0)
    lcdPrintf(lcdhd, "CPU: %.2fC", CPU_temp); // Display CPU temperature on LCD
    fclose(fp);
}
```

Details about lcdPosition() and lcdPrintf():

lcdPosition (int handle, int x, int y);

Set the position of the cursor for subsequent text entry.

lcdPutchar (int handle, uint8_t data)

lcdPuts (int handle, char *string)

lcdPrintf (int handle, char *message, ...)

These output a single ASCII character, a string or a formatted string using the usual print formatting commands to display individual characters (it is how you are able to see characters on your computer monitor).

Next is subfunction printDateTime() used to display System Time. First, it gets the Standard Time and stores it into variable Rawtime, and then converts it to the Local Time and stores it into timeinfo, and finally displays the Time information on the LCD1602 Display.

```
void printDateTime() { //used to print system time
```

```
time_t rawtime;
struct tm *timeinfo;
time(&rawtime); // get system time
timeinfo = localtime(&rawtime); // convert to local time
printf("%s \n", asctime(timeinfo));
lcdPosition(lcdhd, 0, 1); // set the LCD cursor position to (0,1)
lcdPrintf(lcdhd, "Time:%d:%d:%d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
//Display system time on LCD
}
```

Chapter 2 LCD2004

In the previous chapter, we studied the LCD1602 display. In order to display more content, In this chapter, we will learn about the LCD2004 Display Screen.

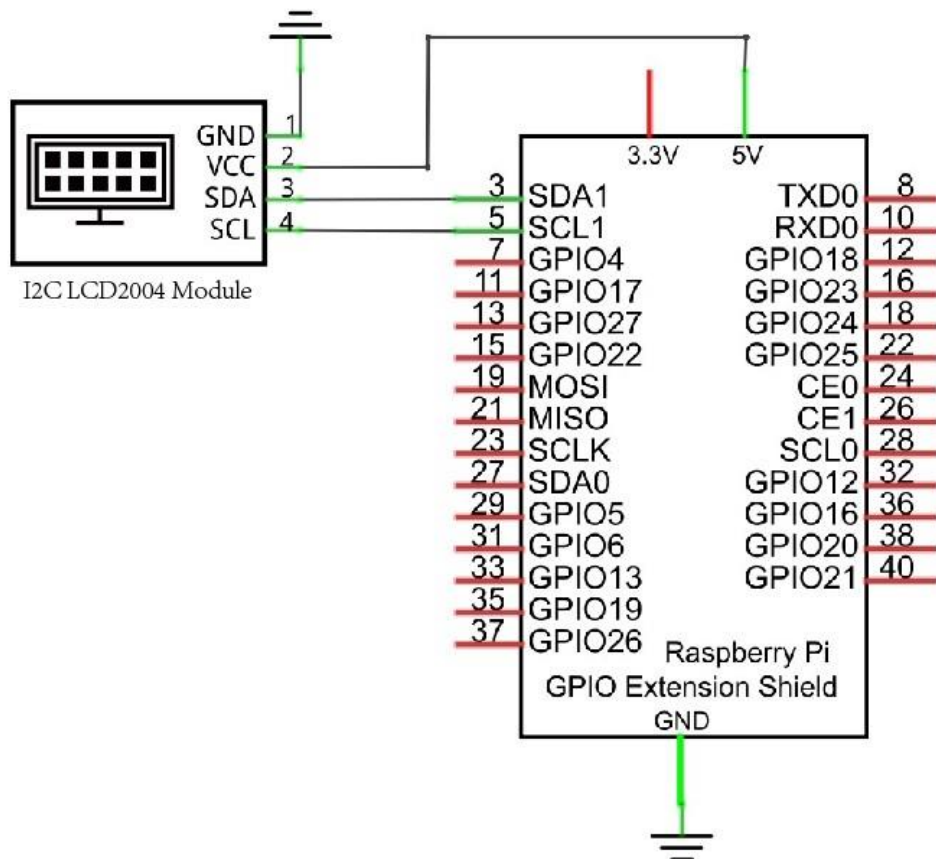
Project 2.1 I2C LCD2004

In this section we learn how to use LCD2004 to display something.

Circuit

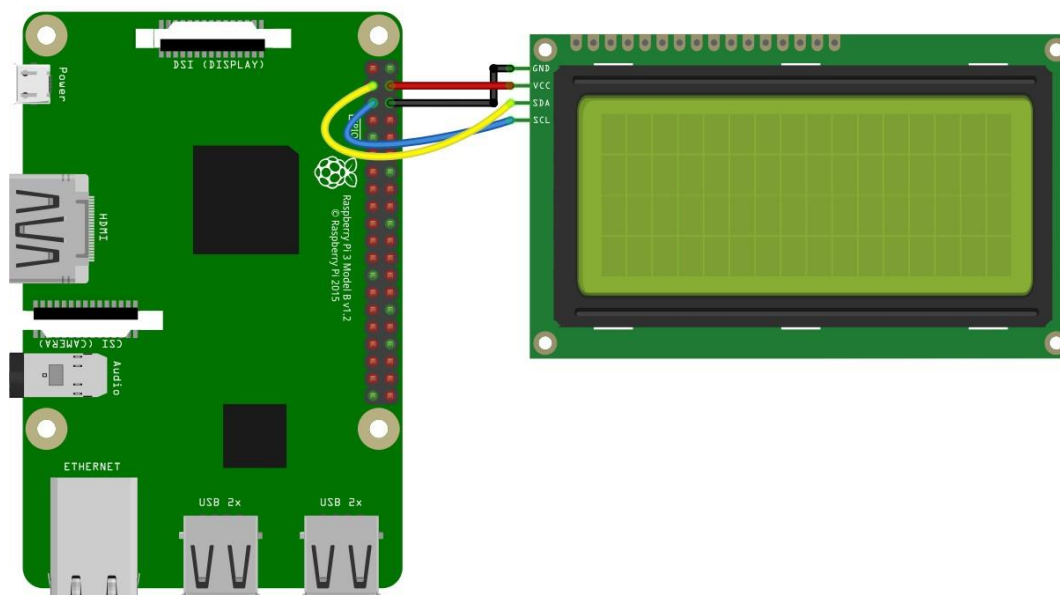
Note that the power supply for I2C LCD2004 in this circuit is 5V.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

NOTE: It is necessary to configure I2C and install Smbus first.



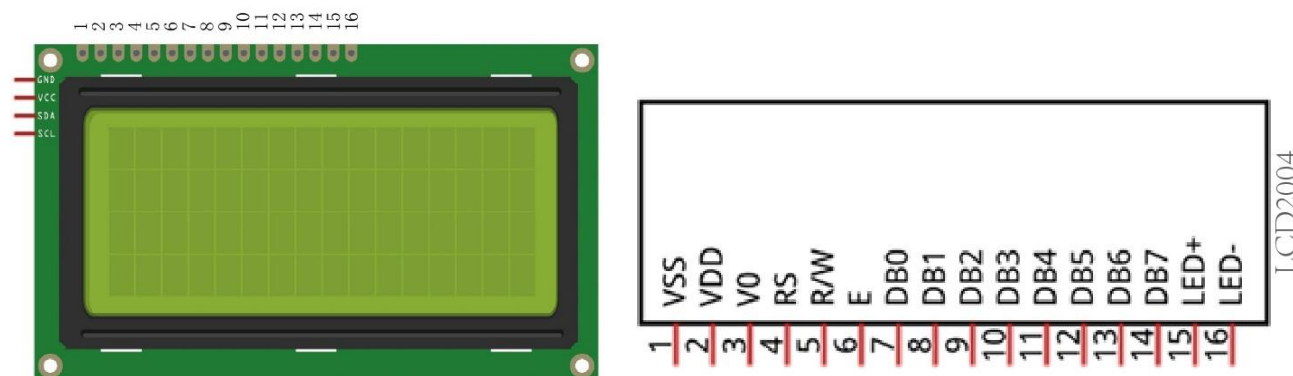
Component knowledge

I2C communication

I2C (Inter-Integrated Circuit) has a two-wire serial communication mode, which can be used to connect a micro-controller and its peripheral equipment. Devices using I2C communications must be connected to the serial data line (SDA), and serial clock line (SCL) (called I2C bus). Each device has a unique address which can be used as a transmitter or receiver to communicate with devices connected via the bus.

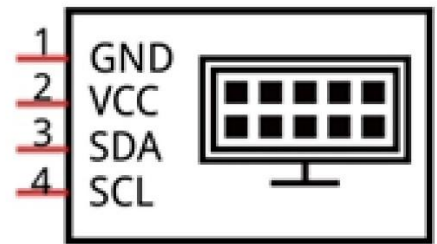
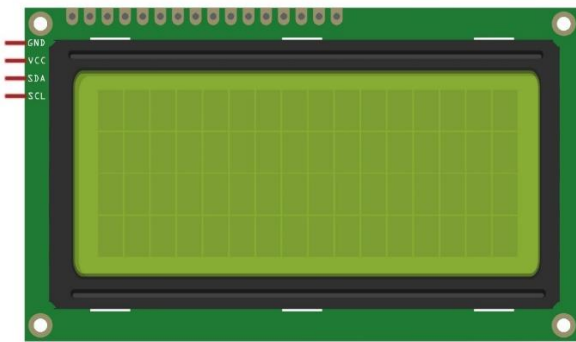
LCD2004 communication

There are LCD2004 display screen and the I2C LCD. We will introduce both of them in this chapter. But what we use in this project is an I2C LCD2004 display screen. The LCD2004 Display Screen can display 4 lines of characters in 20 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD2004 Display Screen along with its circuit pin diagram.



I2C LCD2004 display screen integrates a I2C interface, which connects the serial-input & parallel-output

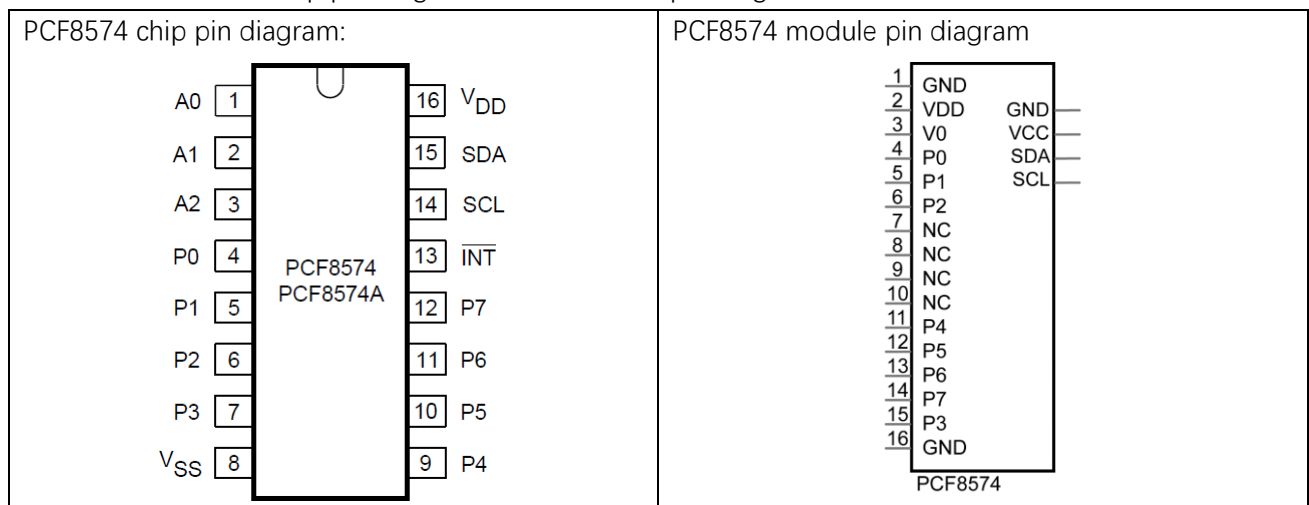
module to the LCD2004 display screen. This allows us to only use 4 lines to operate the LCD2004.



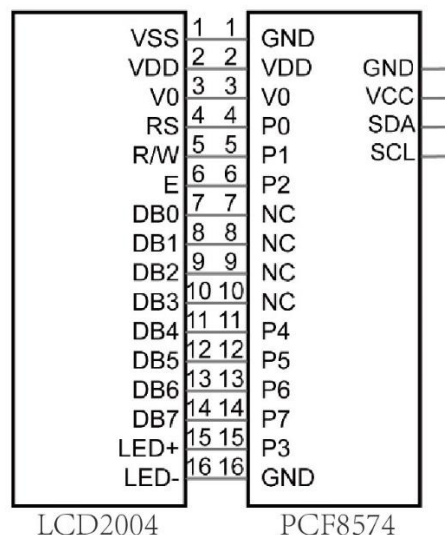
I2C LCD2004 Module

The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F). You can also view the RPI bus on your I2C device address through command "i2cdetect -y 1".

Below is the PCF8574 chip pin diagram and its module pin diagram:



PCF8574 module pins and LCD1602 pins correspond to each other and connected to each other:



Because of this, as stated earlier, we only need 4 pins to control the 16 pins of the LCD2004 Display Screen through the I2C interface.

In this project, we will use the I2C LCD2004 to display some static characters and dynamic variables.

Configure I2C and Install Smbus

If you did not configure I2C and install Smbus, please complete the configuration and installation. If you have, skip this section.

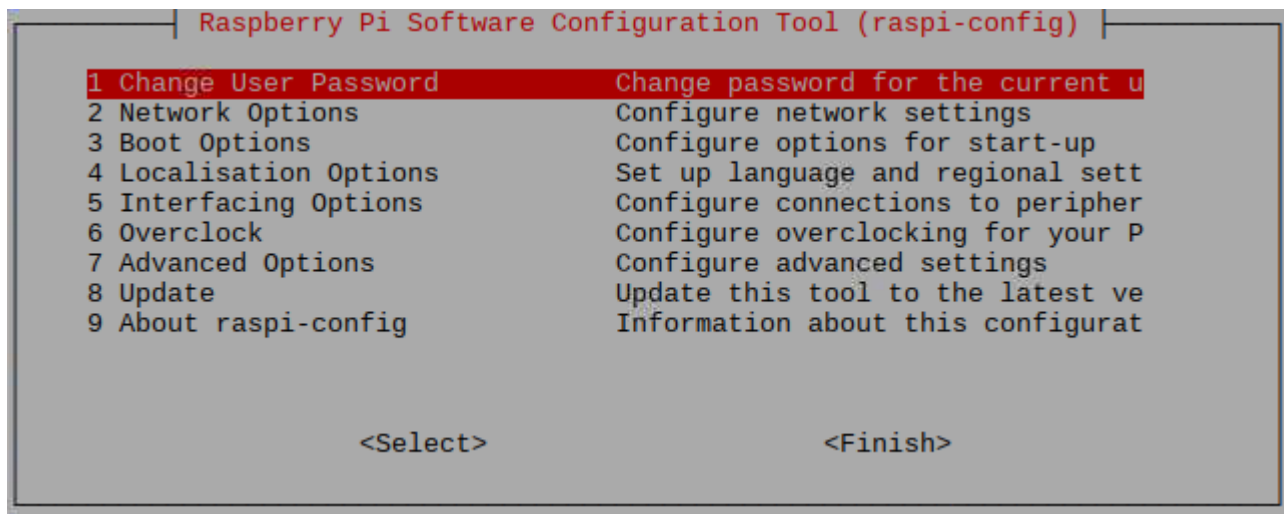
Enable I2C

The I2C interface in Raspberry Pi is disabled by default. You will need to open it manually and enable the I2C interface as follows:

Type command in the Terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose "5 Interfacing Options" then "P5 I2C" then "Yes" and then "Finish" in this order and restart your RPi. The I2C module will then be started.

Type a command to check whether the I2C module is started:

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown. "bcm2708" refers to the CPU model. Different models of Raspberry Pi display different contents depending on the CPU installed:

```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2708      4770  0
i2c_dev         5859  0
pi@raspberrypi:~ $
```

Install I2C-Tools

Next, type the command to install I2C-Tools. It is available with the Raspberry Pi OS by default.

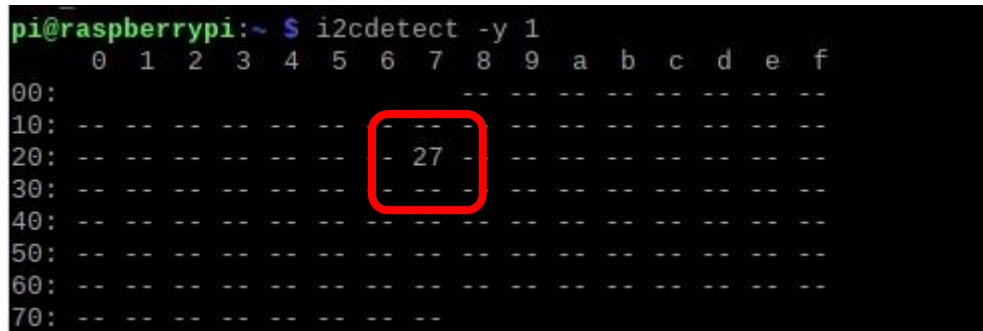
```
sudo apt-get install i2c-tools
```

I2C device address detection:

```
i2cdetect -y 1
```

When you use the serial-parallel IC chip PCF8574T, its I2C default address is 0x27. When the serial-parallel IC chip you use is PCF8574AT, its I2C default address is 0x3F.

When you use the serial-parallel IC chip PCF8574T, the result should look like this:



```
pi@raspberrypi:~$ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Here, 27 (HEX) is the I2C address of LCD2004 Module (PCF8574T).

When you are using PCF8574AT, and its default I2C address is 0x3F.

Install Smbus Module

```
sudo apt-get install python-smbus
```

```
sudo apt-get install python3-smbus
```

Code

This code will have your RPi's CPU temperature and System Time Displayed on the LCD2004.

C Code 2.1 I2CLCD2004

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

If you have any concerns, please contact us via: support@freenove.com

4. Use cd command to enter 1.1_I2CLCD2004 directory of C code.

```
cd ~/Freenove_LCD_Module/Freenove_LCD_Module_for_Raspberry_Pi/C/C_Code/1.1_I2CLCD2004
```

5. Use following command to compile "I2CLCD2004.c" and generate executable file "I2CLCD2004".

```
gcc I2CLCD2004.c -o I2CLCD2004 -lwiringPi -lwiringPiDev
```

6. Then run the generated file "I2CLCD2004".

```
sudo ./I2CLCD2004
```

After the program is executed, the LCD2004 Screen will display your RPi's CPU Temperature and System Time.



NOTE: After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD2004 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.



The following is the program code:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <wiringPi.h>
4  #include <wiringPiI2C.h>
5  #include <pcf8574.h>
6  #include <lcd.h>
7  #include <time.h>
8
9  int pcf8574_address = 0x27;          // PCF8574T:0x27, PCF8574AT:0x3F
10 #define BASE 64                      // BASE any number above 64
11 //Define the output pins of the PCF8574, which are directly connected to the LCD2004 pin.
12 #define RS      BASE+0
13 #define RW      BASE+1
14 #define EN      BASE+2
15 #define LED     BASE+3
16 #define D4      BASE+4
17 #define D5      BASE+5
18 #define D6      BASE+6
19 #define D7      BASE+7
20
21 int lcdhd; // used to handle LCD
22 void printCPUtemperature() { // sub function used to print CPU temperature
23     FILE *fp;
24     char str_temp[15];
25     float CPU_temp;
26     // CPU temperature data is stored in this directory.
27     fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
28     fgets(str_temp, 15, fp);          // read file temp
29     CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
30     printf("CPU's temperature : %.2f \n", CPU_temp);
31     lcdPosition(lcdhd, 0, 2);          // set the LCD cursor position to (0,2)
32     lcdPrintf(lcdhd, "CPU: %.2fC", CPU_temp); // Display CPU temperature on LCD
33     fclose(fp);
34 }
35 void printDateTime() { //used to print system time
36     time_t rawtime;
37     struct tm *timeinfo;
38     time(&rawtime); // get system time
39     timeinfo = localtime(&rawtime); //convert to local time
40     printf("%s \n", asctime(timeinfo));
41     lcdPosition(lcdhd, 0, 3); // set the LCD cursor position to (0,3)
42
43     lcdPrintf(lcdhd, "Time:%02d:%02d:%02d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
```

```

44 //Display system time on LCD
45 }
46 int detectI2C(int addr){ //Used to detect i2c address of LCD
47     int _fd = wiringPiI2CSetup (addr);
48     if (_fd < 0){
49         printf("Error address : 0x%x \n",addr);
50         return 0 ;
51     }
52     else{
53         if(wiringPiI2CWrite(_fd,0) < 0){
54             printf("Not found device in address 0x%x \n",addr);
55             return 0;
56         }
57         else{
58             printf("Found device in address 0x%x \n",addr);
59             return 1 ;
60         }
61     }
62 }
63 int main(void){
64     int i;
65     printf("Program is starting ... \n");
66     wiringPiSetup();
67     if(detectI2C(0x27)) {
68         pcf8574_address = 0x27;
69     }else if(detectI2C(0x3F)) {
70         pcf8574_address = 0x3F;
71     }else{
72         printf("No correct I2C address found, \n"
73             "Please use command 'i2cdetect -y 1' to check the I2C address! \n"
74             "Program Exit. \n");
75         return -1;
76     }
77     pcf8574Setup(BASE,pcf8574_address);//initialize PCF8574
78     for(i=0;i<8;i++){
79         pinMode(BASE+i, OUTPUT); //set PCF8574 port to output mode
80     }
81     digitalWrite(LED, HIGH); //turn on LCD backlight
82     digitalWrite(RW, LOW); //allow writing to LCD
83     lcdhd = lcdInit(4, 20, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return "handle"
84     //used to handle LCD
85     if(lcdhd == -1){
86         printf("lcdInit failed !");
87         return 1;

```

```

87     }
88     lcdPosition(lcdhd,0,0); // set the LCD cursor position to (0,0)
89     lcdPrintf(lcdhd,"FREENOVE");
90     lcdPosition(lcdhd,0,1); // set the LCD cursor position to (0,1)
91     lcdPrintf(lcdhd,"www.freenove.com");
92     while(1) {
93         printCPUTemperature(); //print CPU temperature
94         printDateTime();      // print system time
95         delay(1000);
96     }
97     return 0;
98 }

```

First, define the I2C address of the PCF8574 and the Extension of the GPIO pin, which is connected to the GPIO pin of the LCD2004. LCD2004 has two different i2c addresses. Set 0x27 as default.

```

int pcf8574_address = 0x27; // PCF8574T:0x27, PCF8574AT:0x3F
#define BASE 64 // BASE any number above 64
//Define the output pins of the PCF8574, which are directly connected to the LCD2004 pin.
#define RS BASE+0
#define RW BASE+1
#define EN BASE+2
#define LED BASE+3
#define D4 BASE+4
#define D5 BASE+5
#define D6 BASE+6
#define D7 BASE+7

```

Then, in main function, initialize the PCF8574, set all the pins to output mode, and turn ON the LCD2004 backlight (without the backlight the Display is difficult to read).

```

pcf8574Setup(BASE,pcf8574_address); // initialize PCF8574
for(i=0;i<8;i++) {
    pinMode(BASE+i,OUTPUT); // set PCF8574 port to output mode
}
digitalWrite(LED,HIGH); // turn on LCD backlight

```

Then use lcdInit() to initialize LCD2004 and set the RW pin of LCD2004 to 0 (can be written) according to requirements of this function. The return value of the function called "Handle" is used to handle LCD2004".

```

lcdhd = lcdInit(4, 20, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return
"handle" used to handle LCD

```

Details about lcdInit():

```

int lcdInit (int rows, int cols, int bits, int rs, int strb,
             int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7) ;

```

This is the main initialization function and must be executed first before you use any other LCD functions. **Rows** and **cols** are the rows and columns of the Display (e.g. 2, 16 or 4, 20). **Bits** is the number of how wide the number of bits is on the interface (4 or 8). The **rs** and **strb** represent the pin numbers of the Display's

RS pin and Strobe (E) pin. The parameters **d0** through **d7** are the pin numbers of the 8 data pins connected from the RPi to the display. Only the first 4 are used if you are running the display in 4-bit mode.

The return value is the 'handle' to be used for all subsequent calls to the lcd library when dealing with that LCD, or -1 to indicate a fault (usually incorrect parameter)

For more details about LCD Library, please refer to: <https://projects.drogon.net/raspberry-pi/wiringpi/lcd-library/>

In the next "while", two subfunctions are called to display the RPi's CPU Temperature and the SystemTime. First look at subfunction printCPUtemperature(). The CPU temperature data is stored in the `"/sys/class/thermal/thermal_zone0/temp"` file. We need to read the contents of this file, which converts it to temperature value stored in variable CPU_temp and uses lcdPrintf() to display it on LCD.

```
void printCPUtemperature() { //subfunction used to print CPU temperature

    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
    fgets(str_temp, 15, fp);    // read file temp
    CPU_temp = atof(str_temp)/1000.0;    // convert to Celsius degrees
    printf("CPU's temperature : %.2f \n", CPU_temp);
    lcdPosition(lcdhd, 0, 2);    // set the LCD cursor position to (0,2)
    lcdPrintf(lcdhd, "CPU: %.2fC", CPU_temp); // Display CPU temperature on LCD
    fclose(fp);
}
```

Details about lcdPosition() and lcdPrintf():

lcdPosition (int handle, int x, int y);

Set the position of the cursor for subsequent text entry.

lcdPutchar (int handle, uint8_t data)

lcdPuts (int handle, char *string)

lcdPrintf (int handle, char *message, ...)

These output a single ASCII character, a string or a formatted string using the usual print formatting commands to display individual characters (it is how you are able to see characters on your computer monitor).

Next is subfunction printDateTime() used to display System Time. First, it gets the Standard Time and stores it into variable Rawtime, and then converts it to the Local Time and stores it into timeinfo, and finally displays the Time information on the LCD2004 Display.

```
void printDateTime() { //used to print system time

    time_t rawtime;
    struct tm *timeinfo;
    time(&rawtime); // get system time
    timeinfo = localtime(&rawtime); // convert to local time
    printf("%s \n", asctime(timeinfo));
    lcdPosition(lcdhd, 0, 3); // set the LCD cursor position to (0,3)
    lcdPrintf(lcdhd, "Time: %d: %d: %d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
}
```

```
//Display system time on LCD  
}
```