LAMPIRAN A

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertanda tangan di bawah ini,

Pihak Pertama

Nama	: Ajie Harun Pratama
NIM	: 062030320980
Jurusan	: Teknik Elektro
Program Studi	: D3 Teknik Elektronika

Pihak Kedua

Nama	: Dewi Permata Sari, S. T, M. Kom.	
NIP	: 197612132000032001	
Jurusan	: Teknik Elektro	
Program Studi	: D3 Teknik Elektronika	

Pada hari ini ..15 Januari... tanggal ...15 Januari [illegible]... telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.

Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari ...[illegible]... pukul ...[illegible]..., tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, ...

Pihak Pertama,

Ajie Harun Pratama
NIM 062030320980

Pihak Kedua,

Dewi Permata Sari, S. T, M. Kom.
NIP 197612132000032001

Mengetahui,
Ketua Jurusan

Ir. Iskandar Lutfi, M. T.
NIP 196501291991031002

No. Dok. : F-PBM-16 | Tgl. Berlaku : 13 Desember 2010 | No. Rev. : 00

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id **E-mail** : info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertanda tangan di bawah ini,

Pihak Pertama

Nama	: Ajie Harun Pratama
NIM	: 062030320980
Jurusan	: Teknik Elektro
Program Studi	: D3 Teknik Elektronika

Pihak Kedua

Nama	: Dr. Nyayu Latifah Husni, S. T, M. T	
NIP	: 197605032001122002	
Jurusan	: Teknik Elektro	
Program Studi	: D3 Teknik Elektronika	

Pada hari ini .... tanggal .... telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.

Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari ...., pukul ...., tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, ...

Pihak Pertama,

Ajie Harun Pratama
NIM 062030320980

Pihak Kedua,

Dr. Nyayu Latifah Husni, S. T, M. T
NIP 197605032001122002

Mengetahui,
Ketua Jurusan

Ir. Iskandar Lutfi, M. T.
NIP 196501291991031002

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polsriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Lembar : 1

Nama	:	Ajie Harun Pratama
NIM	:	062030320980
Jurusan/Program Studi	:	Teknik Elektro / D3 Teknik Elektronika
Judul Laporan Akhir	:	Rancang bangun social aplikasi robot volunteer 02 untuk komunitas anak penderita kanker menggunakan Raspberry Pi
Pembimbing I / II *)	:	Dewi Permata Sari, S. T, M. Kom.

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	0 Feb 23	ACC proposal laporan akhir.	
2.	3 Maret 23	Perbaikan latar belakang dan rumusan	
3.	12 April 23	Perbaiki tujuan, manfaat dan batasan masalah	
4.	15 April 23	ACC BAB I lanjut BAB II	
5.	3 Mei 23	Tambahkan tinjauan pustaka Bab II	
6.	27 Mei 23	Tambahkan jurnal atau jurnal terpublikasi max 3 orang	
7.	13 Juni 23	Tambahkan poin pada flowchart dan pecah per...	

No. Dok. : F-PBM-17 Tgl. Berlaku : 13 Desember 2010 No. Rev. : 00

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Lembar : 1

Nama : Ajie Harun Pratama

NIM : 062030320980

Jurusan/Program Studi : Teknik Elektro / D3 Teknik Elektronika

Judul Laporan Akhir : RANCANG BANGUN SOCIAL ASSISTIVE ROBOT [illegible] G2 UNTUK KOMUNITAS ANAK PENDERITA KANKER MENGGUNAKAN RASPBERRY PI

Pembimbing I / II *) : Dr. Nyayu Latifah Husni, S. T, M. T

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	9 Feb 23	ACC [illegible] LA	
2.	2 Mar 23	menyempurnakan latar belakang dan batasan masalah	
3.	10 April 23	perumusan, Tujuan, manfaat pada Laporan Akhir	
4.	15 April 23	Sempurnakan BAB I, dan lanjut Bab 2.	
5.	2 Nov 23	Tambahkan Tinjauan Pustaka Servo MAX30100	
6.	5 Nov 23	Lengkapi Tinjauan Sistem dari input - output - [illegible]	
7.	23 Nov 23	Tambahkan [illegible] pada gambar dan [illegible]	

No. Dok. : F-PBM-17 Tgl. Berlaku : 13 Desember 2010 No. Rev. : 00

Lembar : 2

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	20 Juni 23	Jelaskan Input/Output/Parameter SARVG2 dan Blok Diagram	
9.	16 Juni 23	Perbaikan Penulisan BAB III dan Penambahan data	
10.	30 Juni 23	Mengambil data trajektori SARVG 2.	
11.	2 Agust 22	Perbaikan Bab IV dan V	
12.	7 Agustus 23	Penyempurnaan seluruh LA	

Palembang, ...

Ketua Jurusan/KPS,

(Dewi Permata Sari , S. T. M. Kom.)
NIP 197612132000032001

Catatan:
*) melingkari angka yang sesuai.
Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Laporan Akhir sebelum menandatangani lembar bimbingan ini.
Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir.

No. Dok. : F-PBM-17 Tgl. Berlaku : 13 Desember 2010 No. Rev. : 00

Lembar : 2

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	7 Juni 23	Jelaskan flowchart (buat klasifikasi) dan layar saran	
9.	20 Juni 23	ACC Bab 3. dan rumuskan data SAW GL.	
10.	5 Juli 23	Tambahkan data [illegible] (Bpm, SUP), Trayonom, Tagongan [illegible]	
11.	19 Juli 23	ACC bab 4, dan layar Bab 5.	
12.	2 Agustus 23	ACC Bab 5. lanjut [illegible] sidang LA	

Palembang, ...

Ketua Jurusan/KPS,

(Dewi Permata Sari , S. T, M. Kom.)
NIP 197612132000032001

Catatan:
*) melingkari angka yang sesuai.
Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Laporan Akhir sebelum menandatangani lembar bimbingan ini.
Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id **E-mail** : info@polsri.ac.id

PELAKSANAAN REVISI LAPORAN AKHIR

Mahasiswa berikut,

Nama	:	Ajie Harun Pratama
NIM	:	062030320980
Jurusan/Program Studi	:	Teknik Elektro / D3 Teknik Elektronika
Judul Laporan Akhir	:	RANCANG BANGUN SOCIAL ASSISTIVE ROBOT VOLUNTEER G2 UNTUK KOMUNITAS ANAK PENDERITA KANKER MENGGUNAKAN RASPBERRY PI

Telah melaksanakan revisi terhadap Laporan Akhir yang diujikan pada hari senin tanggal 7 bulan Agustus tahun 2023 Pelaksanaan revisi terhadap Laporan Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No.	Komentar	Nama Dosen Penguji *)	Tanggal	Tanda Tangan
	Acc	Ir. Yordan Hasan, M. Kom		
		Dr. Eng Tresna Dewi S. T, M. Eng.		
		Dewi Permata Sari, S. T, M. Kom.		
	Acc to revised.	Destra Andika Pratama, S. T, M. T.	3/8/23	
	Revised	Renny Maulidda, S. T. M. T.	3/8'23	

Palembang, ...

Ketua Penguji **),

(...)
NIP ....

Catatan:
*) Dosen penguji yang memberikan revisi saat ujian laporan akhir.
**) Dosen penguji yang ditugaskan sebagai Ketua Penguji saat ujian LA.
Lembaran pelaksanaan revisi ini harus dilampirkan dalam Laporan Akhir

LAMPIRAN B

DOKUMENTASI

=

LAMPIRAN C

1.54inch OLED SSD1309 SPI&IIC Module
MSP154W&MSP154B
User Manual

Introduction to OLED

OLED is an Organic Light-Emitting Diode (OLED). OLED display technology has the advantages of self-illumination, wide viewing angle, almost infinite contrast, low power consumption, high reaction speed, flexible panel, wide temperature range, simple structure and process, etc. A generation of flat panel display emerging application technology.

OLED display is different from traditional LCD display, it can self-illuminate, so no backlight is needed, which makes OLED display

The display is thinner than the LCD display and has a better display.

Product Description

The OLED module has a display size of 1.54 inches and has a 128x64 resolution. Three-wire system, 4-wire SPI and IIC communication modes can be selected, and the driver IC is SSD1309. Contains three modules in black, blue or yellow and blue.

Product Features

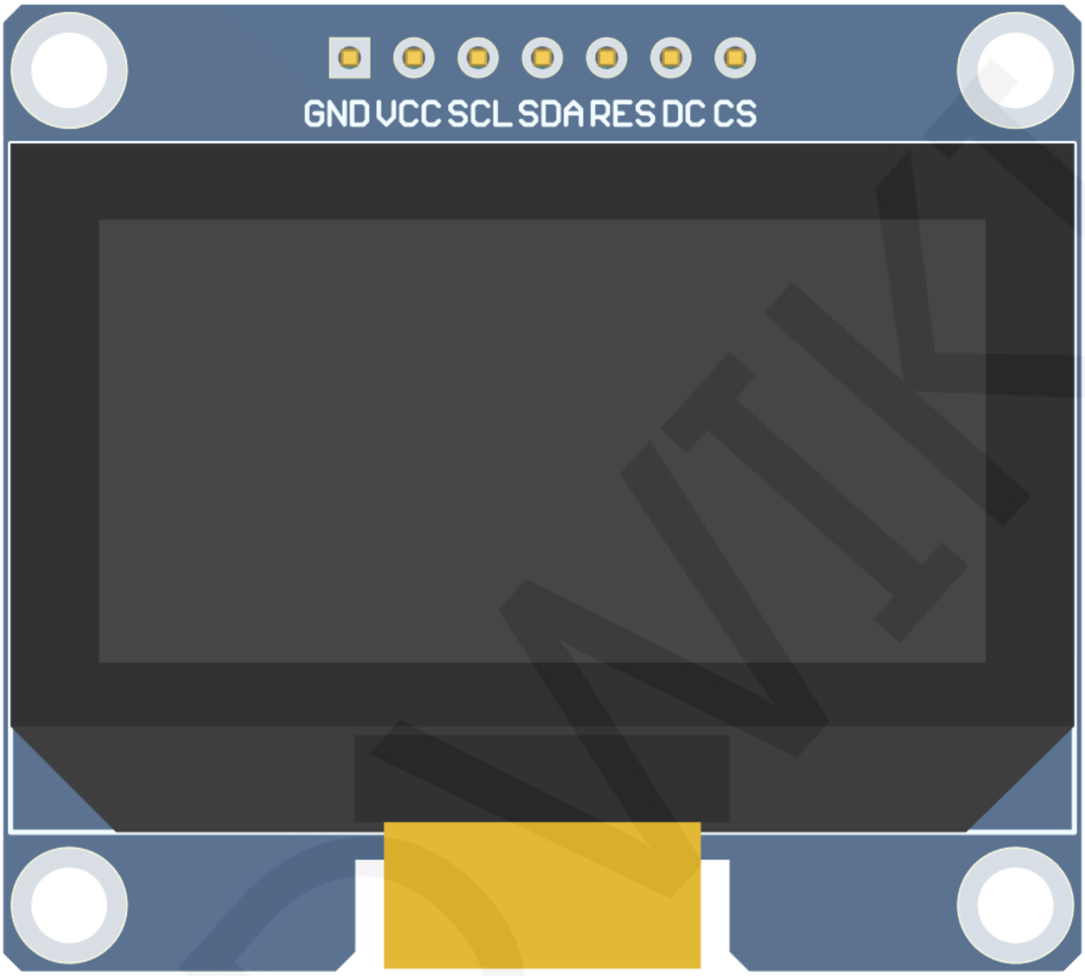
- 1.54 inch OLED screen with black and white, black or blue color display
- 128x64 resolution for clear display and high contrast
- Large viewing angle: greater than 160° (one screen with the largest viewing angle in the display)
- Wide voltage supply (3V~5V), compatible with 3.3V and 5V logic levels, no level shifting chip required
- The default is 4-wire SPI bus, which can choose IIC bus
- Ultra-low power consumption: normal display is only 0.06W (far below the TFT display)
- Military-grade process standards, long-term stable work

- Provides a rich sample program for STM32, C51, Arduino, Raspberry Pi and MSP430 platforms
- Provide underlying driver technical support

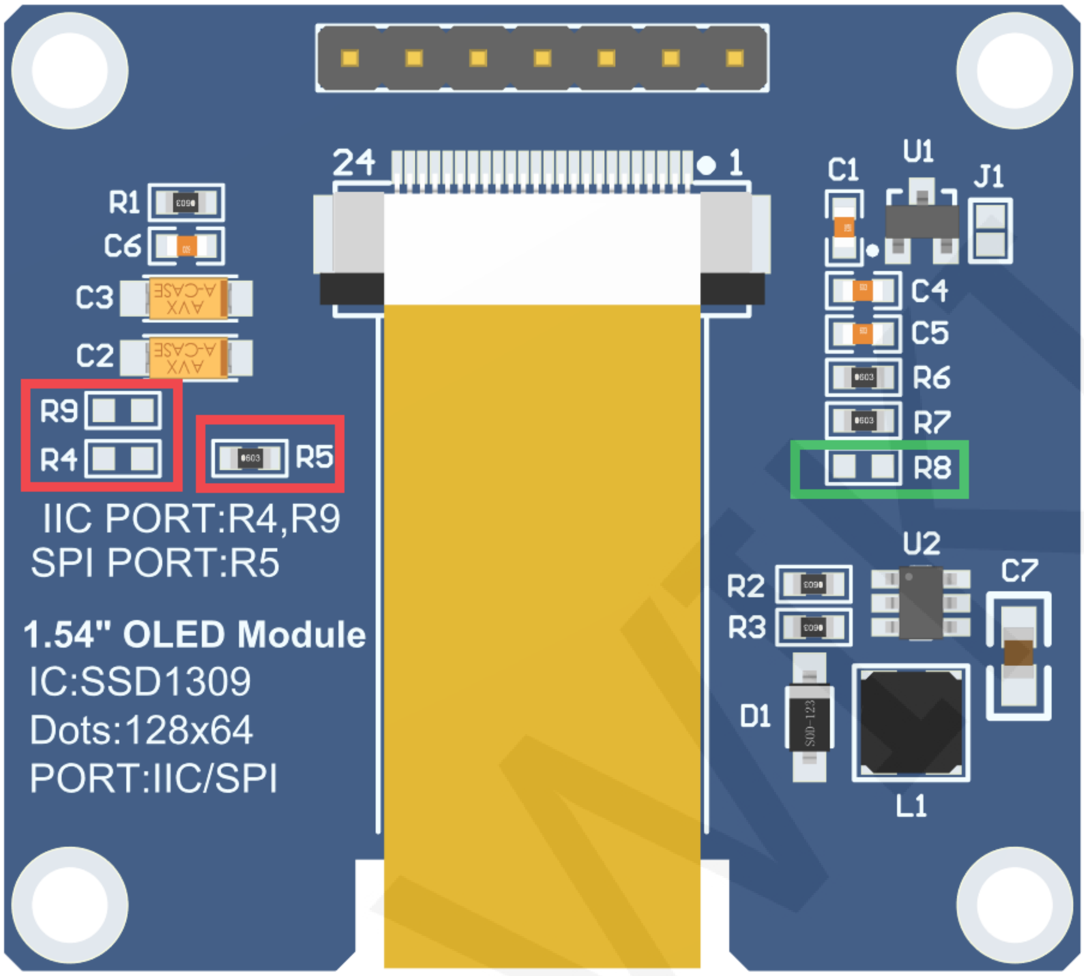
Product Parameters

Name	Description
Display Color	White, blue
SKU	MSP154W MSP154B
Screen Size	1.54(inch)
Type	OLED
Driver IC	SSD1309
Resolution	128*64(Pixel)
Module Interface	4-line SPI、IIC interface
Active Area	35.052x17.516(mm)
Touch Screen Type	have no touch screen
Touch IC	have no touch IC
Module PCB Size	42.40x38.00(mm)
Angle of view	>160°
Operating Temperature	-20℃~60℃
Storage Temperature	-30℃~70℃
Operating Voltage	3.3V/5V
Power Consumption	TBD
Product Weight(With packaging)	12(g)

Interface Description



Picture1. Module pin Label picture



Picture 2. Rear view of the module

NOTE:

1. This module supports IIC and 4-wire SPI interface bus mode switching (as shown in the red box in Picture 2). The details are as follows:
   A. Using 4.7K resistance to solder only R5 resistors, then choose 4-wire SPI bus interface (default);
   B. Using 4.7K resistance to solder only R4 and R9 resistors, then select the IIC bus interface;
2. After the interface bus mode is switched, you need to select the corresponding software and the corresponding wiring pins (as shown in Picture 1) for the module to operate normally. The corresponding wiring pins are described as follows:

A. select the 4-wire SPI bus interface, all pins need to be used;

B. To select the IIC bus interface, only five pins GND, VCC, SCL, SDA and res need to be used. Connect the CS pin to the power supply and ground. The DC pin can be used to select the IIC slave device address. Select 0x7a for high level and 0x78 for low level;

3. Solder R8 resistor (as shown in the green box in Picture2), then CS pin does not need to be connected

important:

1. The following pin numbers 1~7 refer to the module pin number of our company with PCB backplane. If you purchase a bare screen, please refer to the pin definition of the bare screen specification, refer to the wiring according to the signal type instead of directly according to the following. The module pin number is used for wiring. For example: CS is 7 feet on our module. It may be x pin on different size bare screen. The following wiring instructions tell you that the CS signal is connected to the A5 pin of the MCU.

2. About VCC supply voltage: The OLED display module can be connected to 3.3V or 5V.

Number	Module Pin	Pin Description
1	GND	OLED power ground
2	VCC	OLED power positive (3.3V~5V)
3	SCL	OLED SPI and IIC bus clock signals
4	SDA	OLED SPI and IIC bus data signals
5	RES	OLED reset signal, low level reset (this pin need to connected to the high level (can be connected to the VCC) when selecting IIC bus)
6	DC	Select SPI bus as command / data input selection signal, high level: data, low level: command; When selecting the IIC bus, this pin can be used to select the address of the IIC slave device,

		which is connected to the high level selection 0x7a and the low level selection 0x78;
7	**CS**	OLED chip selection signal, low level enabled; Solder R8 resistor (as shown in the green box in Picture 2), then this pin does not need to be connected. Do not weld R8 resistor, select IIC interface, then this pin is grounded

Hardware Configuration

The hardware circuit of the module is composed of four parts: OLED display control circuit, OLED boost circuit, pin array interface, and power supply voltage stabilizing circuit.

OLED display control circuit is mainly used to control OLED display, including chip selection, reset, data and command transmission control, and interface selection.

The OLED boosting circuit is used to boost an input voltage to an OLED light emitting voltage.

The pin array interface is used for external connection of the main control development board.

The power supply voltage stabilizing circuit is used for 3.3V voltage stabilizing power supply.The OLED module adopts 4-wire SPI communication mode by default. In addition, it can also select IIC communication mode. The hardware is configured with 7 pins. Different communication methods are used, and the selected pins are different (see the interface description for details).

working principle

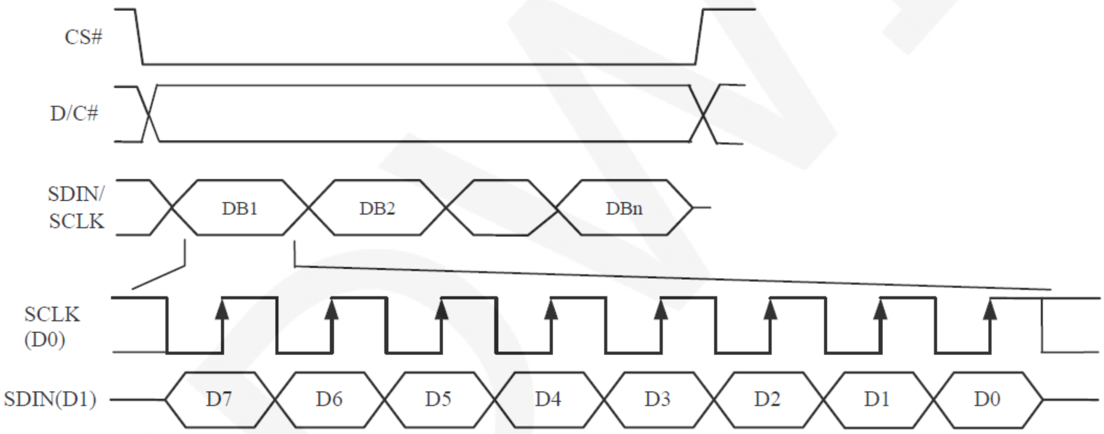
1. Introduction to SSD1309 Controller

The SSD1309 is an OLED/PLED controller that supports a maximum resolution of 128*64 and a 1024-byte GRAM. Support 8-bit 6800 and 8-bit 8080 parallel port data bus, also supports 3-wire and 4-wire SPI serial bus and I2C bus. Since parallel control requires a large number of IO ports, the most commonly used are the SPI serial bus and the I2C bus. It supports vertical scrolling and can be used in small portable devices such as mobile phones, MP3 players and more.

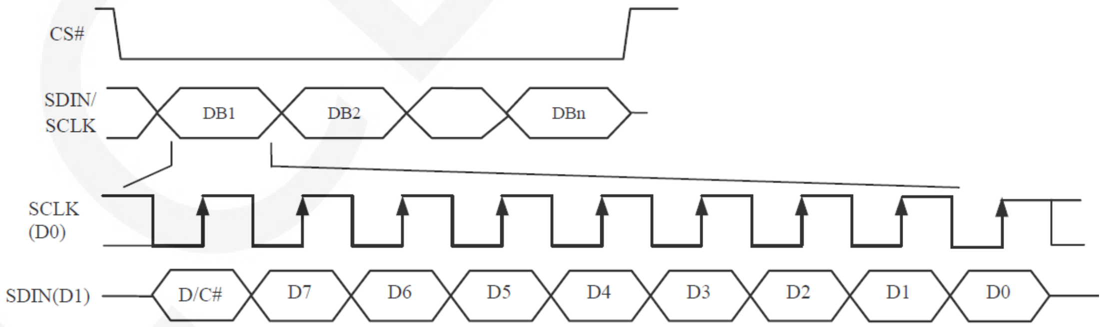
The SSD1309 controller uses 1 bit to control a pixel display, so each pixel can only display black and white or black and blue. The displayed RAM is divided into 8 pages, with 8 lines per page and 128 pixels per line. When setting pixel data, you need to specify the page address first, and then specify the column low address and column height address respectively, so set 8 pixels in the vertical direction at the same time. In order to be able to flexibly control the pixel points at any position, the software first sets a global one-dimensional array of the same size as the display RAM, first maps the pixel point data to the global array, and the process uses the OR or the operation to ensure that the global array is written before. The data is not corrupted, and the data of the global array is then written to the GRAM so that it can be displayed through the OLED.

2. Introduction to SPI communication protocol

The 4-wire SPI bus write mode timing is shown in the following figure:



The 3-wire SPI bus write mode timing is shown in the following figure:



As can be seen from the above timing diagram, the difference between the 3-wire SPI and the 4-wire SPI is as follows:

The 3-wire SPI does not have a D/C# signal, and its D/C# signal is input by SDIN, which first transmits 1 bit of D/C# data, followed by an 8-bit command or data. The 4-wire D/C# signal is directly input by D/C#.

CS# is a slave chip select, and the chip is enabled only when CSX is low.

D/C# is the data/command control pin of the chip. When DCX is low, the command is written. When it is high, the data is written.

SCLK is the SPI bus clock, and each rising edge transmits 1 bit of data;

SDIN is the data transmitted by SPI, and it transmits 8-bit data at a time.The high position is in front and transmitted first.For SPI communication, the data has a transmission timing, that is, a combination of clock phase (CPHA) and clock polarity (CPOL):

The CPOL level determines the idle state level of the serial synchronous clock, CPOL = 0, which is low. CPOL does not have a lot of impact on the transport protocol;

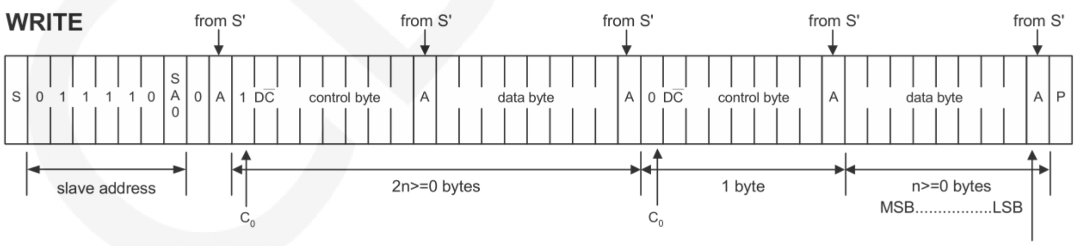
The level of CPHA determines whether the serial synchronous clock is acquired on the first clock transition edge or the second clock transition edge.

When CPHL = 0, data acquisition is performed on the first edge of the transition;

The combination of the two becomes the four SPI communication methods. SPI0 is usually used in China, that is, CPHL = 0, CPOL = 0.

3. Introduction to IIC Communication Protocol

The process of writing data on the IIC bus is shown in the following figure:



After the IIC bus starts working, the slave device address is sent first. After receiving the slave device response, it then sends a control byte to inform the slave device whether the next data to be sent is a command written to the IC register or written. The RAM data, after receiving the slave device response, then sends a value of multiple bytes until the transmission is completed and the IIC bus stops working.

among them:

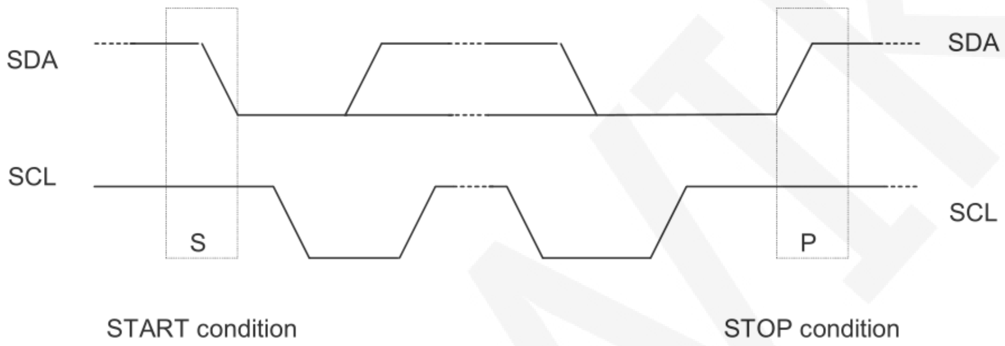
C0=0: This is the last control byte, and all the data bytes sent in the following are all data bytes.

C0=1: The next two bytes to be sent are the data byte and another control byte.

D/C(——)=0: is the register command operation byte

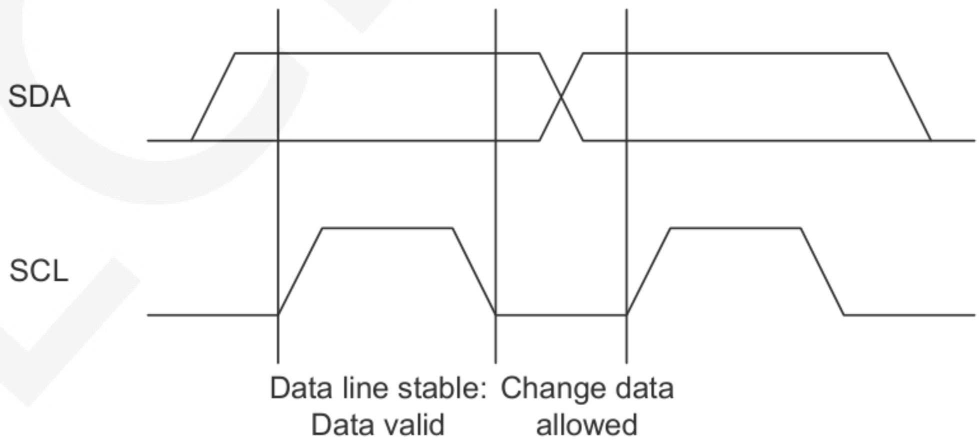
D/C(——)=1: operation byte for RAM data

The IIC start and stop timing diagrams are as follows:



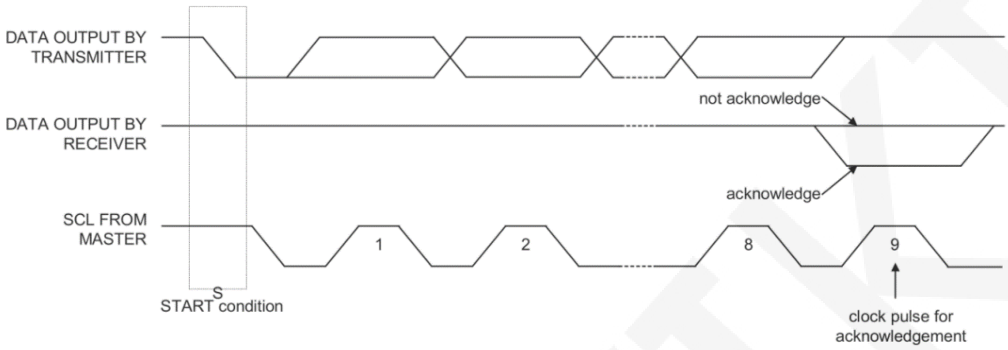
When the data line and the clock line of the IIC are both kept at a high level, the IIC is in an idle state. At this time, the data line changes from a high level to a low level, and the clock line continues to be at a high level, and the IIC bus starts data transmission. When the clock line is held high, the data line changes from low to high, and the IIC bus stops data transmission.

The timing diagram for the IIC to send a bit of data is as follows:



Each clock pulse (the process of pulling high and pulling low) sends 1 bit of data. When the clock line is high, the data line must remain stable, and the data line is allowed to change when the clock line is low.

The ACK transmission timing diagram is as follows:



When the master waits for the ACK of the slave, it needs to keep the clock line high. When the slave sends an ACK, keep the data line low.

Instructions for use

1. Arduino instructions

Wiring instructions:

See the interface description for pin assignments.

Arduino UNO microcontroller test program wiring instructions			
Number	**Module Pin**	**Corresponding to UNO development board wiring pins**	
		SPI	**IIC**
1	**GND**	GND	
2	**VCC**	5V/3.3V	
3	**SCL**	13	A5
4	**SDA**	11	A4
5	**RES**	3.3V	
6	**DC**	9	VCC or GND
7	**CS**	10	GND

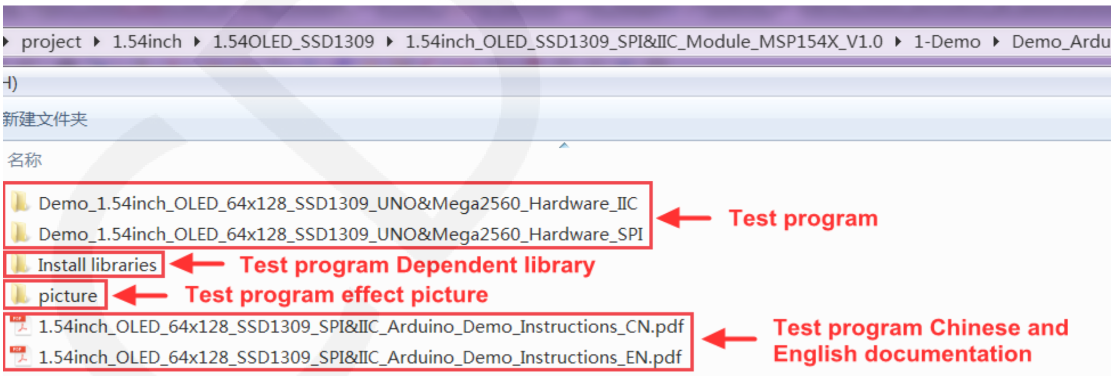
Arduino MEGA2560 microcontroller test program wiring instructions			
Number	**Module Pin**	**Corresponding to MEGA2560 development board wiring pins**	
		SPI	**IIC**
1	**GND**	GND	
2	**VCC**	5V/3.3V	
3	**SCL**	53	21
4	**SDA**	51	20
5	**RES**	3.3V	
6	**DC**	9	VCC or GND
7	**CS**	10	GND

Operating Steps:

A. Connect the OLED module and the Arduino MCU according to the above wiring instructions, and power on;

B. Select the example you want to test, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected sample project, compile and download.

The specific operation methods for the Arduino test program relying on library copy, compile and download are as follows:

http://www.lcdwiki.com/res/PublicFile/Arduino_IDE_Use_Illustration_EN.pdf

D. If the OLED module displays characters and graphics normally, the program runs Successfully;

2. RaspberryPi instructions

Wiring instructions:

See the interface description for pin assignments.

NOTE:

Physical pin refers to the GPIO pin code of the RaspBerry Pi development board.

BCM encoding refers to the GPIO pin coding when using the BCM2835 GPIO library.

WiringPi coding refers to the GPIO pin coding when using the wiringPi GPIO library.

Which GPIO library is used in the code, the pin definition needs to use the corresponding GPIO library code, see Picture 1 GPIO map table for details.

wiringPi编码	BCM编码	功能名	物理引脚BOARD编码		功能名	BCM编码	wiringPi编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

Picture4. GPIO map

Raspberry Pi test program wiring instructions			
Number	Module Pin	Corresponding to development board wiring pin	
		SPI	IIC
1	GND	GND (Physical pin: 6,9,14,20,25,30,34,39)	
2	VCC	5V/3.3V (Physical pin: 1,2,4)	
3	SCL	Physical pin: 23 BCM coding: 11 wiringPi coding: 14	Physical pin: 5 BCM coding: 3 wiringPi coding: 9
4	SDA	Physical pin: 19 BCM coding: 10 wiringPi coding: 12	Physical pin: 3 BCM coding: 2 wiringPi coding: 8
5	RES	Physical pin: 5 BCM coding: 3 wiringPi coding: 9	Physical pin: 23 BCM coding: 11 wiringPi coding: 14
6	DC	Physical pin: 3 BCM coding: 2 wiringPi coding: 8	VCC or GND
7	CS	Physical pin: 24 BCM coding: 8 wiringPi coding: 10	GND

Operating Steps:

A. open the SPI function of RaspberryPi

Log in to the RaspberryPi using a serial terminal tool (such as putty) and enter the following command:

sudo raspi-config

Select Interfacing Options->SPI->YES

Start RaspberryPi's SPI kernel driver

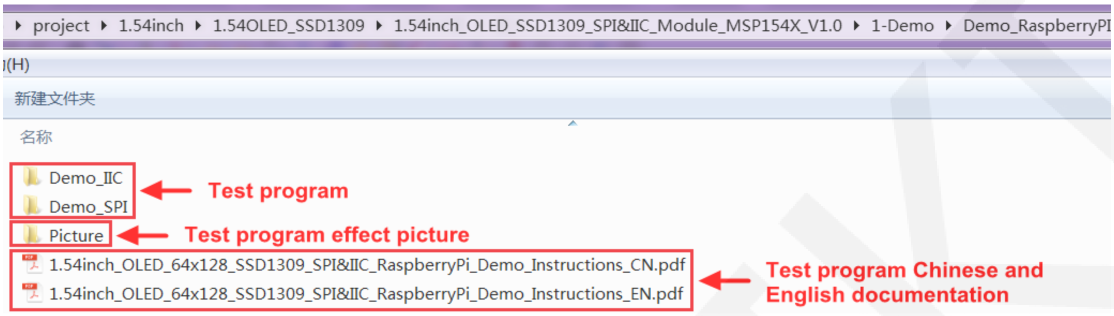
B. install the function library

For detailed installation methods of the bcm2835, wiringPi, and python function libraries of RaspberryPi, see the following documents:

http://www.lcdwiki.com/res/PublicFile/Raspberrypi_Use_Illustration_EN.pdf

C. select the example that needs to be tested, as shown below:

(Please refer to the test program description document for test program description)



D. bcm2835 instructions(Take the 4-wire hardware SPI test program as an example)

a) Connect the OLED module to the RaspberryPi development board according to the above wiring

b) Copy the test program directory

Demo_1.54inch_OLED_64x128_SSD1309_bcm2835_Hardware_4-wire_SPI

to RaspberryPi (can be copied via SD card or via FTP tool (such as FileZilla))

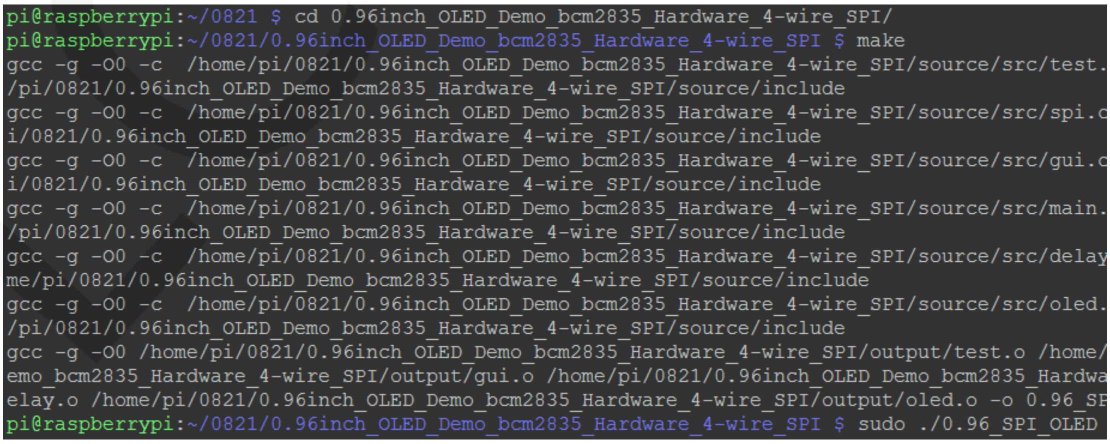
c) Run the following command to run the bcm2835 test program:

cd Demo_1.54inch_OLED_64x128_SSD1309_bcm2835_Hardware_4-wire_SPI

make

sudo ./ 1.54_SPI_OLED

As shown below:



E. wiringPi instructions(Take the 4-wire hardware SPI test program as an example)

a) Connect the OLED module to the RaspberryPi development board according to the above wiring

b) Copy the test program directory

Demo_1.54inch_OLED_64x128_SSD1309_wiringPi_Hardware_4-wire_SPI

to RaspberryPi (can be copied via SD card or via FTP tool (such as FileZilla))

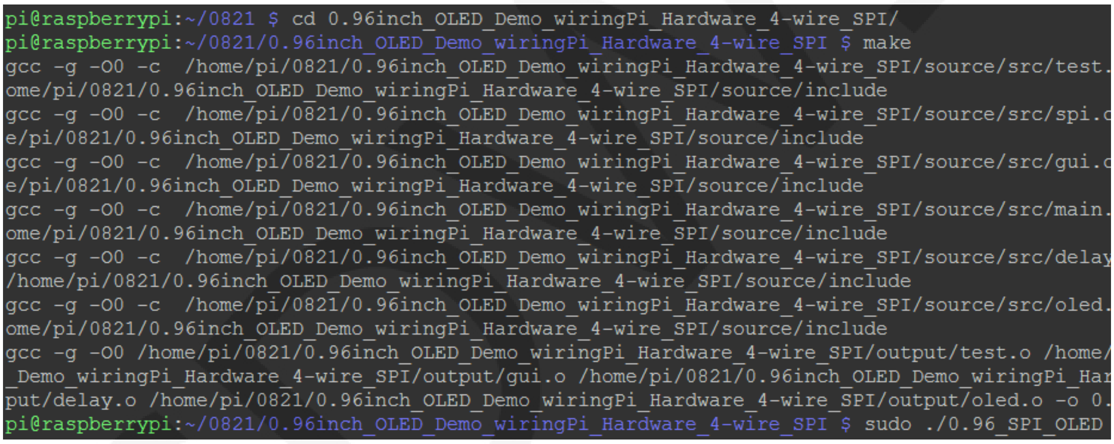
c) Run the following command to run the wiringPi test program:

cd Demo_1.54inch_OLED_64x128_SSD1309_wiringPi_Hardware_4-wire_SPI

make

sudo ./ 1.54_SPI_OLED

As shown below:



```
pi@raspberrypi:~/0821 $ cd 0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI $ make
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/test.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/spi.c
e/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/gui.c
e/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/main.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/delay
/home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/oled.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/output/test.o /home/
_Demo_wiringPi_Hardware_4-wire_SPI/output/gui.o /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Har
put/delay.o /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/output/oled.o -o 0.
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI $ sudo ./0.96_SPI_OLED
```

F. python instructions(Take the 4-wire hardware SPI test program as an example)

a) The image processing library PIL needs to be installed before running the python test program. The specific installation method is as follows:

http://www.lcdwiki.com/res/PublicFile/Python_Image_Library_Install_Illustration_EN.pdf

b) Connect the OLED module to the RaspberryPi development board as described above.

c) Copy the test program directory

Demo_1.54inch_OLED_64x128_SSD1309_python_Hardware_4-wire_SPI

to RaspberryPi (either via SD card or via FTP tool (such as FileZilla))

d) Run the following command to run 3 python test programs separately:

cd Demo_1.54inch_OLED_64x128_SSD1309_python_Hardware_4-wire_SPI/source

sudo python show_graph.py

sudo python show_char.py

sudo python show_bmp.py

As shown below:

```
pi@raspberrypi:~/0821 $ cd 0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source/
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_graph.py
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_char.py
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_bmp.py
```

3. STM32 instructions

Wiring instructions:

See the interface description for pin assignments.

STM32F103C8T6 microcontroller test program wiring instructions			
Number	**Module Pin**	**Corresponding to STM32F103C8T6 development board wiring pin**	
		SPI	**IIC**
1	**GND**	GND	
2	**VCC**	3.3V/5V	
3	**SCL**	PA5	
4	**SDA**	PA7	
5	**RES**	PB8	
6	**DC**	PB7	VCC or GND
7	**CS**	PB9	GND

STM32F103RCT6 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to MiniSTM32 development board wiring pin**

		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PB13	
4	SDA	PB15	
5	RES	PB12	
6	DC	PB10	VCC or GND
7	CS	PB11	GND

STM32F103ZET6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to Elite STM32 development board wiring pin	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PB13	
4	SDA	PB15	
5	RES	PB12	
6	DC	PB10	VCC or GND
7	CS	PB11	GND

STM32F407ZGT6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to Explorer STM32F4 development board wiring pin	
		SPI	IIC
1	GND	GND	

2	VCC	3.3V/5V	
3	SCL	PB3	
4	SDA	PB5	
5	RES	PB12	
6	DC	PB14	VCC or GND
7	CS	PB15	GND

STM32F429IGT6 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PF7	
4	SDA	PF9	
5	RES	PD12	
6	DC	PD5	VCC or GND
7	CS	PD11	GND

STM32F767IGT6 and STM32H743IIT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin
		SPI
1	GND	GND
2	VCC	3.3V/5V
3	SCL	PB13
4	SDA	PB15

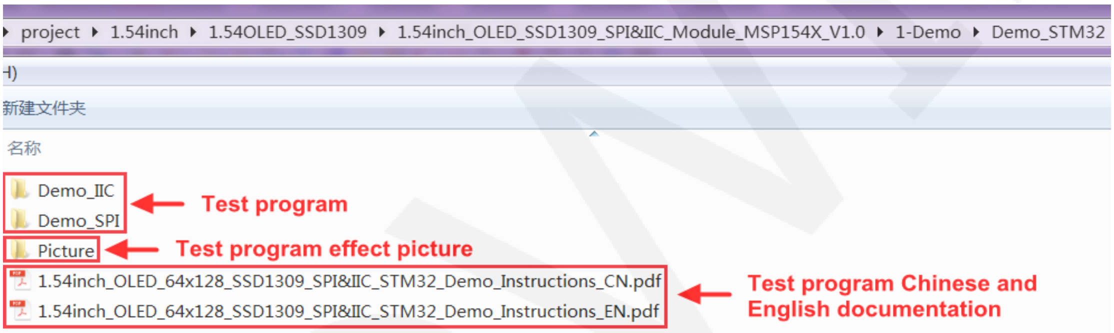
5	**RES**	PD12
6	**DC**	PD5
7	**CS**	PD11

Operating Steps：

A. Connect the IPS module and the STM32 MCU according to the above wiring instructions, and power on;

B. Select the test example according to the model of the microcontroller, as shown in the following figure:

(Please refer to the test program description document in the test package for the test program description)



C. Open the selected test program project, compile and download;

detailed description of the STM32 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_EN.pdf

D. If the OLED module displays characters and graphics normally, the program runs successfully；

4. C51 instructions

Wiring instructions:

See the interface description for pin assignments.

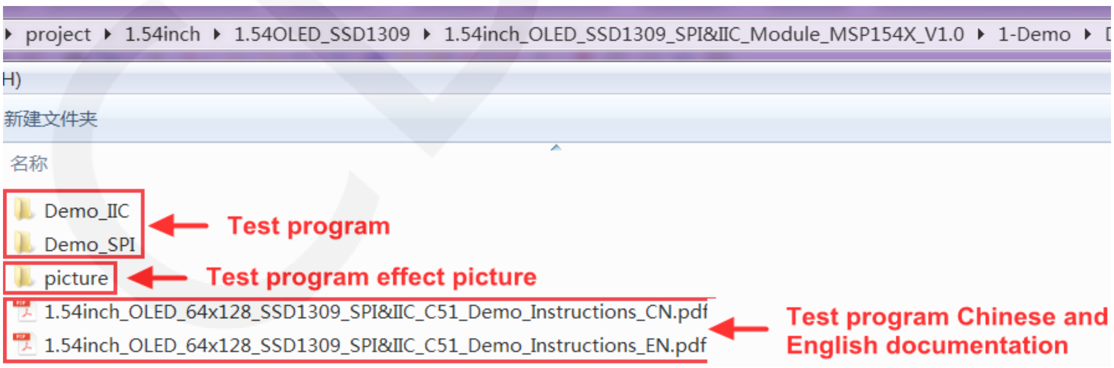
STC89C52RC and STC12C5A60S2 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to STC89/STC12 development board wiring pin	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	P17	
4	SDA	P15	
5	RES	P33	
6	DC	P12	VCC or GND
7	CS	P13	GND

Operating Steps:

A. Connect the IPS module and the C51 MCU according to the above wiring instructions, and power on;

B. Select the C51 test program to be tested, as shown below:

(Please refer to the test program description document in the test package for the test program description)



C. Open the selected test program project, compile and download;

detailed description of the C51 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/C51_Keil%26stc-isp_Use_Illustration_EN.pdf

D. If the OLED module displays characters and graphics normally, the program runs successfully;

5. MSP430 instructions

Wiring instructions:

See the interface description for pin assignments.

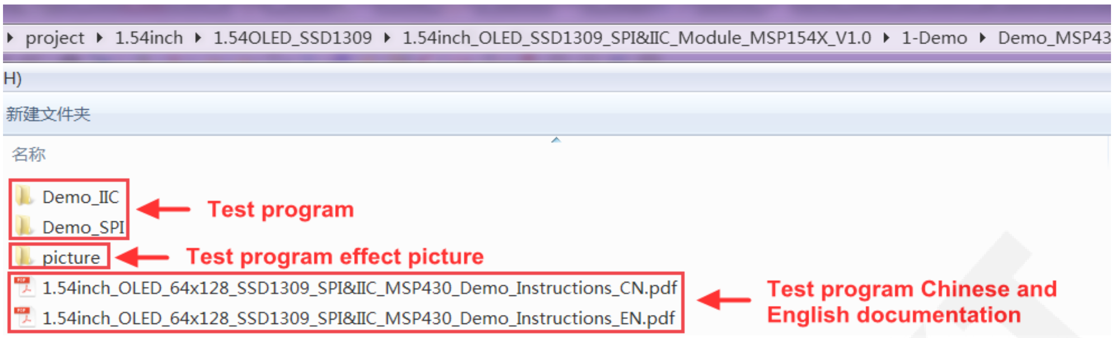
MSP430F149 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to MSP430 development board wiring pin	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	P33	
4	SDA	P31	
5	RES	P22	
6	DC	P21	VCC or GND
7	CS	P20	GND

Operating Steps:

A. Connect the IPS module and the MSP430 MCU according to the above wiring instructions, and power on;

B. Select the MSP430 test program to be tested, as shown below:

(Please refer to the test program description document in the test package for the test program description)



C. Open the selected test program project, compile and download;

detailed description of the MSP430 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/IAR_IDE%26MspFet_Use_Illustration_EN.pdf

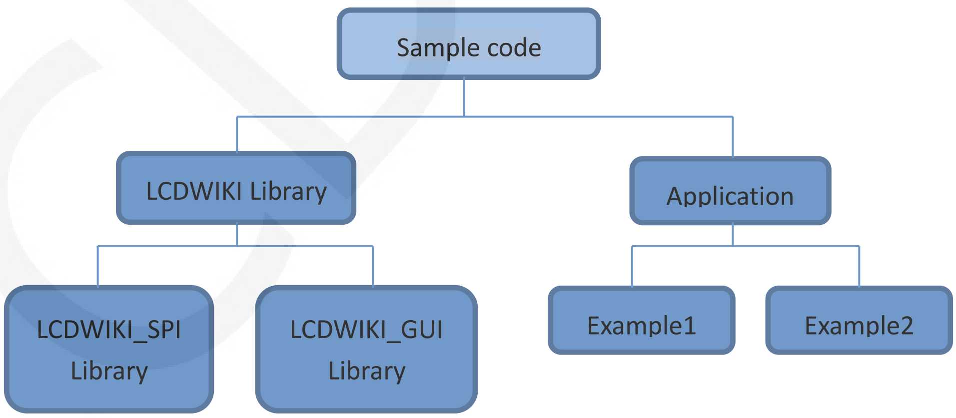
D. If the IPS module displays characters and graphics normally, the program runs successfully;

Software Description

1. Code Architecture

A. Arduino code architecture description

The code architecture is shown below



Arduino's test program code consists of two parts: the LCDWIKI library and application code.

The LCDWIKI library consists of two parts: the LCDWIKI_SPI library and the LCD_GUI library.

The application contains several test examples, each with different test content

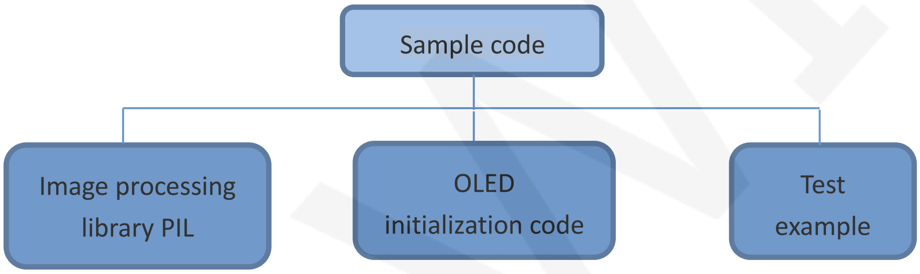
LCDWIKI_SPI is the underlying library, which is associated with hardware. It is mainly responsible for operating registers, including hardware module initialization, data and command transmission, pixel coordinates and color settings, and display mode configuration.

LCDWIKI_GUI is a middle-tier library, which is mainly responsible for drawing graphics and displaying characters using the API provided by the underlying library.

The application is to use the API provided by the LCDWIKI library to write some test examples to achieve some aspects of the test function.

B. **RaspberryPi code architecture description**

The python test program code architecture is shown below:



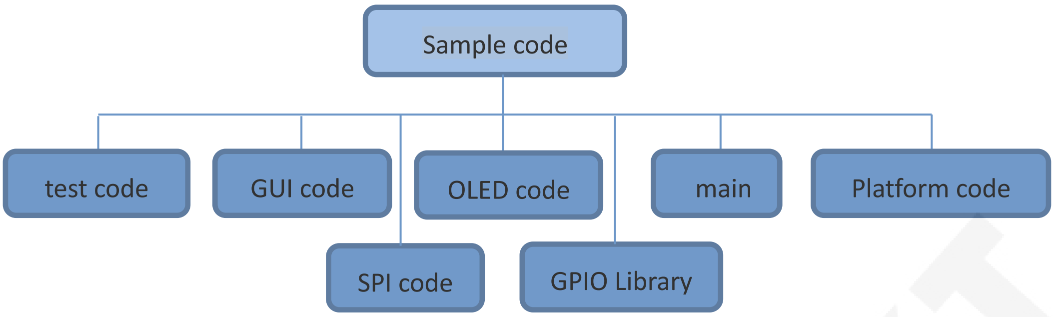
The python test program consists of but part: PIL image processing library, OLED initialization code, test sample code

PIL image processing library is responsible for image drawing, character and text display operations, etc.

OLDE initialization code is responsible for operating registers, including hardware module initialization, data and command transfer, pixel coordinates and color settings, display mode configuration, etc.

The test example is to use the API provided by the above two parts of the code to implement some test functions.

The bcm2835 and wiringPi test program code architecture is as follows:



The Demo API code for the main program runtime is included in the test code;

OLED initialization and related operations are included in the OLED code;

Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code;

The GPIO library provides GPIO operations;

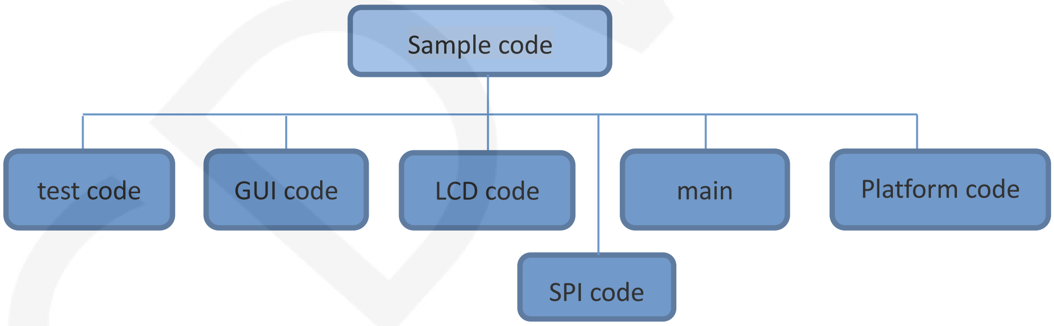
The main function implements the application to run;

Platform code varies by platform;

SPI initialization and configuration related operations are included in the SPI code;

C. C51, STM32 and MSP430 code architecture description

The code architecture is shown below:



The Demo API code for the main program runtime is included in the test code;

OLED initialization and related bin parallel port write data operations are included in the OLED code;

Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code;

The main function implements the application to run;

Platform code varies by platform;

SPI initialization and configuration related operations are included in the SPI code;

2. software SPI and hardware SPI description

The IPS module provides software SPI and hardware SPI sample code (except STC89C52RC, because it does not have hardware SPI function), the two sample code does not make any difference in the display content, but the following aspects are different:

A. display speed

The hardware SPI is significantly faster than the software SPI, which is determined by the hardware.

B. GPIO definition

The software SPI all control pins must be defined, any idle pin can be used, the hardware SPI data and clock signal pins are fixed (depending on the platform), other control pins should be defined by themselves, or any idle reference can be used. foot.

C. initialization

When the software SPI is initialized, only the GPIO for pin definition needs to be initialized (not required by the C51 platform). When the hardware SPI is initialized, the relevant control registers and data registers need to be initialized.

3. GPIO definition description

A. Arduino test program GPIO definition description

The Arduino test program GPIO definitions are placed in the application examples, and each application example can define GPIO. As shown in the figure below (take UNO MCU 4-wire software SPI test program as an example):

```
//paramters define
#define MODEL SSD1306
#define CS    A5
#define DC    A3
#define D1    11
#define D0    13
#define RES   A4
#define LED   -1   //if you don't need to control
```

If using the software SPI, all pin definitions can be modified to any other free GPIO.

If hardware SPI is used, D0 and D1 cannot be modified and do not need to be defined. Other GPIOs can be modified.

If a 3-wire SPI is used, the DC does not need to be defined.

B. RaspberryPi test program GPIO definition description

The RaspberryPi test program uses hardware SPI, so only three GPIO ports need to be defined. The bcm2835 and WiringPi test programs place the GPIO definition in the oled.h file, as shown in the following figure (take the 4-wire SPI test program as an example):

```
//-----------------OLED module pin definition----------------
#define OLED_CS    8   //chip selection control signal  bcm:8
#define OLED_DC    2   //data or command selection control signal bcm:2
#define OLED_RST   3   //reset control signal  bcm:3
```

The Python test program places the GPIO definition in each test example, as shown in the following figure (take the 4-wire SPI test program as an example):

```
# RaspberryPi pin configuration:
DC = 2
RES = 3
CS = 8
```

These three GPIOs can be modified according to the corresponding GPIO library code.

If a 3-wire SPI is used, the OLED_DC or DC does not need to be defined.

C. STM32 test program GPIO definition description

The STM32 test program GPIO definition is divided into two parts: control GPIO definition and SPI GPIO definition

The control GPIO definition is placed in oled.h, and the SPI GPIO definition is placed in spi.h, as shown in the following figure (take the STM32F103RCT6 software 4-wire SPI test program as an example):

```
//-----------------OLED端口定义----------------
#define OLED_CS   GPIO_Pin_11   //片选信号          PB11
#define OLED_DC   GPIO_Pin_10   //数据/命令控制信号  PB10
#define OLED_RST  GPIO_Pin_12   //复位信号          PB12
```

```
//--------------SPI总线引脚定义-----------------------
#define OLED_MOSI       GPIO_Pin_15   //OLED屏SPI写数据信号
#define OLED_CLK        GPIO_Pin_13   //OLED屏SPI时钟信号
```

If using the software SPI, all pin definitions can be modified to any other free GPIO.

If hardware SPI is used, OLED_MOSI and OLED_CLK cannot be modified and do not need to be defined. Other GPIOs Can be modified.

If you use a 3-wire SPI, OLED_DC does not need to be defined.

After modifying the GPIO definition, you need to initialize the GPIO to the OLED_Init_GPIO function in the oled.c file.

D. C51 test program GPIO definition description

The C51 test program GPIO definition is divided into two parts: control GPIO definition and SPI GPIO definition

The control GPIO definition is placed in oled.h, and the SPI GPIO definition is placed in spi.h, as shown in the following figure (take the STC12C5A60S2 software 4-wire SPI test program as an example):

```
//-----------------OLED端口定义----------------
sbit OLED_CS = P1^3;     //片选信号           P13
sbit OLED_DC = P1^2;    //数据/命令控制信号   P12
sbit OLED_RST = P3^3;    //复位信号           P33
//SPI的数据引脚定义和时钟引脚定义都可以任意修改
sbit OLED_MOSI = P1^5;    //OLED屏SPI写数据引脚  P15
sbit OLED_CLK = P1^7;     //OLED屏SPI时钟引脚    P17
```

If using the software SPI, all pin definitions can be modified to any other free GPIO.

If hardware SPI is used, OLED_MOSI and OLED_CLK cannot be modified and do not need to be defined. Other GPIOs can be modified. (Only STC12C5A60S2 microcontroller has hardware SPI function)

If you use a 3-wire SPI, OLED_DC does not need to be defined.

E. MSP430 test program GPIO definition description

MSP430's LCD non-SPI GPIO definition is placed in lcd.h, as shown below (take MSP430F149 software 4-wire SPI test program as an example):

```
//-----------------OLED端口定义----------------
#define OLED_CS   BIT0    //片选信号          P20
#define OLED_DC   BIT1   //数据/命令控制信号  P21
#define OLED_RST  BIT2   //复位信号          P22
```

All pin definitions can be modified and can be defined as any other free GPIO.

If you use a 3-wire SPI, OLED_DC does not need to be defined.

The GPIO definition of the MSP430 LCD SPI is placed in spi.h, as shown in the following figure (take the MSP430F149 software 4-wire SPI test program as an example):

```
//本测试程序使用的是软件SPI接口驱动
//SPI时钟信号以及SPI读、写信号引脚都可以更改

#define SPI_SCLK BIT3    //P33
#define SPI_MOSI BIT1    //P31
```

If using the software SPI, all pin definitions can be modified and can be defined as any other free GPIO.

If you use hardware SPI, these pins do not need to be defined.

4. SPI communication code implementation

A. Arduino test program SPI communication code implementation

The SPI communication code is implemented in the LCDWIKI_SPI library.

The 4-wire software and hardware SPI code implementation is shown below:

```
//spi write for hardware and software
void LCDWIKI_SPI::Spi_Write(uint8_t data)
{
    if(hw_spi)
    {
        SPI.transfer(data);
    }
    else
    {
        uint8_t val = 0x80;
        while(val)
        {
            if(data&val)
            {
                MOSI_HIGH;
            }
            else
            {
                MOSI_LOW;
            }
            CLK_LOW;
            CLK_HIGH;
            val >>= 1;
        }
    }
} « end Spi_Write »
```

The 3-wire software and hardware SPI code implementation is shown below:

```
void LCDWIKI_SPI::Spi_3_wire_Write(uint8_t data,uint8_t cmd)
{
    uint16_t txdata = 0;
    txdata = ((cmd<<15)|(data<<7));
    if(hw_spi)
    {
        SPI.transfer16(txdata);
    }
    else
    {
        uint16_t val = 0x8000;
        while(val>(1<<6))
        {
            if(txdata&val)
            {
                MOSI_HIGH;
            }
            else
            {
                MOSI_LOW;
            }
            CLK_LOW;
            CLK_HIGH;
            val >>= 1;
        }
    }
} « end Spi_3_wire_Write »
```

It is through the flag bit to decide whether to use software SPI or hardware SPI.

B. RaspberryPi test program SPI communication code implementation

The SPI communication code for the bcm2835 and wiringPi test programs is implemented in spi.c.

The SPI communication code for the python test program is implemented in oled.py.

The bcm2835 test program 4-wire hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(uint8_t byte)
 * @date       :2018-08-27
 * @function   :Write a byte of data using RaspberryPi hardware SPI
 * @parameters :Byte:Data to be written
 * @retvalue   :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    bcm2835_spi_transfer(byte);
}
```

The bcm2835 test program 3-wire hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(uint8_t byte, uint8_t cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of data using RaspberryPi hardware SPI
 * @parameters :Byte:Data to be written
                cmd:0-command
                    1-data
 * @retvalue   :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte, uint8_t cmd)
{
    uint16_t data=0;
    char txbuf[2]={0};
    data=((cmd<<15)|(byte<<7));
    txbuf[0]=(char)(data>>8);
    txbuf[1]=(char)(data&0xFF);
        bcm2835_spi_transfern(txbuf,2);
}
```

The wiringPi test program 4-wire hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(uint8_t byte)
 * @date       :2018-08-27
 * @function   :Write a byte of data using RaspberryPi hardware SPI
 * @parameters :Byte:Data to be written
 * @retvalue   :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    wiringPiSPIDataRW(CHANNEL,&byte,1);
}
```

The wiringPi test program 3-wire hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(uint8_t byte, uint8_t cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of data using RaspberryPi hardware SPI
 * @parameters :Byte:Data to be written
                cmd:0-command
                    1-data
 * @retvalue   :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte, uint8_t cmd)
{
    uint16_t data;
    unsigned char txbuf[2]={0};
    data=((cmd<<15)|(byte<<7));
    txbuf[0]=(unsigned char)(data>>8);
    txbuf[1]=(unsigned char)(data&0xFF);
    wiringPiSPIDataRW(CHANNEL,txbuf,2);
}
```

The python test program 4-wire hardware SPI code is implemented as shown below:

```
def writebyte(self,val,flag):
    """send one byte data to oled module"""
    if flag == OLED_COMMAND:
        GPIO.output(self.oleddc,GPIO.LOW)
    else:
        GPIO.output(self.oleddc,GPIO.HIGH)
    GPIO.output(self.oledcs,GPIO.LOW)
    self.oledspi.writebytes([val])
    self.oledspi.xfer([val],8000000)
    GPIO.output(self.oledcs,GPIO.HIGH)
```

The python test program 3-wire hardware SPI code is implemented as shown below:

```
def writebyte(self,val,flag):
    """send two byte data to oled module"""
    data=((flag<<15)|(val<<7))
    txbuf=[(data>>8)&0xFF,data&0xFF]
    GPIO.output(self.oledcs,GPIO.LOW)
    self.oledspi.writebytes(txbuf)
    self.oledspi.xfer(txbuf,8000000)
    GPIO.output(self.oledcs,GPIO.HIGH)
```

C. STM32 test program SPI communication code implementation

The SPI communication code is implemented in spi.c. (take STM32F103RCT6 test program as an example)

The 4-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************************
 * @name        :void SPI_WriteByte(u8 Data)
 * @date        :2018-08-27
 * @function    :Write a byte of data using STM32's Software SPI
 * @parameters  :Data:Data to be written
 * @retvalue    :None
******************************************************************/
void SPI_WriteByte(u8 Data)
{
  unsigned char i=0;
  for(i=8;i>0;i--)
  {
    if(Data&0x80)
    {
      OLED_MOSI_SET(); //写数据1
    }
    else
    {
      OLED_MOSI_CLR(); //写数据0
    }
    OLED_CLK_CLR();     //将时钟拉低拉高
    OLED_CLK_SET();     //发送1bit数据
    Data<<=1;
  }
}
```

Hardware SPI:

```
/*****************************************************************
 * @name        :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
 * @date        :2018-08-27
 * @function    :Write a byte of data using STM32's hardware SPI
 * @parameters  :SPIx: SPI type,x for 1,2,3
                 Byte:Data to be written
 * @retvalue    :Data received by the bus
******************************************************************/
u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
{
  while((SPIx->SR&SPI_I2S_FLAG_TXE)==RESET);    //等待发送区空
  SPIx->DR=Byte;    //发送一个byte
  while((SPIx->SR&SPI_I2S_FLAG_RXNE)==RESET);//等待接收完一个byte
  return SPIx->DR;                  //返回收到的数据
}
```

The 3-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(u8 data,u8 Cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of data using STM32's Software SPI
 * @parameters :Data:Data to be written
       Cmd:0-command
           1-data
 * @retvalue   :None
******************************************************************************/
void SPI_WriteByte(u8 data,u8 Cmd)
{
  unsigned char i=0;
  u16 Data;
  Data = ((Cmd<<15)|(data<<7));
  for(i=9;i>0;i--)
  {
    if(Data&0x8000)
    {
      OLED_MOSI_SET(); //写数据1
    }
    else
    {
      OLED_MOSI_CLR(); //写数据0
    }
    OLED_CLK_CLR();     //将时钟拉低拉高
    OLED_CLK_SET();     //发送1bit数据
    Data<<=1;
  }
}
```

Hardware SPI:

```
/*****************************************************************************
 * @name       :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte,u8 cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
                Byte:Data to be written
                cmd:0-write command
                    1-write data
 * @retvalue   :Data received by the bus
******************************************************************************/
u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte,u8 cmd)
{
  while((SPIx->SR&SPI_I2S_FLAG_TXE)==RESET);     //等待发送区空
  SPIx->DR=((cmd<<15)|(Byte<<7));    //发送两个byte
  while((SPIx->SR&SPI_I2S_FLAG_RXNE)==RESET);//等待接收完两个byte
  return SPIx->DR;                  //返回收到的数据
}
```

D. C51 test program SPI communication code implementation

The SPI communication code is implemented in spi.c. (taking the STC12C5A60S2 test program as an example)

The 4-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************************
 * @name        :void SPI_WriteByte(u8 byte)
 * @date        :2018-08-09
 * @function    :Write a byte of data using C
 * @parameters  :byte:Data to be written
 * @retvalue    :None
******************************************************************
void SPI_WriteByte(u8 byte)
{
  u8 i;
  for(i=0;i<8;i++)
  {
    if(byte&0x80)
    {
      OLED_MOSI_Set();
    }
    else
    {
      OLED_MOSI_Clr();
    }
    OLED_CLK_Clr();
    OLED_CLK_Set();
    byte<<=1;
  }
}
```

Hardware SPI:

```
/*****************************************************************
 * @name        :void SPI_WriteByte(u8 byte)
 * @date        :2018-08-09
 * @function    :Write a byte of data using C51's Hardware SPI
 * @parameters  :byte:Data to be written
 * @retvalue    :None
******************************************************************
void SPI_WriteByte(u8 byte)
{
      SPDAT = byte;    //发送一个字节
      while((SPSTAT & SPIF)==0) ; //等待发送完成
      SPSTAT = SPIF+WCOL;      //清0 SPIF和WCOL标志
}
```

The 3-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************
 * @name       :void SPI_WriteByte(u8 byte, u8 cmd)
 * @date       :2018-08-09
 * @function   :Write a byte of data using C51's so
 * @parameters :byte:Data to be written
                cmd:0-command
                    1-data
 * @retvalue   :None
******************************************************
void SPI_WriteByte(u8 byte, u8 cmd)
{
  u8 i;
  u16 Data=0;
  Data=((cmd<<15)|(byte<<7));
  for(i=0;i<9;i++)
  {
    if(Data&0x8000)
    {
      OLED_MOSI_Set();
    }
    else
    {
      OLED_MOSI_Clr();
    }
    OLED_CLK_Clr();
    OLED_CLK_Set();
    Data<<=1;
  }
}
```

Hardware SPI:

```
/**************************************************************
 * @name       :void SPI_WriteByte(u8 byte, u8 cmd)
 * @date       :2018-08-09
 * @function   :Write a byte of data using C51's Hardware SPI
 * @parameters :byte:Data to be written
                cmd:0-command
                    1-data
 * @retvalue   :None
***************************************************************
void SPI_WriteByte(u8 byte, u8 cmd)
{
    u8 i=0;
    u16 Data=0;
    Data=((cmd<<15)|(byte<<7));
    for(i=2;i>0;i--)
    {
      SPDAT = (Data>>((i-1)*8));     //发送一个字节
      while((SPSTAT & SPIF)==0) ; //等待发送完成
      SPSTAT = SPIF+WCOL;      //清0 SPIF和WCOL标志
    }

}
```

E. MSP430 test program SPI communication code implementation

The software SPI communication code is implemented in spi.c.

The 4-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************************
 * @name       :void  SPI_WriteByte(u8 Data)
 * @date       :2018-08-09
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
                Byte:Data to be written
 * @retvalue   :Data received by the bus
******************************************************************/
void SPI_WriteByte(u8 Data)
{
        unsigned char i=0;
        for(i=8;i>0;i--)
        {
          if(Data&0x80)
          SPI_MOSI_SET; //输出数据
      else SPI_MOSI_CLR;

      SPI_SCLK_CLR;
      SPI_SCLK_SET;
      Data<<=1;
        }
}
```

Hardware SPI:

```
/*****************************************************************************
 * @name       :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
 * @date       :2018-08-09
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
                Byte:Data to be written
 * @retvalue   :Data received by the bus
******************************************************************************/
u8 SPI_WriteByte(u8 Byte)
{
  while ((IFG1&UTXIFG0) ==0);   // wait while not ready / for RX
  U0TXBUF = Byte;
  while ((IFG1&URXIFG0)==0);   // wait for RX buffer (full)
  return (U0RXBUF);
}
```

The software SPI communication code is implemented in spi.c.

The 3-wire software and hardware SPI communication code implementation is as follows:

Software SPI:

```
/*****************************************************************
 * @name       :void  SPI_WriteByte(u8 Data)
 * @date       :2018-08-09
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
                Byte:Data to be written
 * @retvalue   :Data received by the bus
*****************************************************************
void SPI_WriteByte(u8 val, u8 cmd)
{
        unsigned char i=0;
        u16 Data=0;
        Data = ((cmd<<15)|(val<<7));
        for(i=9;i>0;i--)
        {
          if(Data&0x8000)
          SPI_MOSI_SET; //输出数据
          else SPI_MOSI_CLR;

          SPI_SCLK_CLR;
          SPI_SCLK_SET;
          Data<<=1;
        }
}
```

Hardware SPI:

```
/******************************************************************
 * @name       :void OLED_WR_Byte(unsigned dat,unsigned cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of content to the OLED screen
 * @parameters :dat:Content to be written
                cmd:0-write command
                                                                1-
 * @retvalue   :None
******************************************************************/
void OLED_WR_Byte(unsigned dat,unsigned cmd)
{
        u16 data=0;
        data=((cmd<<15)|(dat<<7));
        OLED_CS_Clr;
        SPI_WriteByte((data>>8)&0xFF);
        SPI_WriteByte(data&0xFF);
        OLED_CS_Set;
}
```

Common software

This set of test examples needs to display Chinese and English, symbols and pictures, so PCtoLCD2002 modulo software is used. Here, the setting of the modulo software is explained only for the test program.

The **PCtoLCD2002** modulo software settings are as follows:

Dot matrix format select Dark code

the modulo mode select the progressive mode(C51 and MSP430 test programs need to choose determinant)

Take the model to choose the direction (high position first)(C51 and MSP430 test procedures need to choose reverse (low position first))

Output number system selects hexadecimal number

Custom format selection C51 format

The specific setting method is as follows:

http://www.lcdwiki.com/Chinese_and_English_display_modulo_settings

Bedienungsanleitung
Best.-Nr. 86010030

HOBBYWING

Getaktetes Digital BEC 8A

Das Digital BEC bietet die Möglichkeit, die Empfangsanlage im Modell aus dem Antriebsakku zu versorgen. Die Eingangsspannung darf dabei zwischen 6,0V und 12,6V liegen (2-3 Zellen LiPo).

Im Gegensatz zu herkömmlichen Reglern mit BEC-Schaltkreis, kann das Digital BEC wesentlich höhere Ströme liefern und eine sichere Stromversorgung garantieren. Besonders bei Servos mit hoher Stromaufnahme und Antriebsakkus mit hohen Spannungen, kommt es schnell zur Überlastung der herkömmlichen BEC-Schaltkreise. Dies führt zu einer starken Hitzeentwicklung und kann zu Aussetzern in der RC-Anlage führen.

Technische Daten
Ausgangsspannung: 5V/8A oder 6V/8A wählbar (kurzzeitig 15A)
Eingangsspannung: 6-12,6V (2-3 Zellen LiPo)
Abmessungen: 42,0x39,0x9,0 mm
Gewicht: 38g

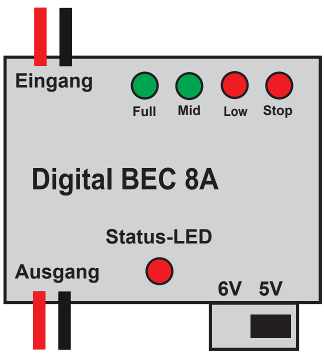
Features
Das Digital BEC besitzt einen Überlast- und Temperaturschutz. Der Wirkungsgrad des Spannungsreglers liegt bei ca. 92%. Durch die geringen Abmessungen passt der digitale Spannungsregler in nahezu jedes Modell. Der Regler erkennt die Zellenzahl automatisch und zeigt den Ladezustand über 4 LEDs an. Der Ausgangsstrom beträgt konstant 8A, kurzzeitig sind 15A möglich. Der Status des Reglers wird über eine LED angezeigt. Die LED leuchtet, wenn der Regler im normalen Bereich arbeitet. Die Stromversorgung des Empfängers erfolgt über zwei Kabel, die an den Empfänger angeschlossen werden.

Die elektronischen Bauteile auf der Platine sind weitestgehend abgeschirmt. Die Kabel sind mit einem Ferrit-Kern ausgestattet, um Störungen der RC-Anlage auszuschließen.

Der Anschluss des Digital BEC

WICHTIGER HINWEIS!
Das Digital BEC erzeugt ein elektromagnetisches Feld. Montieren Sie das BEC mindestens 5 cm oder mehr vom Empfänger entfernt im Modell!

Hinweis: Diesem Regler liegt ein separates Kabel bei, dass die BEC-Spannung um 0,7V reduziert. Dies ist erforderlich bei einigen Hi-Speed-Servos in Verbindung mit empfindlichen Kreiselsystemen. Das Kabel wird im Bedarfsfalle zwischen Kreisel und Empfänger oder zwischen Servo und Empfänger gesteckt.



Schalter für Ausgangs-Spannung

Achten Sie beim Anschluss auf die korrekte Polung! Bei Verpolung wird die Elektronik irreparabel zerstört! Der Spannungsregler ist ausschließlich für 2-3 zellige LiPo-Akkus zugelassen! NiMH- und NiCd-Akkus sollen NICHT an diesem Regler angeschlossen werden!

Einstellen der Ausgangsspannung
Die Ausgangsspannung für die RC-Anlage wird über den Schalter eingestellt.

LED Status-Anzeige
Die LED zeigt an, ob der Regler im normalen Bereich arbeitet. Leuchtet die LED nicht, müssen die Akku-Anschlüsse überprüft werden.

LED Kapazitäts-Anzeige

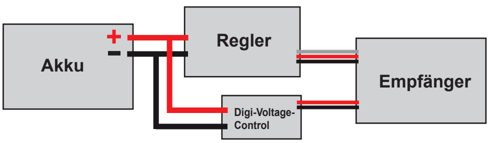
LED Status				The voltage of the lithium battery pack	
Full	Mid	Low	Stop	2S battery pack	3S battery pack
○	○	○	○	7.8—8.4V	11.7—12.6V
●	○	○	○	7.2—7.8V	10.8—11.7V
●	●	○	○	6.6—7.2V	9.9—10.8V
●	●	●	○	5.4—6.6V	<9.9V

○ = LED leuchtet   ● = LED leuchtet nicht

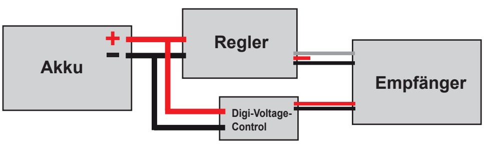
Leuchtet nur eine LED bei Verwendung eines 3-zelligen LiPo-Akkus, bedeutet dies dass die Akkuspannung unter 9,9V liegt. In diesem Fall muss der Akku geladen werden, um eine Tiefentladung zu vermeiden.

Schalter
Mit dem Schalter kann die RC-Anlage bequem ein- und ausgeschaltet warden.

Elektronischer Regler OHNE BEC-Funktion
Der digitale Spannungsregler wird mit dem Antriebsakku verbunden. Der Servostecker wird in einen freien Kanal des Empfängers gesteckt.



Elektronischer Regler MIT BEC-Funktion
In diesem Fall muss zunächst die BEC-Funktion des Reglers deaktiviert werden! Dies kann entweder über die Software im Regler erfolgen oder es wird der ROTE DRAHT aus dem Empfängeranschlusskabel des Reglers unterbrochen.



Hype • Nikolaus-Otto-Str. 4 • D-24568 Kaltenkirchen
Helpdesk: 04191-932678 • helpdesk@hype-rc.de • www.hype-rc.de

Instruction Manual
Best.-Nr. 86010030

8A Switching-Mode UBEC

1. Why do you need UBEC?

The 8A-UBEC is a switching-mode DC-DC regulator supplied with a 2-3 cells lithium battery pack and outputs a consistent safe voltage for your receiver, gyro and servos. It is very suitable for nitro powered RC helicopter (above 30 class) and big fixed-wing aircraft.
Compared with the linear mode UBEC, the overall efficiency of the switching-mode BEC is much higher, so it can extend the working time of the receiver battery pack, and because a switching- mode UBEC can significantly reduce the heat emission, it can avoid the loss of control caused by the over-heat problem which is frequently happened with the linear mode UBEC.

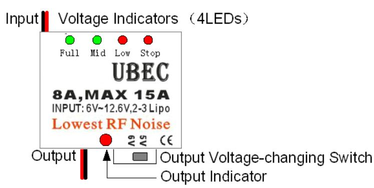
2. Specification:

2.1. **Output:** 5V/8A or 6V/8A (Changeable with an output-voltage select switch)
2.2. **Input:** 6V-12.6V (2-3 cells lithium battery pack)
2.3. **Size:** 42mm*39mm*9mm (length*width*height)
2.4. **Weight:** 38g
2.5. **Quiescent current:** 60mA

3. Features:

3.1. Designed with an advanced switching-mode DC-DC regulator IC.
3.2. The output current is very large, the continuous output current is up to 8A, and the burst output current is 15A.
3.3. With the output short-circuit protection function.
3.4. A metal shield covers almost all the electronic components, and a specially made filter (ferrite ring) is attached with the output wires to significantly reduce the electro magnetic interference.
3.5. Automatically detects the number of the lithium battery pack (2 cells or 3 cells), and shows the battery capacity with 4 indicators (LEDs).
3.6. Shows the working status with an indicator (LED), lights when the output is in normal range.
3.7. 2 output leads to reduce the resistance when connecting the UEBC to the receiver.
3.8. Accessory: A step-down voltage regulator with 0.7V down (from 6.0V to 5.3V).

4. Wiring Method



5. Special Explanation

5.1. Although we have tried our best to reduce the electromagnetic interference caused by switching-model UBEC, it still may cause some interference to the receiver. So please install the filter far away from the UBEC's main board, and DON'T stack the filter on the main board. Please put the whole UBEC as far as possible away from the receiver.

5.2. This UBEC is only designed for using lithium batter pack; we don't recommend the use of NiMh / NiCd battery pack.

5.3. The input polarity must be correct; otherwise the UBEC will be damaged. Please check the polarity carefully before connecting the battery pack.

6. How to Use the UBEC?

6.1. Change the output voltage
The voltage is chosen by an output-voltage select switch.

6.2. Working status indicator (LED)
[illegible] when the UBEC has the normal output. If it doesn't light, please check the battery connections.

6.3. Battery capacity indicators (4 LEDs)

LED Status				The voltage of the lithium battery pack	
Full	Mid	Low	Stop	2S battery pack	3S battery pack
○	○	○	○	7.8—8.4V	11.7—12.6V
●	○	○	○	7.2—7.8V	10.8—11.7V
●	●	○	○	6.6—7.2V	9.9—10.8V
●	●	●	○	5.4—6.6V	<9.9V
4 LEDs flash at the same time				1)The voltage <5.4V 2)The voltage >13.5V	1)The voltage >13.5V
One LED flashes for a short time				The voltage of the battery pack is just at the critical edge of each range.	

○ means the LED lights, ● means the LED does not light

When you are using a 3 cells lithium battery pack, if there is only one LED („STOP") lights, that means the voltage is less than 9.9v, please change the battery pack as soon as possible, otherwise it will be damaged because of over-discharging. For such a fully-discharged 3S battery pack, if the voltage is less than 9V, please don't use it again before it is recharged, otherwise the UBEC may mistakenly consider this battery as 2 cells, so the power capacity indication function will be confused.

6.4. Turn on or turn off the output
Set the main switch to the „ON" position to turn on the output; Set the main switch to the "OFF" position to turn off the output.

6.5. About the 0.7V step-down voltage regulator
Allowing use of Futaba servo models 9241, 9251, 9253, 9254, 9255, 9256 and other digital servos not capable of handling 6V. This small device can change the voltage from 6V to 5.3V. When the UBEC output is set to 6V, the step-down voltage regula tor is useful.
Method: Just connect the regulator inline between the Gyro and the rudder servo (Or between the receiver and the servo), that's OK.
If you are using a servo that can accept 6V input, the regulator is not required.

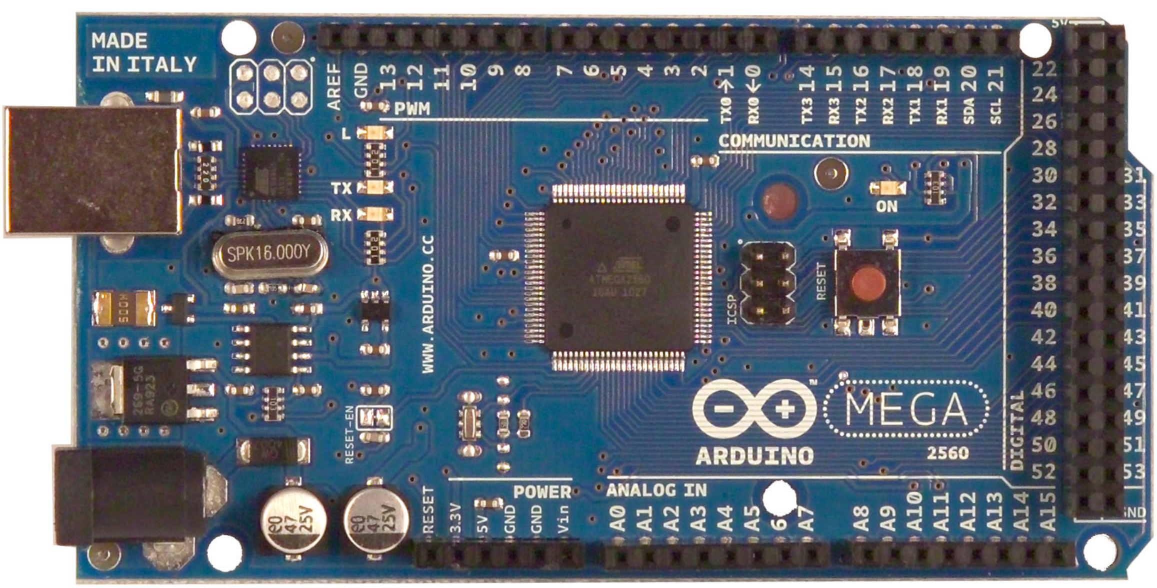
RobotShop

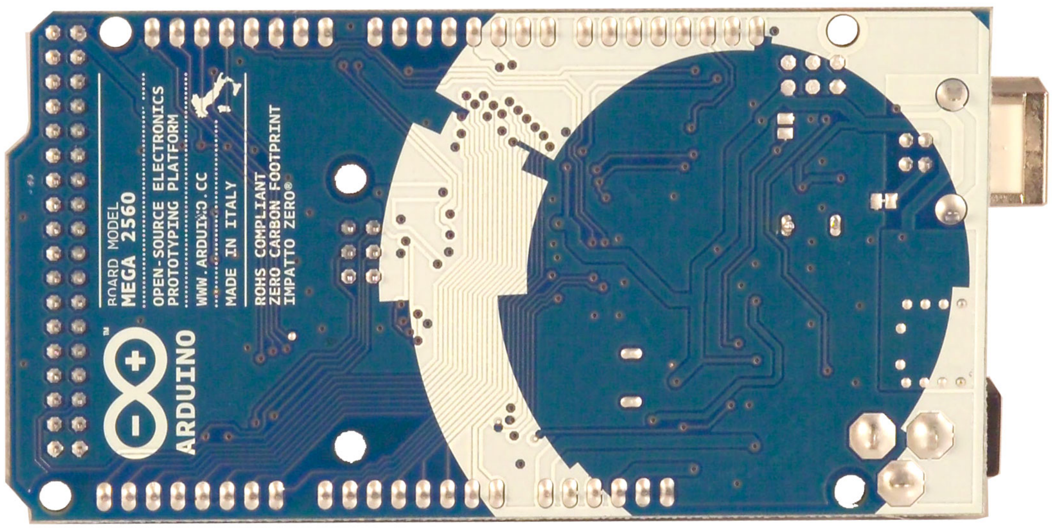
www.robotshop.com

La robotique à votre service! - Robotics at your service!

ARDUINO

Arduino Mega 2560 Datasheet





Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH

value, the LED is on, when the pin is LOW, it's off.

- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It

communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I_2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

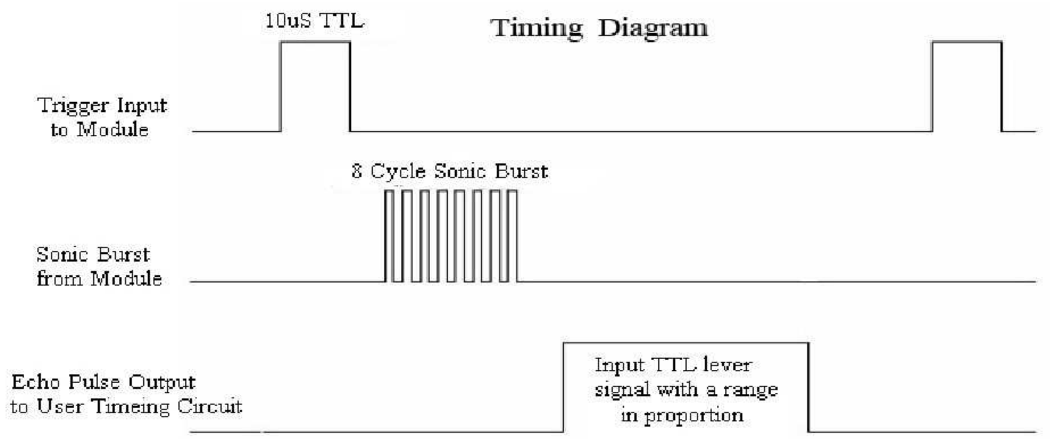
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



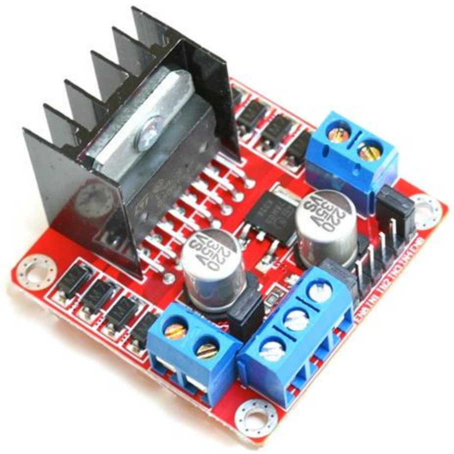
Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com

L298N Dual H-Bridge Motor Driver

This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

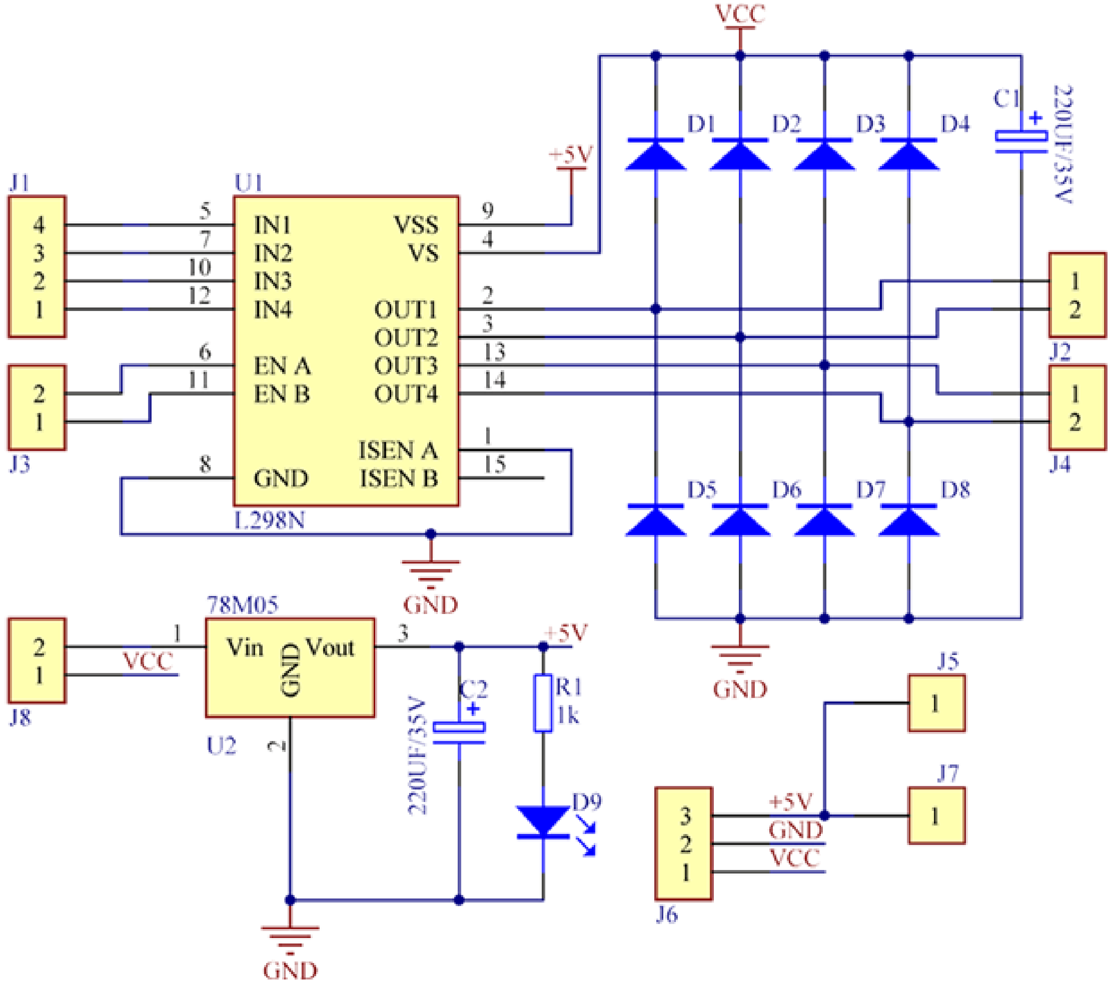


SKU: **DRV-1006**

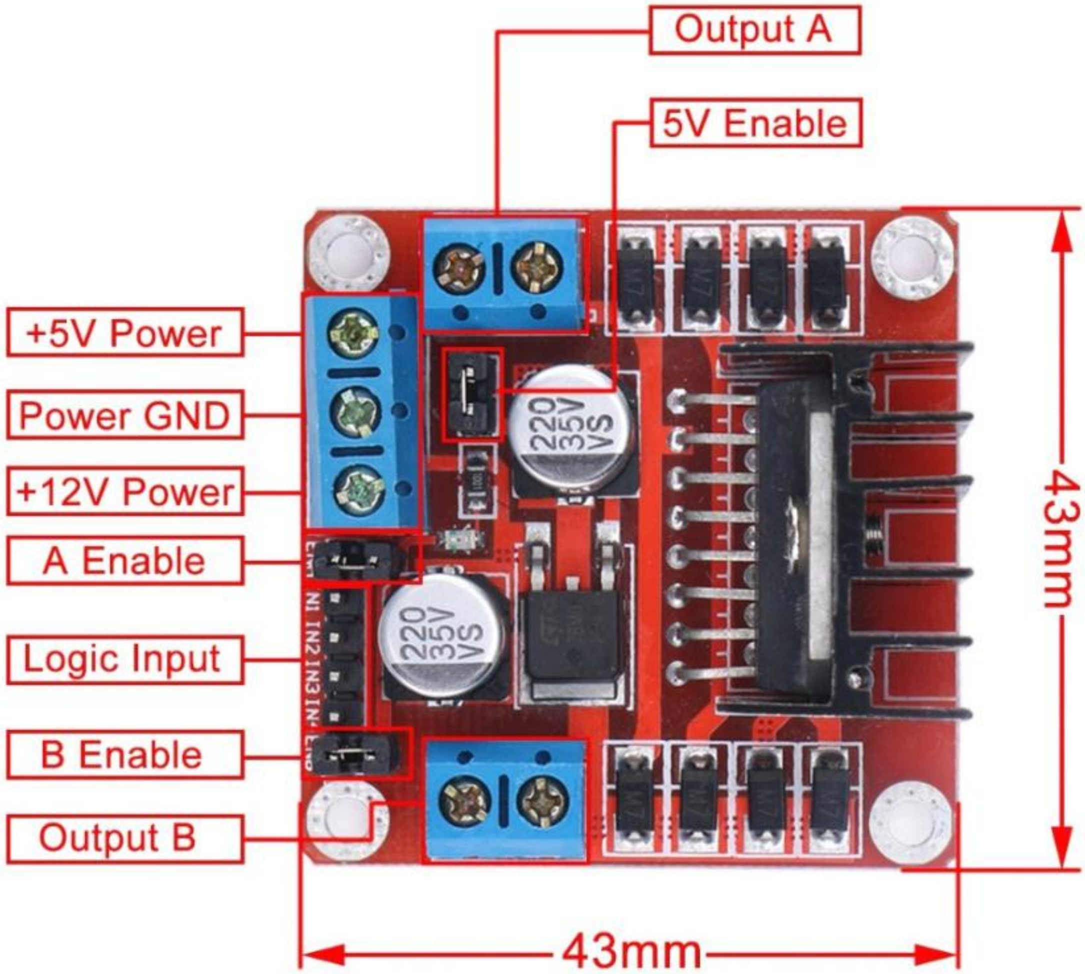
Brief Data:

- Input Voltage: 3.2V ~ 40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver.
- Power Supply: 5 ~ 35 Vdc.
- Peak current: 2 Amp.
- Operating current range: 0 ~ 36mA.
- Control signal input voltage range :
  - Low: -0.3V ⩽ Vin ⩽ 1.5V.
  - High: 2.3V ⩽ Vin ⩽ Vss.
- Enable signal input voltage range :
  - Low: -0.3 ⩽ Vin ⩽ 1.5V (control signal is invalid).
  - High: 2.3V ⩽ Vin ⩽ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

Schematic Diagram:



Board Dimension & Pins Function:



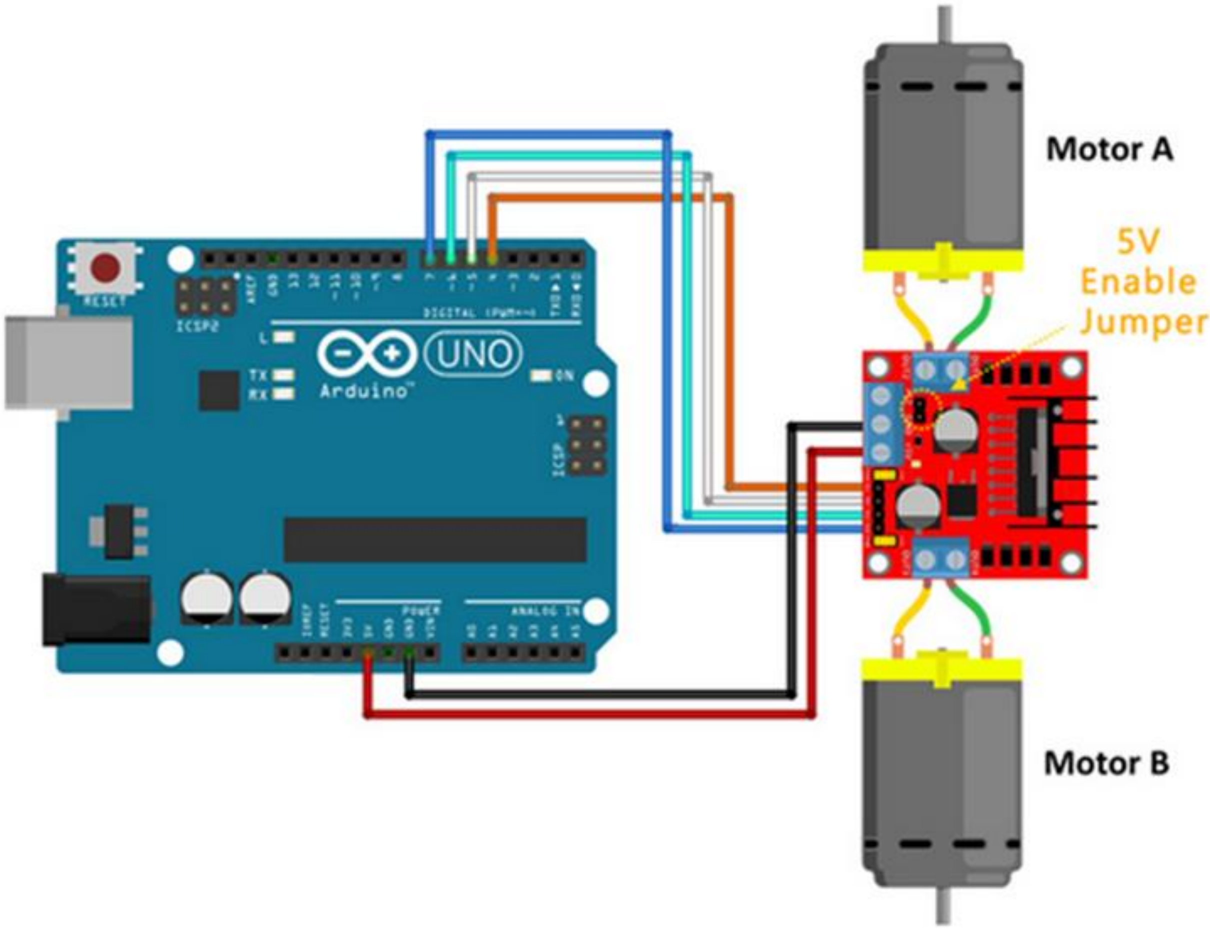
Web Resources:

- https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

Connection Examples:

Controlling 2-DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.



Sketch Listing:

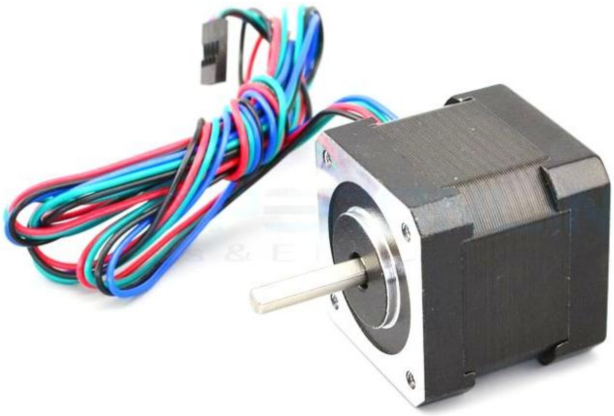
Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

```
/*============================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : L298N Motor Driver
//  Source-Code : L298N_Motor.ino
//  Program: Control 2 DC motors using L298N H Bridge Driver
//============================================================================
*/

// Definitions Arduino pins connected to input H Bridge
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
 // Set the output pins
```

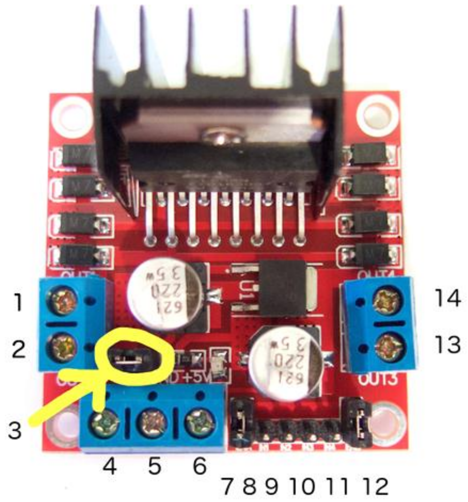
```
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}

void loop()
{
 // Rotate the Motor A clockwise
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotate the Motor B clockwise
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);

 // Rotates the Motor A counter-clockwise
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotates the Motor B counter-clockwise
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);
}
```

Controlling Stepper Motor

In this example we have a typical NEMA-17 stepper motor with four wires:



The key to successful stepper motor control is identifying the wires - that is which one is which. You will need to determine the A+, A-, B+ and B- wires. With our example motor these are red, green, yellow and blue. Now let's get the wiring done.



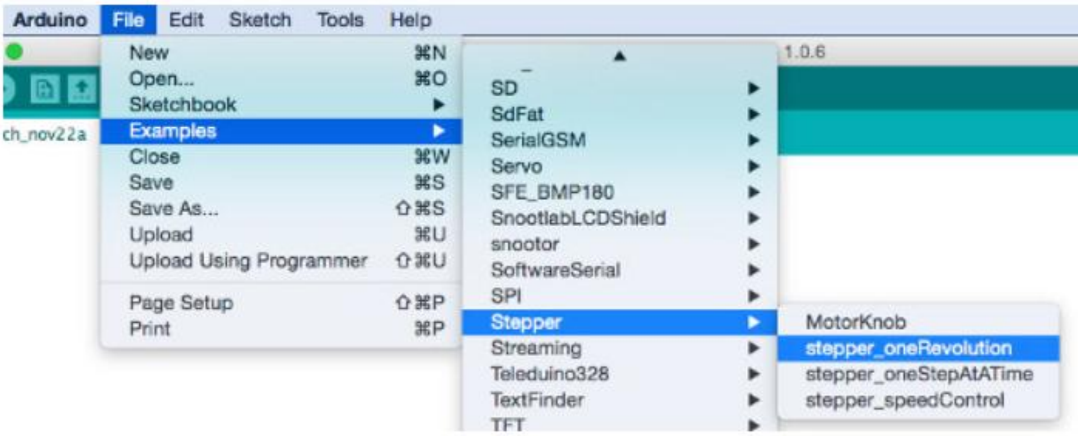
Connect the A+, A-, B+ and B- wires from the stepper motor to the module connections 1, 2, 13 and 14 respectively. Place the jumpers included with the L298N module over the pairs at module points 7 and 12. Then connect the power supply as required to points 4 (positive) and 5 (negative/GND).

Once again if your stepper motor's power supply is less than 12V, fit the jumper to the module at point 3 which gives you a neat 5V power supply for your Arduino.

Next, connect L298N module pins IN1, IN2, IN3 and IN4 to Arduino digital pins D8, D9, D10 and D11 respectively. Finally, connect Arduino GND to point 5 on the module, and Arduino 5V to point 6 if sourcing 5V from the module.

Controlling the stepper motor from your sketches is very simple, thanks to the *Stepper* Arduino library included with the Arduino IDE as standard.

To demonstrate your motor, simply load the "*stepper_oneRevolution*" sketch that is included with the *Stepper* library, for example:



Finally, check the value for

```
const int stepsPerRevolution = 200;
```

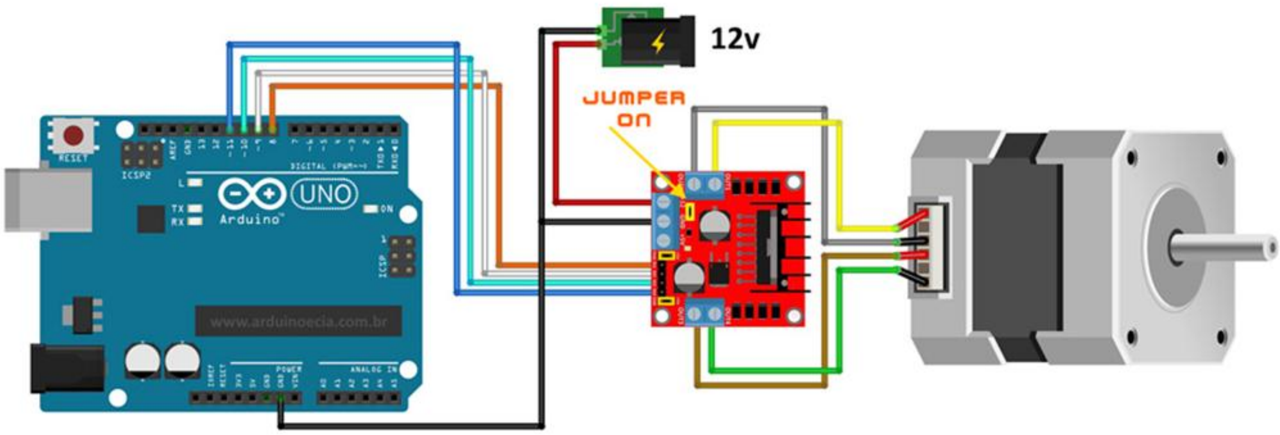
in the sketch and change the 200 to the number of steps per revolution for your stepper motor, and also the speed which is preset to 60 RPM in the following line:

```
myStepper.setSpeed(60);
```

Now you can save and upload the sketch, which will send your stepper motor around one revolution, then back again. This is achieved with the function

```
myStepper.step(stepsPerRevolution); // for clockwise
myStepper.step(-stepsPerRevolution); // for anti-clockwise
```

Connection for the sketch "*stepper_oneRevolution*":



HandsOn Technology provides a multimedia and interactive platform for everyone interested in electronics. From beginner to diehard, from student to lecturer. Information, education, inspiration and entertainment. Analog and digital, practical and theoretical; software and hardware.

Hands*On* Technology support Open Source Hardware (OSHW) Development Platform.

Learn : Design : Share

www.handsontec.com

The Face behind our product quality…

In a world of constant change and continuous technological development, a new or replacement product is never far away – and they all need to be tested.

Many vendors simply import and sell wihtout checks and this cannot be the ultimate interests of anyone, particularly the customer. Every part sell on Handsotec is fully tested. So when buying from Handsontec products range, you can be confident you're getting outstanding quality and value.

We keep adding the new parts so that you can get rolling on your next project.

Breakout Boards & Modules

Connectors

Electro-Mechanical Parts

Engineering Material

Mechanical Hardware

P

Electronics Components

Power Supply

ARDUINO

Arduino Board & Shield

Tools & Accessory

logitech

Logitech HD C270 Webcam for Education

Learning comes to life with brilliant, lifelike interactions

THE VALUE OF DYNAMIC VIDEO

Whether presenting live or recording material for sharing later, a standalone webcam provides the flexibility to take still or video images from different angles and show work more clearly, making instruction more dynamic, detailed and interactive. The result?

More engaged, connected and involved students.

works with chromebook

MEET THE HD C270 FOR EDUCATION

WIDESCREEN HD VIDEO

720p video displays lifelike, natural images even in dim lighting, to enable students to see detailed information.

CRISP VIDEO

Camera takes high-quality pictures for in-class sharing and for use in homework or other studying.

CLEAR SOUND

Noise-reducing mic filters out background noise so students can focus and hear every word clearly.

FLEXIBLE POSITIONING

Clip fits a screen, shelf or tripod so teachers can move to fit their setup and display multiple angles to students.

A COMMITMENT TO EDUCATION

At Logitech, we are curious lifelong learners, constantly questioning and inventing. We are relentless in our pursuit of tools that make an impact.

This passion drives us every day to design innovative products that create transformational and accessible experiences for students, teachers and schools.

Supporting education is inherent to our mission as we know that today's students are tomorrow's inventors.

Every one of our vcation products has been designed with the needs of schools in mind. When you choose Logitech for your school, you can count on:

EASY CLEANABILITY

Designed & tested to withstand cleaning and disinfecting after each use, supporting safe, long-term, shared use[1].

RUGGED DURABILITY

Our education products are drop-tested to withstand falls from standard school desk heights.

PLUG-AND-PLAY USB COMPATIBILITY

Get students and teachers working easily with hassle-free setup — simply plug the web-cam into the USB port and go.

3-YEAR WARRANTY WITH CUSTOMER CARE SUPPORT

Our products are designed to work without fail but should a problem occur, we've got your back.

WHY IT MATTERS

A standalone webcam makes it easier to share work and builds strong visual connections.

89% of teachers said their Logitech webcam enabled more seamless lesson delivery.[2]

81% of teachers said their Logitech webcam increase student engagement during lessons.[2]

HD C270 WEBCAM SPECIFICATIONS

FEATURES	C270 WEBCAM FOR EDUCATION	C270 WEBCAM FOR CONSUMER
VIDEO DISPLAY	Widescreen, High-Definition	
VIDEO RESOLUTION, MAX	720p/30fps	
CAMERA	HD 720p video	
FOCUS TYPE	Fixed	
LENS TECHNOLOGY	Standard	
FIELD OF VIEW	60"	
CERTIFICATIONS	Works with Chromebook	
COMPATIBILITY, OPERATING SYSTEM	Chrome OS™[3] Windows 7,8,10 Mac OS 10.10 or later Android v5.0 or above	
COMPATIBILITY, VIDEO CALLING SOFTWARE	Works with all popular platforms including Zoom, Google Meet and Microsoft Teams. Works in usb video device class mode with supported clients.	
CABLE LENGTH	5ft/1.5m	
PACKAGE CONTENTS	C270 Webcam for Education Cable User Documentation	C270 Webcam Cable User Documentation
WARRANTY	3 years with Customer Care support	2 years, limited
PACKAGING	Education packaging designed for fast unboxing and quick scanning of products without the need to remove each item	Standard
PART NO.	960-000694	981-000612

READY TO GET STARTED?

Contact Logitech Education Sales
Education@Logitech.com

logitech®

[1] Tested to withstand 2,700 wipe cycles with alcohol; equal to 5 classroom sessions per day, 180 classroom days per year over 3 years.

[2] Logitech 2020 survey of teachers across the US after receiving donated Logitech products, n=1381.

[3] This product has been certified by Logitech to meet Google's compatibility standards. Google is not responsible for the operation of this product or its compliance with safety requirements.

MAX30100

Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health

General Description

The MAX30100 is an integrated pulse oximetry and heart-rate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals.

The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

Applications

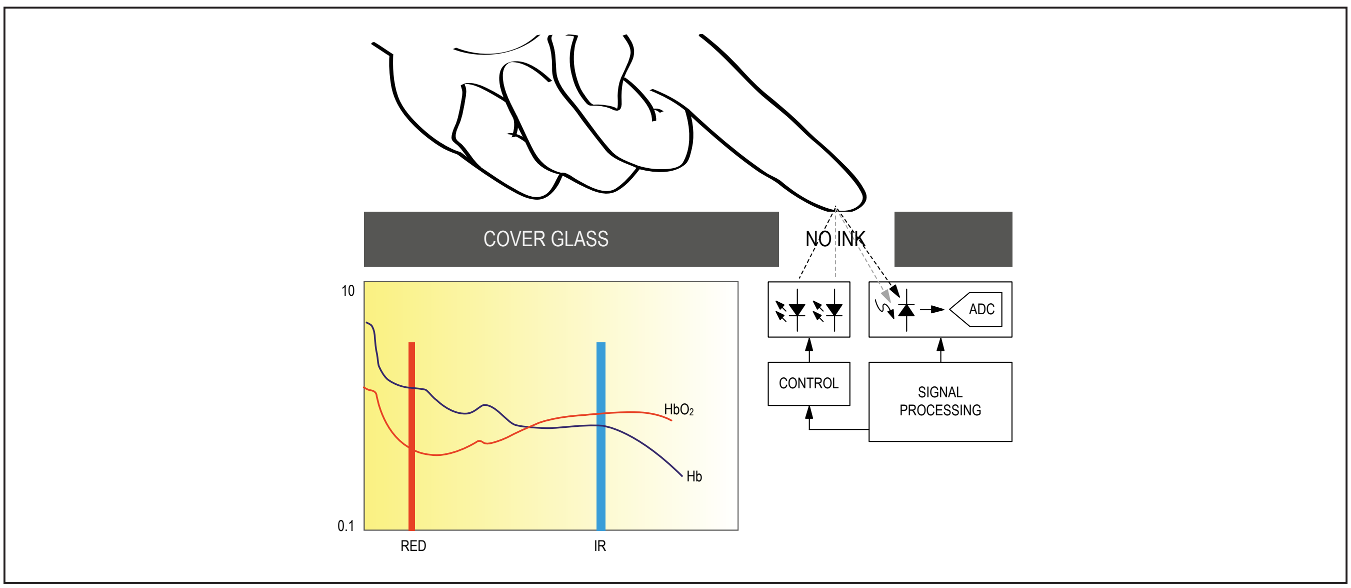
- Wearable Devices
- Fitness Assistant Devices
- Medical Monitoring Devices

Benefits and Features

- Complete Pulse Oximeter and Heart-Rate Sensor Solution Simplifies Design
  - Integrated LEDs, Photo Sensor, and High-Performance Analog Front -End
  - Tiny 5.6mm x 2.8mm x 1.2mm 14-Pin Optically Enhanced System-in-Package
- Ultra-Low-Power Operation Increases Battery Life for Wearable Devices
  - Programmable Sample Rate and LED Current for Power Savings
  - Ultra-Low Shutdown Current (0.7µA, typ)
- Advanced Functionality Improves Measurement Performance
  - High SNR Provides Robust Motion Artifact Resilience
  - Integrated Ambient Light Cancellation
  - High Sample Rate Capability
  - Fast Data Output Capability

***Ordering Information* appears at end of data sheet.**

System Block Diagram



Absolute Maximum Ratings

V_{DD} to GND ....-0.3V to +2.2V
GND to PGND ....-0.3V to +0.3V
x_DRV, x_LED+ to PGND ....-0.3V to +6.0V
All Other Pins to GND ....-0.3V to +6.0V
Output Short-Circuit Current Duration......................Continuous
Continuous Input Current into Any Terminal ....±20mA

Continuous Power Dissipation (T_A = +70°C)
OESIP (derate 5.8mW/°C above +70°C) 464mW
Operating Temperature Range.......................... -40°C to +85°C
Soldering Temperature (reflow).....................................+260°C
Storage Temperature Range........................... -40°C to +105°C

Package Thermal Characteristics (Note 1)

OESIP
Junction-to-Ambient Thermal Resistance (θ_{JA}) ....150°C/W
Junction-to-Case Thermal Resistance (θ_{JC})............170°C/W

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to **www.maximintegrated.com/thermal-tutorial**.

Electrical Characteristics

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY							
Power-Supply Voltage	V_{DD}	Guaranteed by RED and IR count tolerance		1.7	1.8	2.0	V
LED Supply Voltage (R_LED+ or IR_LED+ to PGND)	V_{LED+}	Guaranteed by PSRR of LED Driver		3.1	3.3	5.0	V
Supply Current	I_{DD}	SpO_2 and heart rate modes, PW = 200µs, 50sps			600	1200	µA
		Heart rate only mode, PW = 200µs, 50sps			600	1200	
Supply Current in Shutdown	I_{SHDN}	T_A = +25°C, MODE = 0x80			0.7	10	µA
SENSOR CHARACTERISTICS							
ADC Resolution					14		bits
Red ADC Count (Note 3)	RED_C	Propriety ATE setup RED_PA = 0x05, LED_PW = 0x00, SPO2_SR = 0x07, T_A = +25°C		23,000	26,000	29,000	Counts
IR ADC Count (Note 3)	IR_C	Propriety ATE setup IR_PA = 0x09, LED_PW = 0x00, SPO2_SR = 0x07, T_A = +25°C		23,000	26,000	29,000	Counts
Dark Current Count	DC_C	RED_PA = IR_PA = 0x00, LED_PW = 0x03, SPO2_SR = 0x01			0	3	Counts
DC Ambient Light Rejection (Note 4)	ALR	Number of ADC counts with finger on sensor under direct sunlight (100K lux) LED_PW = 0x03, SPO2_SR = 0x01	RED LED		0		Counts
			IR LED		0		

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
IR ADC Count—PSRR (V_{DD})	$PSRR_{VDD}$	Propriety ATE setup 1.7V < V_{DD} < 2.0V, LED_PW = 0x03, SPO2_SR = 0x01, IR_PA = 0x09, IR_PA = 0x05, T_A = +25°C		0.25	2	%
		Frequency = DC to 100kHz, 100mV_{P-P}		10		LSB
RED/IR ADC Count—PSRR (X_LED+)	$PSRR_{LED}$	Propriety ATE setup 3.1V < X_LED+ < 5V, LED_PW = 0x03, SPO2_SR = 0x01, IR_PA = 0x09, IR_PA = 0x05, T_A = +25°C		0.05	2	%
		Frequency = DC to 100kHz, 100mV_{P-P}		10		LSB
ADC Integration Time	INT	LED_PW = 0x00		200		µs
		LED_PW = 0x03		1600		µs
IR LED CHARACTERISTICS (Note 4)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	870	880	900	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		30		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		1.4		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		6.5		mW
RED LED CHARACTERISTICS (Note 4)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	650	660	670	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		20		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		2.1		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		9.8		mW
TEMPERATURE SENSOR						
Temperature ADC Acquisition Time	T_T	T_A = +25°C		29		ms
Temperature Sensor Accuracy	T_A	T_A = +25°C		±1		°C
Temperature Sensor Minimum Range	T_{MIN}			-40		°C
Temperature Sensor Maximum Range	T_{MAX}			85		°C

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DIGITAL CHARACTERISTICS (SDA, SDA, $\overline{INT}$)						
Output Low Voltage SDA, $\overline{INT}$	V_{OL}	I_{SINK} = 6mA			0.4	V
I2C Input Voltage Low	V_{IL_I2C}	SDA, SCL			0.4	V
I2C Input Voltage High	V_{IH_I2C}	SDA, SCL	1.4			V
Input Hysteresis	V_{HYS}	SDA, SCL		200		mV
Input Capacitance	C_{IN}	SDA, SCL		10		pF
Input Leakage Current	I_{IN}	V_{IN} = 0V, T_A = +25°C (SDA, SCL, INT)		0.01	1	µA
		V_{IN} = 5.5V, T_A = +25°C (SDA, SCL, INT)		0.01	1	µA
I2C TIMING CHARACTERISTICS (SDA, SDA, $\overline{INT}$)						
I2C Write Address				AE		Hex
I2C Read Address				AF		Hex
Serial Clock Frequency	f_{SCL}		0		400	kHz
Bus Free Time Between STOP and START Conditions	t_{BUF}		1.3			µs
Hold Time (Repeated) START Condition	$t_{HD,START}$		0.6			µs
SCL Pulse-Width Low	t_{LOW}		1.3			µs
SCL Pulse-Width High	t_{HIGH}		0.6			µs
Setup Time for a Repeated START Condition	$t_{SU,START}$		0.6			µs
Data Hold Time	$t_{HD,DAT}$		0		900	ns
Data Setup Time	$t_{SU,DAT}$		100			ns
Setup Time for STOP Condition	$t_{SU,STOP}$		0.6			µs
Pulse Width of Suppressed Spike	t_{SP}		0		50	ns
Bus Capacitance	C_B				400	pF
SDA and SCL Receiving Rise Time	t_R		20 + 0.1C_B		300	ns
SDA and SCL Receiving Fall Time	t_{RF}		20 + 0.1C_B		300	ns
SDA Transmitting Fall Time	t_{TF}		20 + 0.1C_B		300	ns

Note 2: All devices are 100% production tested at T_A = +25°C. Specifications over temperature limits are guaranteed by Maxim Integrated's bench or proprietary automated test equipment (ATE) characterization.

Note 3: Specifications are guaranteed by Maxim Integrated's bench characterization and by 100% production test using proprietary ATE setup and conditions.

Note 4: For design guidance only. Not production tested.

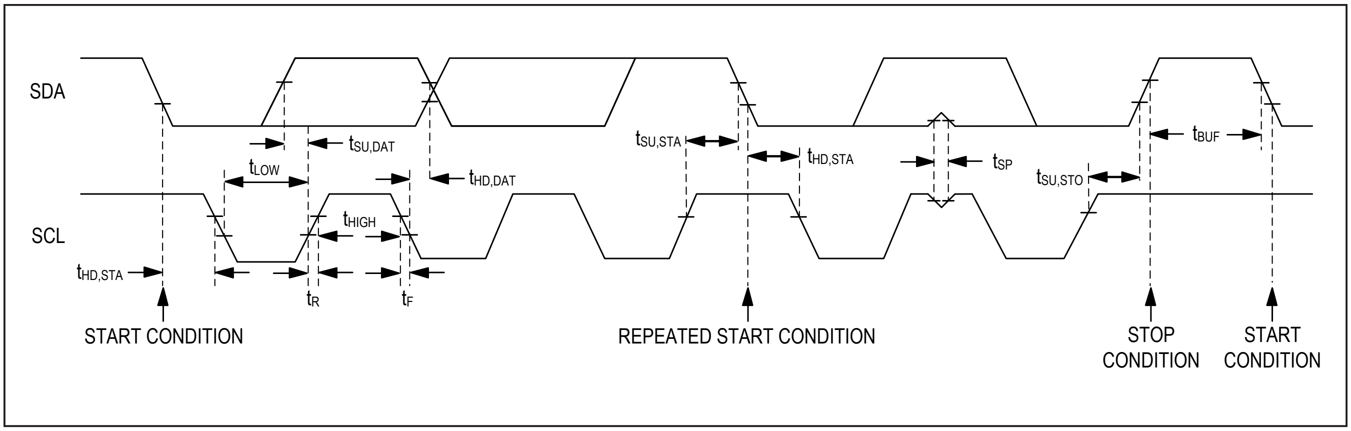
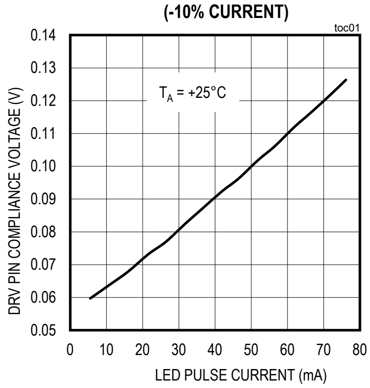


Figure 1. I2C-Compatible Interface Timing Diagram

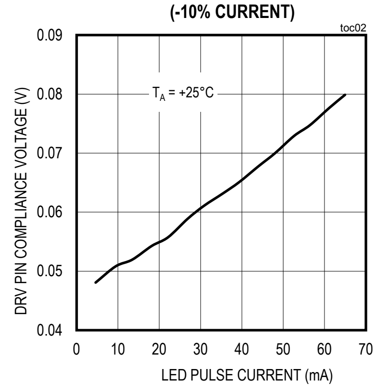
Typical Operating Characteristics

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, unless otherwise noted.)

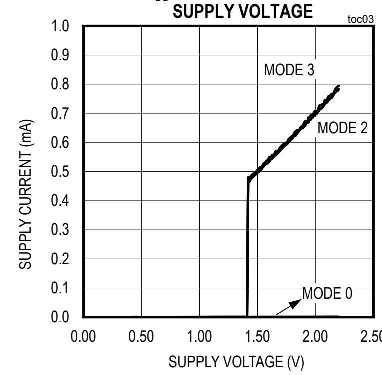
RED LED SUPPLY HEADROOM (-10% CURRENT)



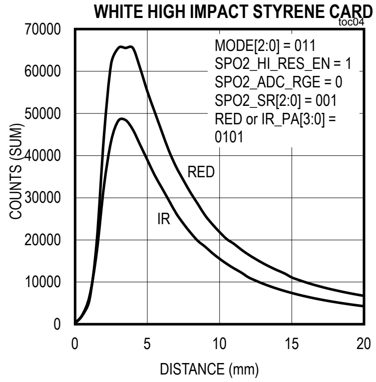
IR LED SUPPLY HEADROOM (-10% CURRENT)



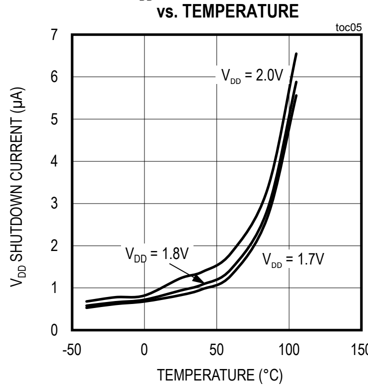
V_{DD} SUPPLY CURRENT vs. SUPPLY VOLTAGE



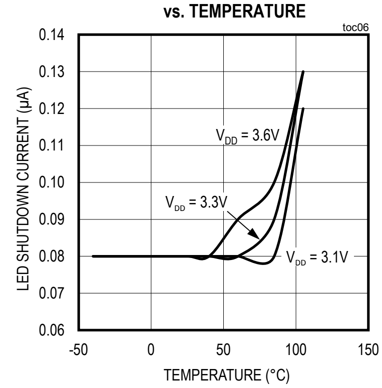
DC COUNTS vs. DISTANCE FOR WHITE HIGH IMPACT STYRENE CARD



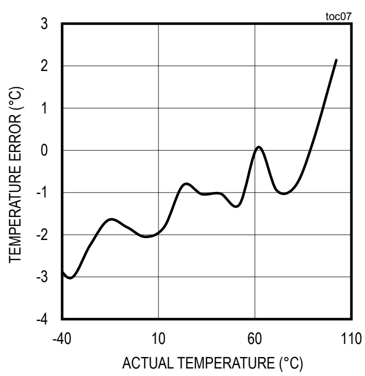
V_{DD} SHUTDOWN CURRENT vs. TEMPERATURE



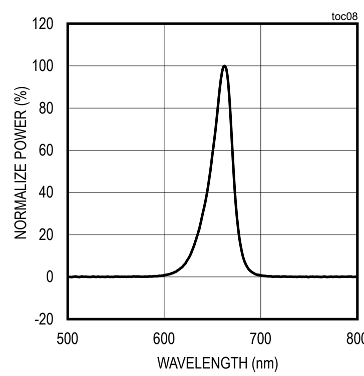
LED SHUTDOWN CURRENT vs. TEMPERATURE



ON-BOARD TEMPERATURE vs. ERROR

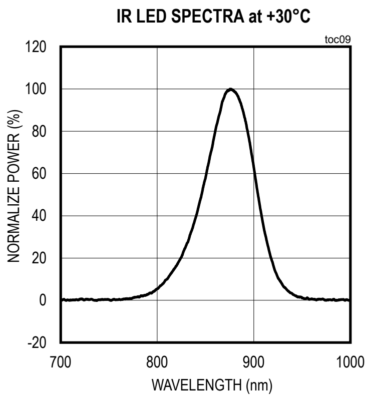


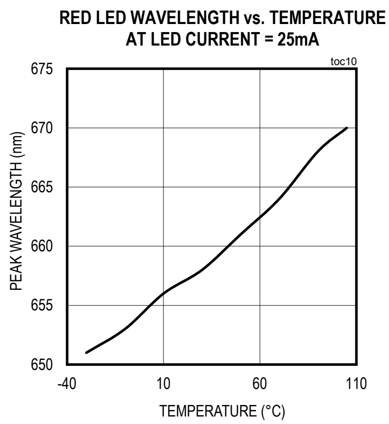
RED LED SPECTRA at +30°C

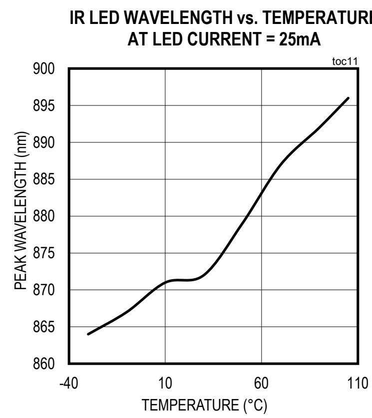


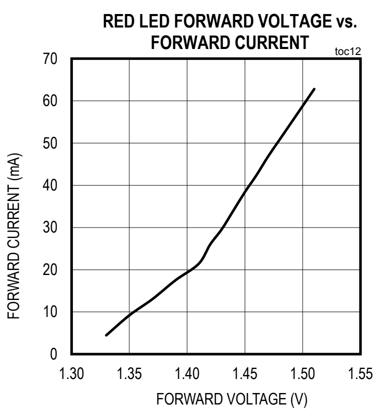
Typical Operating Characteristics (continued)

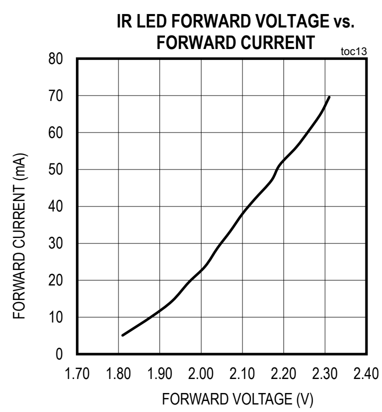
(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, unless otherwise noted.)



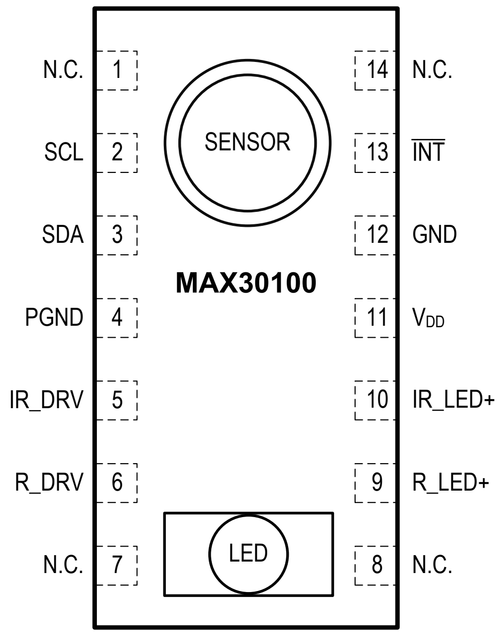








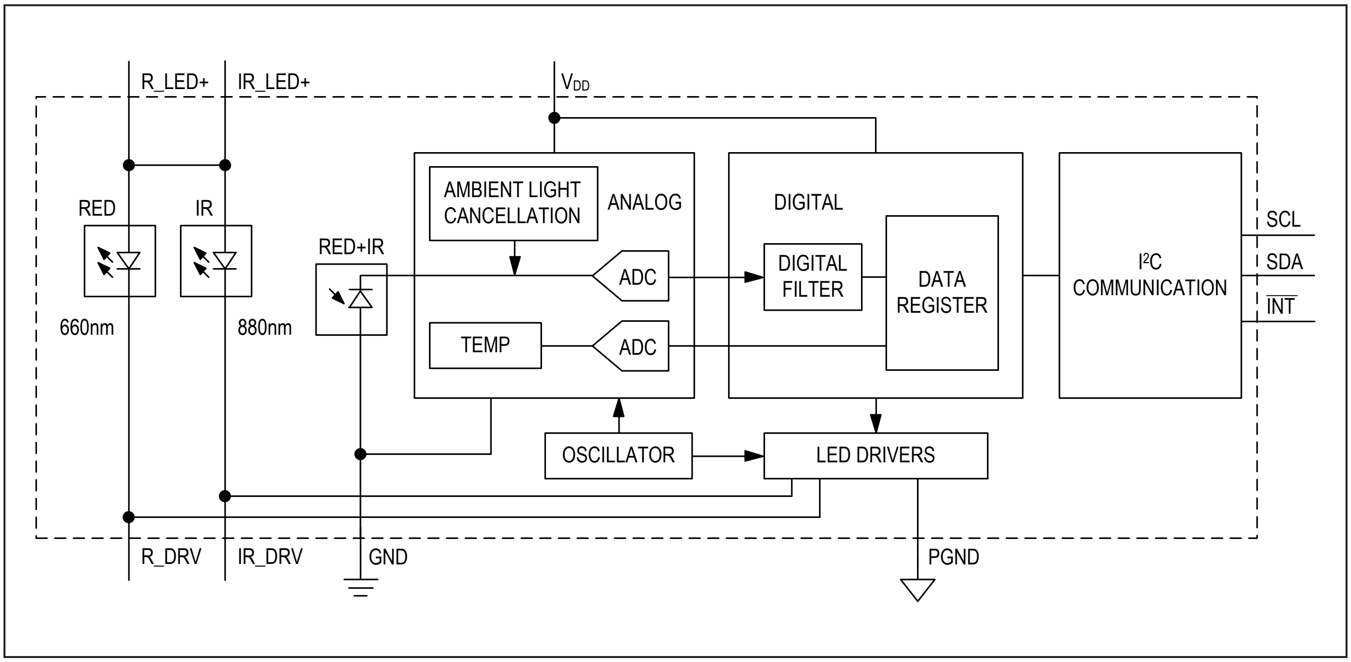
Pin Configuration



Pin Description

PIN	NAME	FUNCTION
1, 7, 8, 14	N.C.	No Connection. Connect to PCB Pad for Mechanical Stability.
2	SCL	I^2C Clock Input
3	SDA	I^2C Clock Data, Bidirectional (Open-Drain)
4	PGND	Power Ground of the LED Driver Blocks
5	IR_DRV	IR LED Cathode and LED Driver Connection Point. Leave floating in circuit.
6	R_DRV	Red LED Cathode and LED Driver Connection Point. Leave floating in circuit.
9	R_LED+	Power Supply (Anode Connection) for Red LED. Bypass to PGND for best performance. Connected to IR_LED+ internally.
10	IR_LED+	Power Supply (Anode Connection) for IR LED. Bypass to PGND for best performance. Connected to R_LED+ internally.
11	V_{DD}	Analog Power Supply Input. Bypass to GND for best performance.
12	GND	Analog Ground
13	$\overline{INT}$	Active-Low Interrupt (Open-Drain)

Functional Diagram



Detailed Description

The MAX30100 is a complete pulse oximetry and heart-rate sensor system solution designed for the demanding requirements of wearable devices. The MAX30100 provides very small total solution size without sacrificing optical or electrical performance. Minimal external hardware components are needed for integration into a wearable device.

The MAX30100 is fully configurable through software registers, and the digital output data is stored in a 16-deep FIFO within the device. The FIFO allows the MAX30100 to be connected to a microcontroller or microprocessor on a shared bus, where the data is not being read continuously from the device's registers.

SpO_2 Subsystem

The SpO_2 subsystem in the MAX30100 is composed of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter.

The SpO_2 ADC is a continuous time oversampling sigma delta converter with up to 16-bit resolution. The ADC output data rate can be programmed from 50Hz to 1kHz. The MAX30100 includes a proprietary discrete time filter to reject 50Hz/60Hz interference and low-frequency residual ambient noise.

Temperature Sensor

The MAX30100 has an on-chip temperature sensor for (optionally) calibrating the temperature dependence of the SpO_2 subsystem.

The SpO_2 algorithm is relatively insensitive to the wavelength of the IR LED, but the red LED's wavelength is critical to correct interpretation of the data. The temperature sensor data can be used to compensate the SpO_2 error with ambient temperature changes.

LED Driver

The MAX30100 integrates red and IR LED drivers to drive LED pulses for SpO_2 and HR measurements. The LED current can be programmed from 0mA to 50mA (typical only) with proper supply voltage. The LED pulse width can be programmed from 200µs to 1.6ms to optimize measurement accuracy and power consumption based on use cases.

Table 1. Register Maps and Descriptions

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
STATUS											
Interrupt Status	A_FULL	TEMP_RDY	HR_RDY	SPO2_RDY				PWR_RDY	0x00	0X00	R
Interrupt Enable	ENB_A_FULL	ENB_TEP_RDY	ENB_HR_RDY	ENB_SO2_RDY					0x01	0X00	R/W
FIFO											
FIFO Write Pointer					FIFO_WR_PTR[3:0]				0x02	0x00	R/W
Over Flow Counter					OVF_COUNTER[3:0]				0x03	0x00	R/W
FIFO Read Pointer					FIFO_RD_PTR[3:0]				0x04	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x05	0x00	R/W
CONFIGURATION											
Mode Configuration	SHDN	RESET			TEMP_EN	MODE[2:0]			0x06	0x00	R/W
SPO2 Configuration		SPO2_HI_RES_EN	RESERVED	SPO2_SR[2:0]			LED_PW[1:0]		0x07	0x00	R/W
RESERVED									0x08	0x00	R/W
LED Configuration	RED_PA[3:0]				IR_PA[3:0]				0x09	0x00	R/W
RESERVED									0x0A – 0x15	0x00	R/W
TEMPERATURE											
Temp_Integer	TINT[7:0]								0x16	0x00	R/W
Temp_Fraction					TFRAC[3:0]				0x17	0x00	R/W
RESERVED									0x8D	0x00	R/W
PART ID											
Revision ID	REV_ID[7:0]								0xFE	0xXX*	R
Part ID	PART_ID[7]								0xFF	0x11	R/W

*XX denotes any 2-digit hexidecimal number (00 to FF). Contact Maxim Integrated for the Revision ID number assigned for your product.

Interrupt Status (0x00)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Status	A_FULL	TEMP_ RDY	HR_RDY	SPO2_ RDY				PWR_ RDY	0x00	0X00	R

There are 5 interrupts and the functionality of each is exactly the same: pulling the active-low interrupt pin into its low state until the interrupt is cleared.

The interrupts are cleared whenever the interrupt status register is read, or when the register that triggered the interrupt is read. For example, if the SpO_2 sensor triggers an interrupt due to finishing a conversion, reading either the FIFO data register or the interrupt register clears the interrupt pin (which returns to its normal high state), and also clears all the bits in the interrupt status register to zero.

Bit 7: FIFO Almost Full Flag (A_FULL)

In SpO_2 and heart-rate modes, this interrupt triggers when the FIFO write pointer is the same as the FIFO read pointer minus one, which means that the FIFO has only one unwritten space left. If the FIFO is not read within the next conversion time, the FIFO becomes full and future data is lost.

Bit 6: Temperature Ready Flag (TEMP_RDY)

When an internal die temperature conversion is finished, this interrupt is triggered so the processor can read the temperature data registers.

Bit 5: Heart Rate Data Ready (HR_RDY)

In heart rate or SPO_2 mode, this interrupt triggers after every data sample is collected. A heart rate data sample consists of one IR data point only. This bit is automatically cleared when the FIFO data register is read.

Bit 4: SpO_2 Data Ready (SPO2_RDY)

In SpO_2 mode, this interrupt triggers after every data sample is collected. An SpO_2 data sample consists of one IR and one red data points. This bit is automatically cleared when the FIFO data register is read.

Bit 3: RESERVED

This bit should be ignored and always be zero in normal operation.

Bit 2: RESERVED

This bit should be ignored and always be zero in normal operation.

Bit 1: RESERVED

This bit should be ignored and always be zero in normal operation.

Bit 0: Power Ready Flag (PWR_RDY)

On power-up or after a brownout condition, when the supply voltage V_{DD} transitions from below the UVLO voltage to above the UVLO voltage, a power-ready interrupt is triggered to signal that the IC is powered up and ready to collect data.

Interrupt Enable (0x01)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Enable	ENB_A_FULL	ENB_TEP_RDY	ENB_HR_RDY	ENB_SO2_RDY					0x01	0X00	R/W

Each source of hardware interrupt, with the exception of power ready, can be disabled in a software register within the MAX30100 IC. The power-ready interrupt cannot be disabled because the digital state of the MAX30100 is reset upon a brownout condition (low power-supply voltage), and the default state is that all the interrupts are disabled. It is important for the system to know that a brownout condition has occurred, and the data within the device is reset as a result.

When an interrupt enable bit is set to zero, the corresponding interrupt appears as 1 in the interrupt status register, but the $\overline{\text{INT}}$ pin is not pulled low.

The four unused bits (B3:B0) should always be set to zero (disabled) for normal operation.

FIFO (0x02–0x05)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
FIFO Write Pointer					FIFO_WR_PTR[3:0]				0x02	0x00	R/W
Over Flow Counter					OVF_COUNTER[3:0]				0x03	0x00	R/W
FIFO Read Pointer					FIFO_RD_PTR[3:0]				0x04	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x05	0x00	R/W

FIFO Write Pointer

The FIFO write pointer points to the location where the MAX30100 writes the next sample. This pointer advances for each sample pushed on to the FIFO. It can also be changed through the I^2C interface when MODE[2:0] is nonzero.

FIFO Overflow Counter

When the FIFO is full, samples are not pushed on to the FIFO, samples are lost. OVF_COUNTER counts the number of samples lost. It saturates at 0xF. When a complete sample is popped from the FIFO (when the read pointer advances), OVF_COUNTER is reset to zero.

FIFO Read Pointer

The FIFO read pointer points to the location from where the processor gets the next sample from the FIFO via the I^2C interface. This advances each time a sample is popped from the FIFO. The processor can also write to this pointer after reading the samples, which would allow rereading samples from the FIFO if there is a data communication error.

FIFO Data

The circular FIFO depth is 16 and can hold up to 16 samples of SpO_2 channel data (Red and IR). The FIFO_DATA register in the I^2C register map points to the next sample to be read from the FIFO. FIFO_RD_PTR points to this sample. Reading FIFO_DATA register does not automatically increment the register address; burst reading this register reads the same address over and over. Each sample is 4 bytes of data, so this register has to be read 4 times to get one sample.

The above registers can all be written and read, but in practice, only the FIFO_RD_PTR register should be written to in operation. The others are automatically incremented or filled with data by the MAX30100. When starting a new SpO_2

or heart-rate conversion, it is recommended to first clear the FIFO_WR_PTR, OVF_COUNTER, and FIFO_RD_PTR registers to all zeros (0x00) to ensure the FIFO is empty and in a known state. When reading the MAX30100 registers in one burst-read I2C transaction, the register address pointer typically increments so that the next byte of data sent is from the next register, etc. The exception to this is the FIFO data register, register 0x05. When reading this register, the address pointer does not increment, but the FIFO_RD_PTR does. So the next byte of data sent will represent the next byte of data available in the FIFO.

Reading from the FIFO

Normally, reading registers from the I2C interface autoincrements the register address pointer, so that all the registers can be read in a burst read without an I2C restart event. In the MAX30100, this holds true for all registers except for the FIFO_DATA register (0x05).

Reading the FIFO_DATA register does not automatically increment the register address; burst reading this register reads the same address over and over. Each sample is 4 bytes of data, so this register has to be read 4 times to get one sample.

The other exception is 0xFF, reading more bytes after the 0xFF register does not advance the address pointer back to 0x00, and the data read is not meaningful.

FIFO Data Structure

The data FIFO consists of a 16-sample memory bank that stores both IR and RED ADC data. Since each sample consists of one IR word and one RED word, there are 4 bytes of data for each sample, and therefore, 64 total bytes of data can be stored in the FIFO. Figure 2 shows the structure of the FIFO graphically.

The FIFO data is left-justified as shown in Table 1; i.e. the MSB bit is always in the bit 15 position regardless of ADC resolution.

Each data sample consists of an IR and a red data word (2 registers), so to read one sample requires 4 I2C byte reads in a row. The FIFO read pointer is automatically incremented after each 4-byte sample is read.

In heart-rate only mode, the 3rd and 4th bytes of each sample return zeros, but the basic structure of the FIFO remains the same.

Write/Read Pointers

Table 2. FIFO Data

ADC RESOLUTION	IR [15]	IR [14]	IR [13]	IR [12]	IR [11]	IR [10]	IR [9]	IR [8]	IR [7]	IR [6]	IR [5]	IR [4]	IR [3]	IR [2]	IR [1]	IR [0]
16-bit																
14-bit																
12-bit																
10-bit																

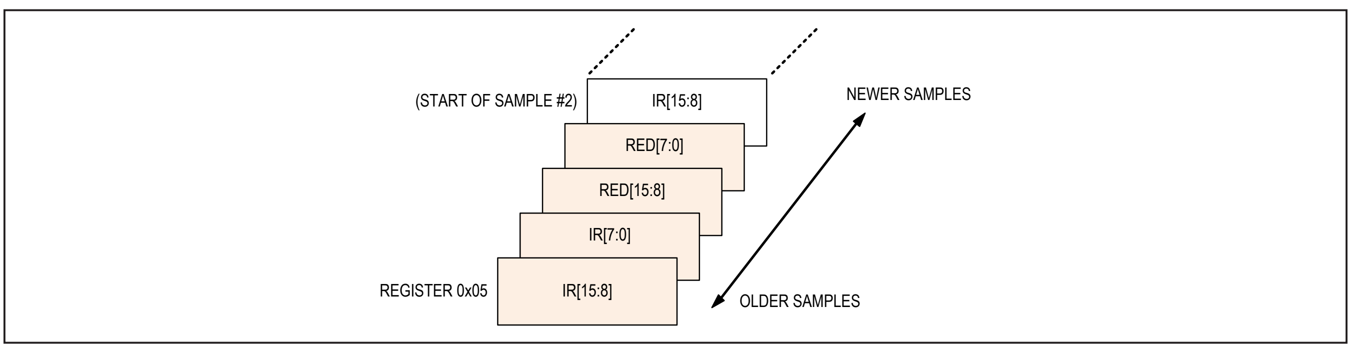


Figure 2. Graphical Representation of the FIFO Data Register

The locations to store new data, and the read pointer for reading data, are used to control the flow of data in the FIFO. The write pointer increments every time a new sample is added to the FIFO. The read pointer is incremented automatically every time a sample is read from the FIFO. To reread a sample from the FIFO, decrement its value by one and read the data register again.

The SpO_2 write/read pointers should be cleared (back to 0x0) upon entering SpO_2 mode or heart-rate mode, so that there is no old data represented in the FIFO. The pointers are not automatically cleared when changing modes, but they are cleared if V_{DD} is power cycled so that the V_{DD} voltage drops below its UVLO voltage.

Pseudo-Code Example of Reading Data from FIFO

First transaction: Get the FIFO_WR_PTR:

```
START;
Send device address + write mode
Send address of FIFO_WR_PTR;
REPEATED_START;
Send device address + read mode
Read FIFO_WR_PTR;
STOP;
```

The central processor evaluates the number of samples to be read from the FIFO:

```
NUM_AVAILABLE_SAMPLES = FIFO_WR_PTR – FIFO_RD_PTR
(Note: pointer wrap around should be taken into account)
NUM_SAMPLES_TO_READ = < less than or equal to NUM_AVAILABLE_SAMPLES >
```

Second transaction: Read `NUM_SAMPLES_TO_READ` **samples from the FIFO**:

```
START;
Send device address + write mode
Send address of FIFO_DATA;
REPEATED_START;
Send device address + read mode
for (i = 0; i < NUM_SAMPLES_TO_READ; i++) {
Read FIFO_DATA;
Save IR[15:8];
Read FIFO_DATA;
Save IR[7:0];
Read FIFO_DATA;
Save R[15:8];
Read FIFO_DATA;
Save R[7:0];
      }
STOP;
```

Third transaction: Write to FIFO_RD_PTR register. If the second transaction was successful, FIFO_RD_PTR points to the next sample in the FIFO, and this third transaction is not necessary. Otherwise, the processor updates the FIFO_RD_PTR appropriately, so that the samples are reread.

```
START;
Send device address + write mode
Send address of FIFO_RD_PTR;
Write FIFO_RD_PTR;
STOP;
```

Mode Configuration (0x06)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Mode Configu-ration	SHDN	RESET			TEMP_EN	MODE[2:0]			0x06	0x00	R/W

Bit 7: Shutdown Control (SHDN)

The part can be put into a power-save mode by setting this bit to one. While in power-save mode, all registers retain their values, and write/read operations function as normal. All interrupts are cleared to zero in this mode.

Bit 6: Reset Control (RESET)

When the RESET bit is set to one, all configuration, threshold, and data registers are reset to their power-on-state. The only exception is writing both RESET and TEMP_EN bits to one at the same time since temperature data registers 0x16 and 0x17 are not cleared. The RESET bit is cleared automatically back to zero after the reset sequence is completed.

Bit 3: Temperature Enable (TEMP_EN)

This is a self-clearing bit which, when set, initiates a single temperature reading from the temperature sensor. This bit is cleared automatically back to zero at the conclusion of the temperature reading when the bit is set to one in heart rate or SpO_2 mode.

Bits 2:0: Mode Control

These bits set the operating state of the MAX30100. Changing modes does not change any other setting, nor does it erase any previously stored data inside the data registers.

Table 3. Mode Control

MODE[2:0]	MODE
000	Unused
001	Reserved (Do not use)
010	HR only enabled
011	SPO_2 enabled
100–111	Unused

SpO2 Configuration (0x07)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
SPO_2 Configuration		SPO2_HI_ RES_EN	Reserved	SPO2_SR[2:0]			LED_PW[1:0]		0x07	0x00	R/W

Bit 6: SpO_2 High Resolution Enable (SPO2_HI_RES_EN)

Set this bit high. The SpO_2 ADC resolution is 16-bit with 1.6ms LED pulse width.

Bit 5: Reserved. Set low (default).

Bit 4:2: SpO_2 Sample Rate Control

These bits define the effective sampling rate, with one sample consisting of one IR pulse/conversion and one RED pulse/ conversion.

The sample rate and pulse width are related, in that the sample rate sets an upper bound on the pulse width time. If the user selects a sample rate that is too high for the selected LED_PW setting, the highest possible sample rate will instead be programmed into the register.

Bits 1:0: LED Pulse Width Control

These bits set the LED pulse width (the IR and RED have the same pulse width), and therefore, indirectly set the integration time of the ADC in each sample. The ADC resolution is directly related to the integration time.

Table 4. SpO_2 Sample Rate Control

SPO2_SR[2:0]	SAMPLES (PER SECOND)
000	50
001	100
010	167
011	200
100	400
101	600
110	800
111	1000

Table 5. LED Pulse Width Control

LED_PW[1:0]	PULSE WIDTH (µs)	ADC RESOLUTION (BITS)
00	200	13
01	400	14
10	800	15
11	1600	16

LED Configuration (0x09)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
LED Configuration	RED_PA[3:0]				IR_PA[3:0]				0x09	0x00	R/W

Bits 7:4: Red LED Current Control

These bits set the current level of the Red LED as in Table 6.

Bits 3:0: IR LED Current Control

These bits set the current level of the IR LED as in Table 6.

Table 6. LED Current Control

Red_PA[3:0] OR IR_PA[3:0]	TYPICAL LED CURRENT (mA)*
0000	0.0
0001	4.4
0010	7.6
0011	11.0
0100	14.2
0101	17.4
0110	20.8
0111	24.0
1000	27.1
1001	30.6
1010	33.8
1011	37.0
1100	40.2
1101	43.6
1110	46.8
1111	50.0

*Actual measured LED current for each part can vary widely due to the proprietary trim methodology.

Temperature Data (0x16–0x17)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Temp_Integer	TINT[7:0]								0x16	0x00	R/W
Temp_Fraction					TFRAC[3:0]				0x17	0x00	R/W

Temperature Integer

The on-board temperature ADC output is split into two registers, one to store the integer temperature and one to store the fraction. Both should be read when reading the temperature data, and the following equation shows how to add the two registers together:

$$T_{MEASURED} = T_{INTEGER} + T_{FRACTION}$$

This register stores the integer temperature data in two's complement format, where each bit corresponds to degree Celsius.

Table 7. Temperature Integer

REGISTER VALUE (hex)	TEMPERATURE (°C)
0x00	0
0x00	+1
...	...
0x7E	+126
0x7F	+127
0x80	-128
0x81	-127
...	...
0xFE	-2
0xFF	-1

Temperature Fraction

This register stores the fractional temperature data in increments of 0.0625°C (1/16th of a degree).

If this fractional temperature is paired with a negative integer, it still adds as a positive fractional value (e.g., -128°C + 0.5°C = -127.5°C).

Applications Information

Sampling Rate and Performance

The MAX30100 ADC is a 16-bit sigma delta converter. The ADC sampling rate can be configured from 50sps to 1ksps. The maximum sample rate for the ADC depends on the selected pulse width, which in turn, determines the ADC resolution. For instance, if the pulse width is set to 200µs, then the ADC resolution is 13 bits and all sample rates from 50sps to 1ksps are selectable. However, if the pulse width is set to 1600µs, then only sample rates of 100sps and 50sps can be set. The allowed sample rates for both SpO_2 and HR mode are summarized in Table 8 and Table 9.

Power Considerations

The LEDs in MAX30100 are pulsed with a low duty cycle for power savings, and the pulsed currents can cause ripples in the LED power supply. To ensure these pulses do not translate into optical noise at the LED outputs, the power supply must be designed to handle peak LED current. Ensure that the resistance and inductance from the power supply (battery, DC/DC converter, or LDO) to the device LED+ pins is much smaller than 1Ω, and that there is at least 1µF of power-supply bypass capacitance to a low impedance ground plane. The decoupling capacitor should be located physically as close as possible to the MAX30100 device.

In the heart-rate only mode, the red LED is inactive, and only the IR LED is used to capture optical data and determine the heart rate. This mode allows power savings due to the red LED being off; in addition, the IR_LED+ power supply can be reduced to save power because the forward voltage of the IR LED is significantly less than that of the red LED.

The average I_{DD} and LED current as function of pulse width and sampling rate is summarized in Table 10 to Table 13.

Table 8. SpO_2 Mode (Allowed Settings)

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	O	O	O	O
100	O	O	O	O
167	O	O	O	
200	O	O	O	
400	O	O		
600	O			
800	O			
1000	O			
Resolution (bits)	13	14	15	16

Table 9. Heart-Rate Mode (Allowed Settings)

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	O	O	O	O
100	O	O	O	O
167	O	O	O	
200	O	O	O	
400	O	O		
600	O	O		
800	O	O		
1000	O	O		
Resolution (bits)	13	14	15	16

Table 10. SpO_2 Mode: Average IDD Current (µA) R_PA = 0x3, IR_PA = 0x3

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	628	650	695	782
100	649	691	776	942
167	678	748	887	
200	692	775	940	
400	779	944		
600	865			
800	952			
1000	1037			

Table 11. SpO_2 Mode: Average LED Current (mA) R_PA = 0x3, IR_PA = 0x3

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	0.667	1.332	2.627	5.172
100	1.26	2.516	4.96	9.766
167	2.076	4.145	8.173	
200	2.491	4.93	9.687	
400	4.898	9.765		
600	7.319			
800	9.756			
1000	12.17			

Table 12. Heart-Rate Mode: Average IDD Current (µA) IR_PA = 0x3

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	608	616	633	667
100	617	634	669	740
167	628	658	716	831
200	635	670	739	876
400	671	740	878	
600	707	810		
800	743	881		
1000	779	951		

Table 13. Heart-Rate Mode: Average LED Current (mA) IR_PA = 0x3

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	0.256	0.511	1.020	2.040
100	0.512	1.022	2.040	4.077
167	0.854	1.705	3.404	6.795
200	1.023	2.041	4.074	8.130
400	2.042	4.074	8.123	
600	3.054	6.089		
800	4.070	8.109		
1000	5.079	10.11		

Hardware Interrupt

The active-low interrupt pin pulls low when an interrupt is triggered. The pin is open-drain and requires a pullup resistor or current source to an external voltage supply (up to +5V from GND). The interrupt pin is not designed to sink large currents, so the pullup resistor value should be large, such as 4.7kΩ.

The internal FIFO stores up to 16 samples, so that the system processor does not need to read the data after every sample. Temperature data may be needed to properly interpret SpO_2 data, but the temperature does not need to be sampled very often—once a second or every few seconds should be sufficient. In heart-rate mode temperature information is not necessary.

Table 14. Red LED Current Settings vs. LED Temperature Rise

RED LED CURRENT SETTING	RED LED DUTY CYCLE (% OF LED PULSE WIDTH TO SAMPLE TIME)	ESTIMATED TEMPERATURE RISE (ADD TO TEMPERATURE SENSOR MEASUREMENT) (°C)
0001 (3.1mA)	8	0.1
1111 (35mA)	8	2
0001 (3.1mA)	16	0.3
1111 (35mA)	16	4
0001 (3.1mA)	32	0.6
1111 (35mA)	32	8

Timing for Measurements and Data Collection

Timing in SpO_2 Mode

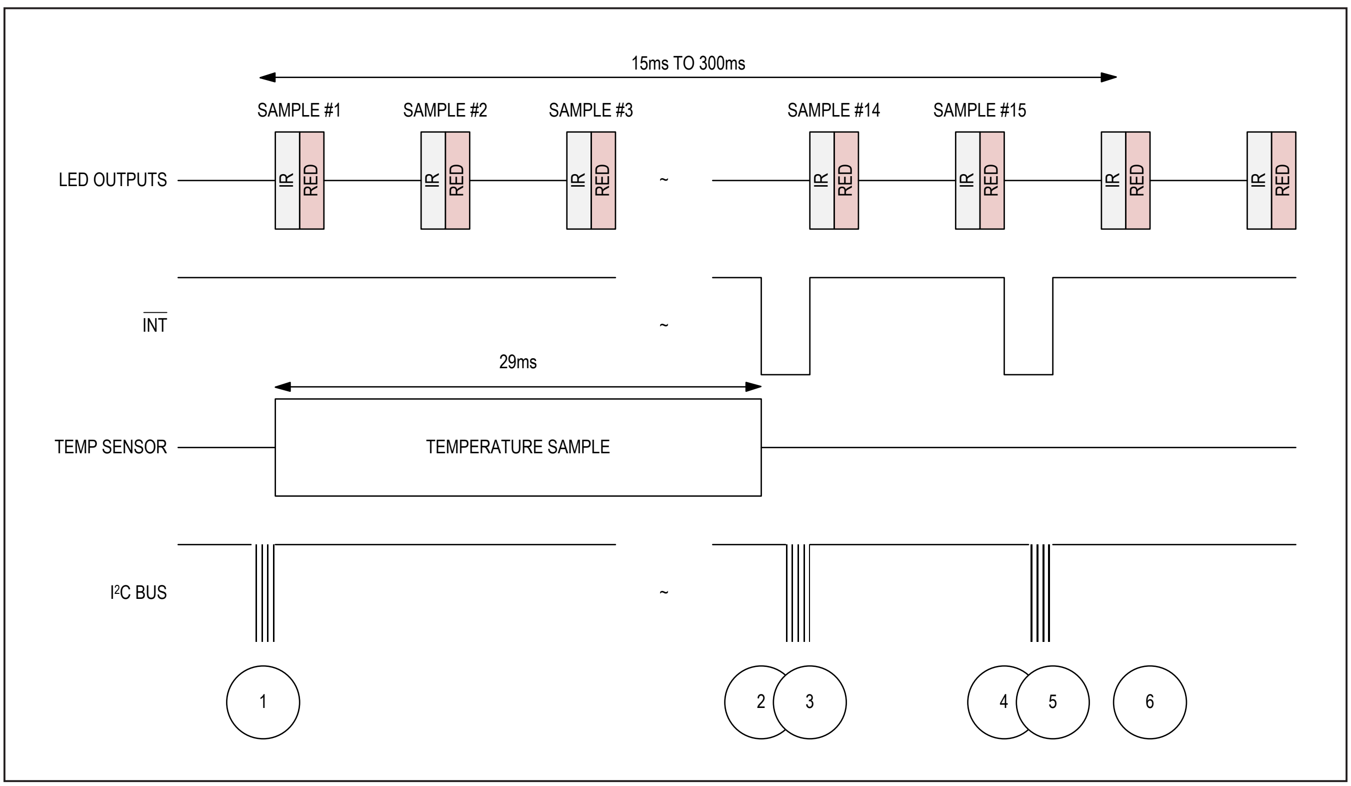


Figure 3. Timing for Data Acquisition and Communication When in SpO_2 Mode

Table 15. Events Sequence for Figure 3 in SpO_2 Mode

EVENT	DESCRIPTION	COMMENTS
1	Enter into SpO_2 mode. Initiate a temperature measurement.	I^2C Write Command Sets MODE[2:0] = 0x03. At the same time, set the TEMP_EN bit to initiate a single temperature measurement. Mask the SPO2_RDY Interrupt.
2	Temperature measurement complete, interrupt generated	TEMP_RDY interrupt triggers, alerting the central processor to read the data.
3	Temp data is read, interrupt cleared	
4	FIFO is almost full, interrupt generated	Interrupt is generated when the FIFO has only one empty space left.
5	FIFO data is read, interrupt cleared	
6	Next sample is stored	New sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Timing in Heart-Rate Mode

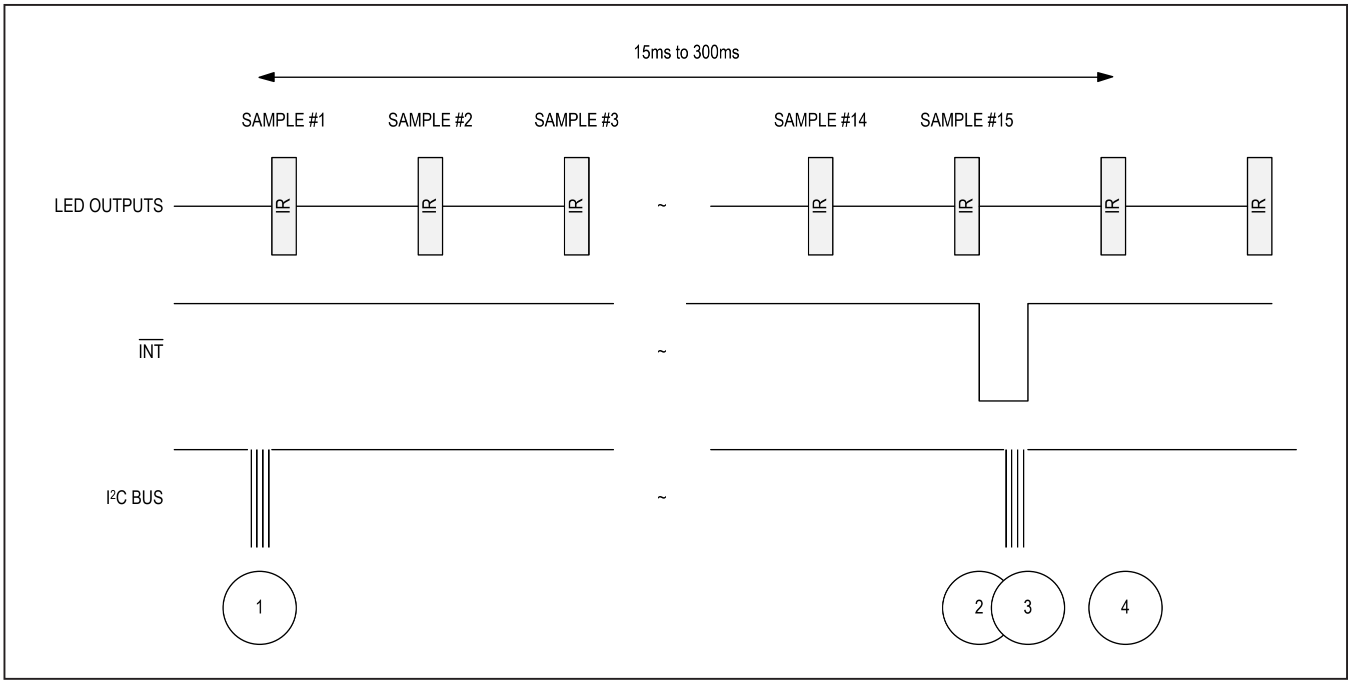


Figure 4. Timing for Data Acquisition and Communication When in Heart Rate Mode

Table 16. Events Sequence for Figure 4 in Heart-Rate Mode

EVENT	DESCRIPTION	COMMENTS
1	Enter into heart rate mode	I²C Write Command Sets MODE[2:0] = 0x02. Mask the HR_RDY interrupt.
2	FIFO is almost full, interrupt generated	Interrupt is generated when the FIFO has only one empty space left.
3	FIFO data is read, interrupt cleared	
4	Next sample is stored	New sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Power Sequencing and Requirements

Power-Up Sequencing

Figure 5 shows the recommended power-up sequence for the MAX30100.

It is recommended to power the V_{DD} supply first, before the LED power supplies (R_LED+, IR_LED+). The interrupt and I²C pins can be pulled up to an external voltage even when the power supplies are not powered up.

After the power is established, an interrupt occurs to alert the system that the MAX30100 is ready for operation. Reading the I²C interrupt register clears the interrupt, as shown in Figure 5.

Power-Down Sequencing

The MAX30100 is designed to be tolerant of any power-supply sequencing on power-down.

I²C Interface

The MAX30100 features an I²C/SMBus-compatible, 2-wire serial interface consisting of a serial data line (SDA) and a serial clock line (SCL). SDA and SCL facilitate communication between the MAX30100 and the master at clock rates up to 400kHz. Figure 1 shows the 2-wire interface timing diagram. The master generates SCL and initiates data transfer on the bus. The master device writes data to the MAX30100 by transmitting the proper slave address followed by data. Each transmit sequence is framed by a START (S) or REPEATED START (Sr) condition and a STOP (P) condition. Each word transmitted to the MAX30100 is 8 bits long and is followed by an acknowledge clock pulse. A master reading data from the MAX30100 transmits the proper slave address followed by a series of nine SCL pulses.

The MAX30100 transmits data on SDA in sync with the master-generated SCL pulses. The master acknowledges receipt of each byte of data. Each read sequence is framed by a START (S) or REPEATED START (Sr) condition, a not acknowledge, and a STOP (P) condition. SDA operates as both an input and an open-drain output. A pullup resistor, typically greater than 500Ω, is required on SDA. SCL operates only as an input. A pullup resistor, typically greater than 500Ω, is required on SCL if there are multiple masters on the bus, or if the single master has an open-drain SCL output.

Bit Transfer

One data bit is transferred during each SCL cycle. The data on SDA must remain stable during the high period of the SCL pulse. Changes in SDA while SCL is high are control signals. See the *START and STOP Conditions* section.

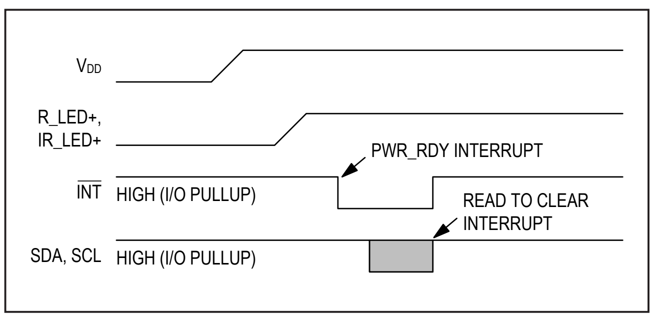


Figure 5. Power-Up Sequence of the Power-Supply Rails

START and STOP Conditions

SDA and SCL idle high when the bus is not in use. A master initiates communication by issuing a START condition. A START condition is a high-to-low transition on SDA with SCL high. A STOP condition is a low-to-high transition on SDA while SCL is high (Figure 6). A START condition from the master signals the beginning of a transmission to the MAX30100. The master terminates transmission, and frees the bus, by issuing a STOP condition. The bus remains active if a REPEATED START condition is generated instead of a STOP condition.

Early STOP Conditions

The MAX30100 recognizes a STOP condition at any point during data transmission except if the STOP condition occurs in the same high pulse as a START condition. For proper operation, do not send a STOP condition during the same SCL high pulse as the START condition.

Slave Address

A bus master initiates communication with a slave device by issuing a START condition followed by the 7-bit slave ID. When idle, the MAX30100 waits for a START condition followed by its slave ID. The serial interface compares each slave ID bit by bit, allowing the interface to power down and disconnect from SCL immediately if an incorrect slave ID is detected. After recognizing a START condition followed by the correct slave ID, the MAX30100 is ready to accept or send data. The LSB of the slave ID word is the Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$) bit. R/$\overline{\text{W}}$ indicates whether the master is writing to or reading data from the MAX30100. R/$\overline{\text{W}}$ = 0 selects a write condition, R/$\overline{\text{W}}$ = 1 selects a read condition). After receiving the proper slave ID, the MAX30100 issues an ACK by pulling SDA low for one clock cycle.

The MAX30100 slave ID consists of seven fixed bits, B7–B1 (set to 0b1010111). The most significant slave ID bit (B7) is transmitted first, followed by the remaining bits. Table 18 shows the possible slave IDs of the device.

Acknowledge

The acknowledge bit (ACK) is a clocked 9th bit that the MAX30100 uses to handshake receipt each byte of data when in write mode (Figure 7). The MAX30100 pulls down SDA during the entire master-generated 9th clock pulse if the previous byte is successfully received. Monitoring ACK allows for detection of unsuccessful data transfers. An unsuccessful data transfer occurs if a receiving device is busy or if a system fault has occurred. In the event of an unsuccessful data transfer, the bus master will retry communication. The master pulls down SDA during the 9th clock cycle to acknowledge receipt of data when the MAX30100 is in read mode. An acknowledge is sent by the master after each read byte to allow data transfer to continue. A not-acknowledge is sent when the master reads the final byte of data from the MAX30100, followed by a STOP condition.

Table 17. Slave ID Description

B7	B6	B5	B4	B3	B2	B1	B0	WRITE ADDRESS	READ ADDRESS
1	0	1	0	1	1	1	R/$\overline{\text{W}}$	0xAE	0xAF

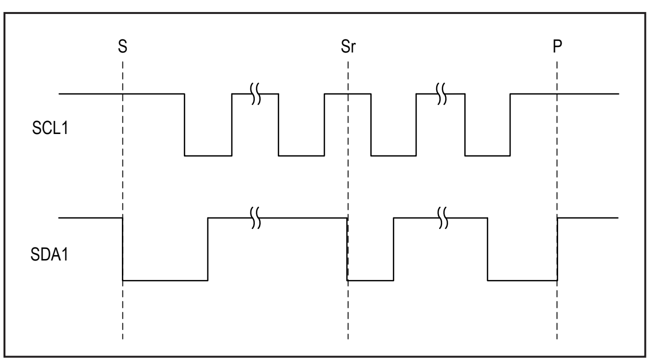


Figure 6. START, STOP, and REPEATED START Conditions

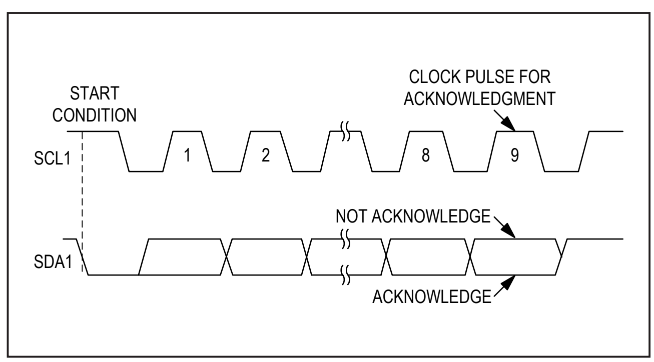


Figure 7. Acknowledge

Write Data Format

For the write operation, send the slave ID as the first byte followed by the register address byte and then one or more data bytes. The register address pointer increments automatically after each byte of data received. For example, the entire register bank can be written by at one time. Terminate the data transfer with a STOP condition. The write operation is shown in Figure 8.

The internal register address pointer increments automatically, so writing additional data bytes fill the data registers in order.

Read Data Format

For the read operation, two I2C operations must be performed. First, the slave ID byte is sent followed by the I2C register that you wish to read. Then a REPEATED START (Sr) condition is sent, followed by the read slave ID. The MAX30100 then begins sending data beginning with the register selected in the first operation. The read pointer increments automatically, so the MAX30100 continues sending data from additional registers in sequential order until a STOP (P) condition is received. The exception to this is the FIFO_DATA register, at which the read pointer no longer increments when reading additional bytes. To read the next register after FIFO_DATA, an I2C write command is necessary to change the location of the read pointer.

An initial write operation is required to send the read register address.

Data is sent from registers in sequential order, starting from the register selected in the initial I2C write operation. If the FIFO_DATA register is read, the read pointer does not automatically increment, and subsequent bytes of data contain the contents of the FIFO.

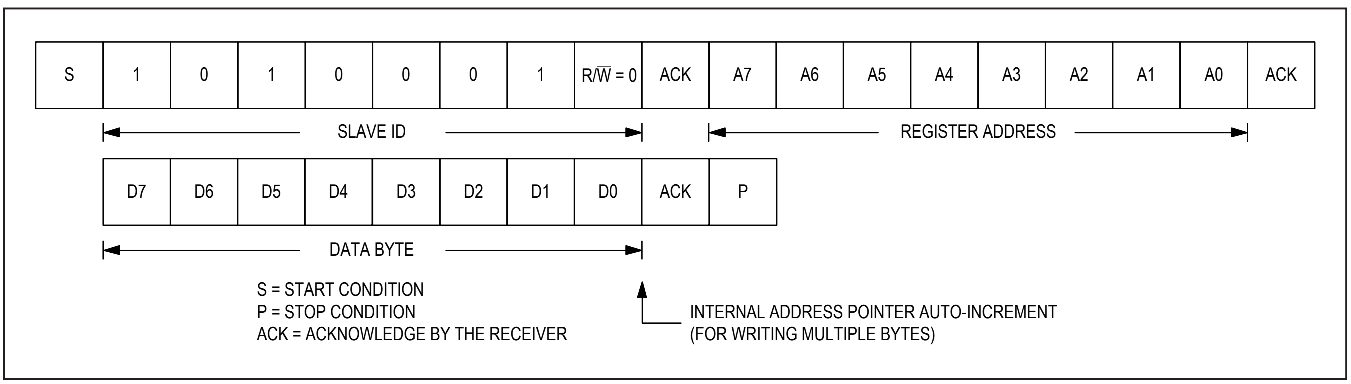


Figure 8. Writing One Data Byte to the MAX30100

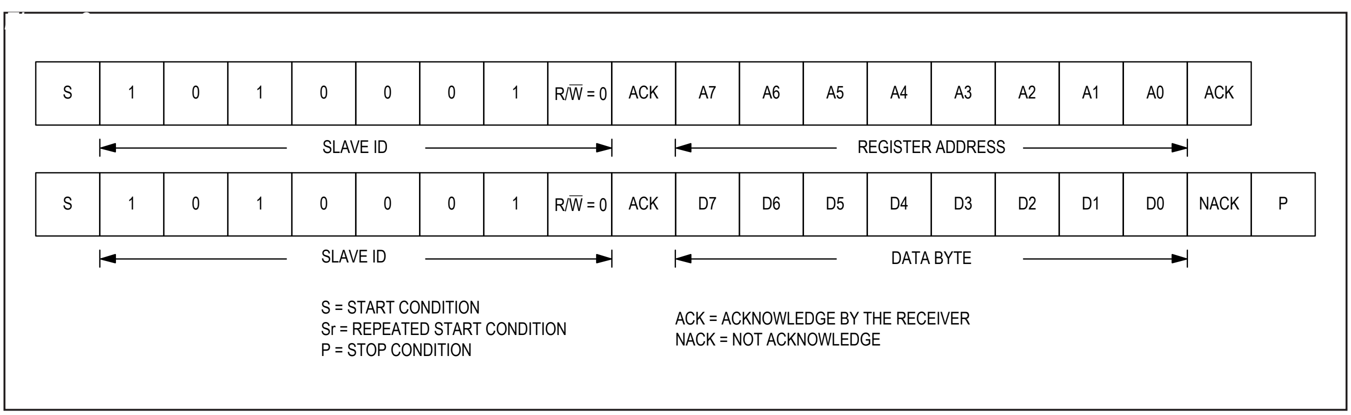


Figure 9. Reading One Byte of Data from the MAX30100

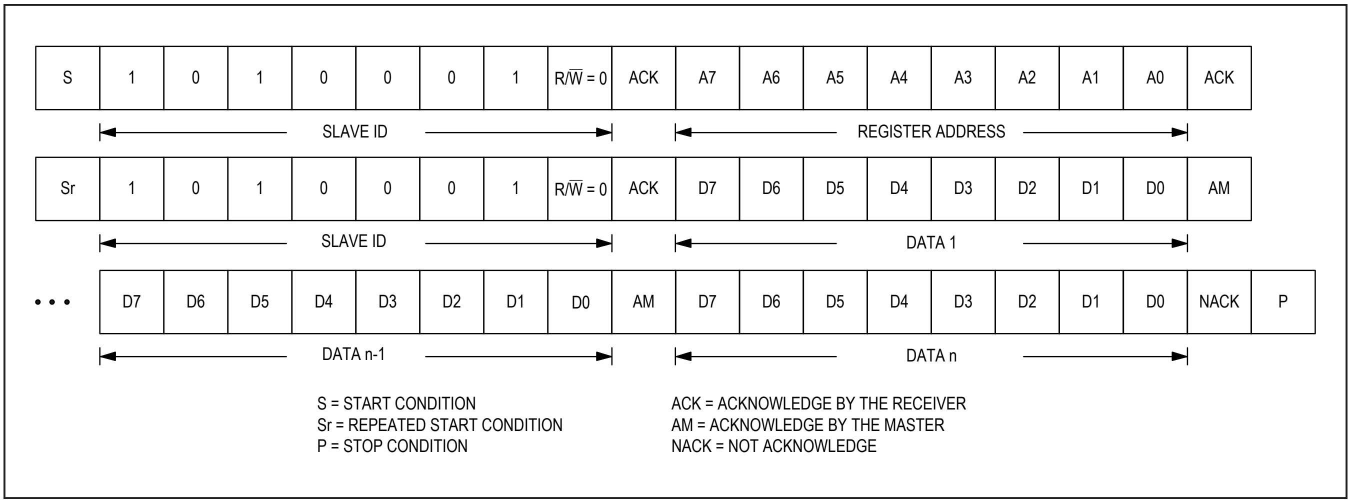
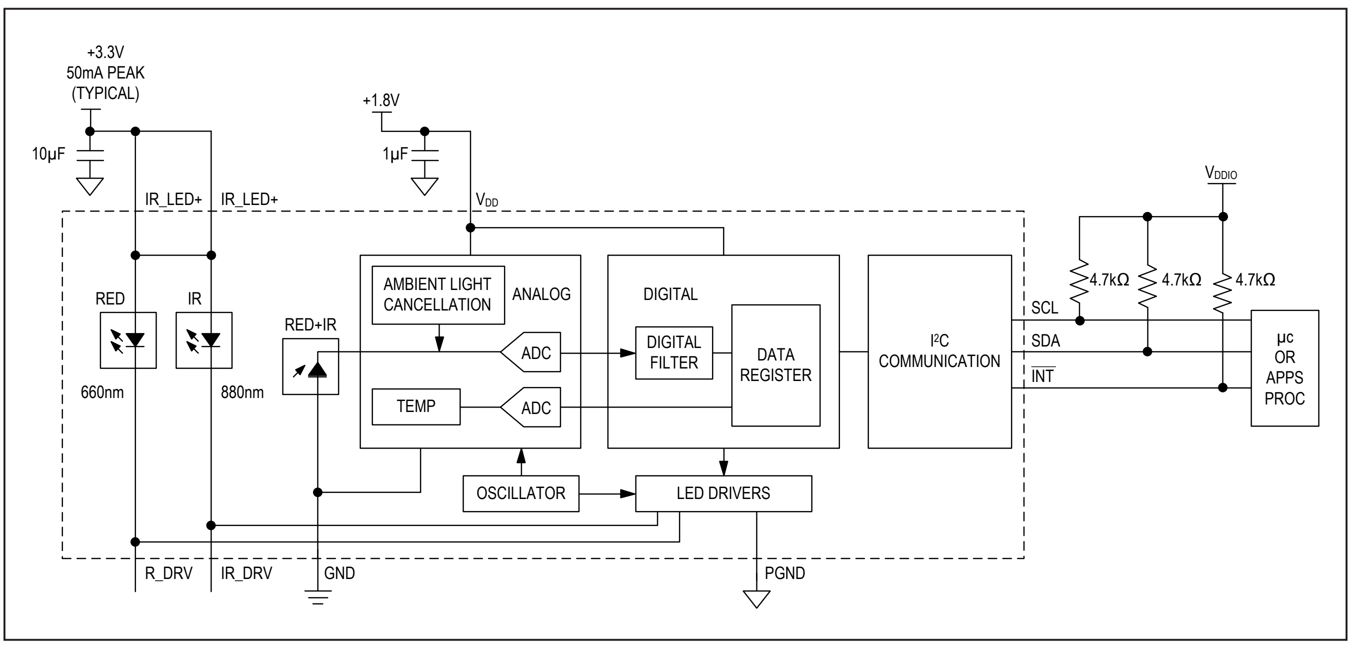


Figure 10. Reading Multiple Bytes of Data from the MAX30100

Typical Application Circuit



Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX30100EFD+	-40°C to +85°C	14 OESIP (0.8mm pitch)

+Denotes a lead(Pb)-free/RoHS-compliant package.

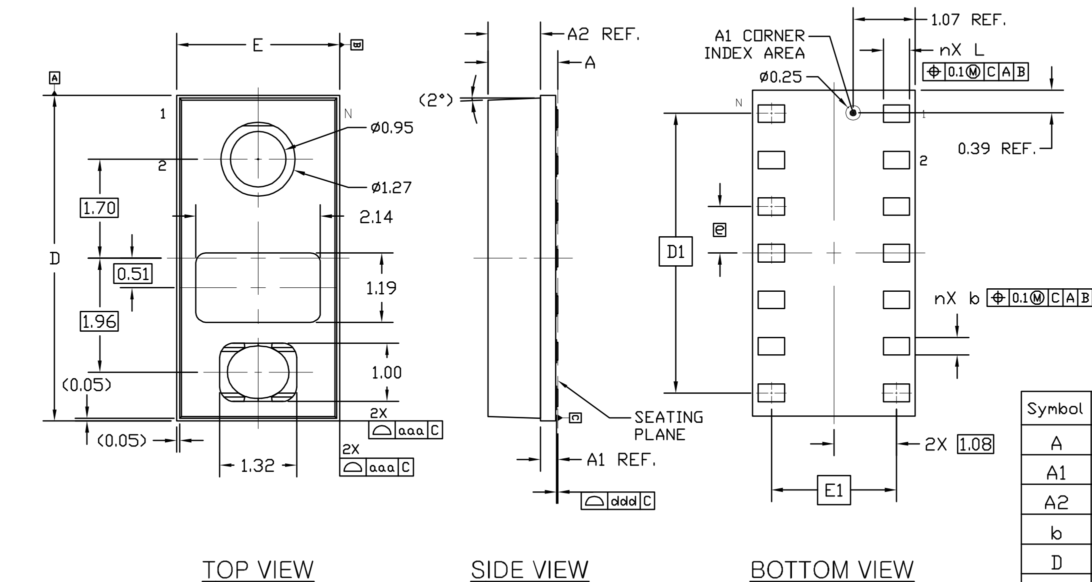
Chip Information

PROCESS: BiCMOS

Package Information

For the latest package outline information and land patterns (footprints), go to **www.maximintegrated.com/packages**. Note that a “+”, “#”, or “-” in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

PACKAGE TYPE	PACKAGE CODE	OUTLINE NO.	LAND PATTERN NO.
14 OESIP	F142D5+2	21-0880	90-0461



Symbol	MIN	NOM	MAX
A	1.1	1.2	1.3
A1	0.3 REF.		
A2	0.9 REF.		
b	0.25	0.30	0.35
D	5.5	5.6	5.7
D1	4.8 BSC		
E	2.7	2.8	2.9
E1	2.15 BSC		
e	0.8 BSC		
L	0.40	0.45	0.50
N	14		
aaa	0.1		
ddd	0.08		

NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS. ANGLES ARE IN DEGREES.
2. DIMENSIONING & TOLERANCING CONFORM TO ASME Y14.5M–1994.
3. N IS THE TOTAL NUMBER OF TERMINALS.
4. MATERIAL MUST BE COMPLIANT WITH MAXIM SPECIFICATION 10-0131 FOR SUBSTANCE CONTENT, MUST BE Eu ROHS COMPLIANT WITHOUT EXEMPTION AND PB-FREE.
5. ALL DIMENSIONS APPLY TO Pb FREE (+) PARTS.
6. PACKAGE CODE: F142D5+2

maxim integrated

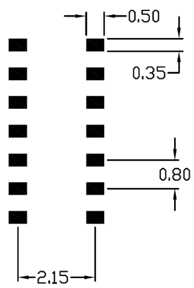
TITLE: PACKAGE OUTLINE, 14L OLGA, 2.8x5.6x1.2mm

APPROVAL	DOCUMENT CONTROL NO.	REV.	
	21-0880	B	1/1

–DRAWING NOT TO SCALE–

Package Information (continued)

For the latest package outline information and land patterns (footprints), go to **www.maximintegrated.com/packages**. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.



NOTES:
1. REFERENCE PKG. OUTLINE: 21-0880.
2. LAND PATTERN COMPLIES TO: IPC7351A.
3. TOLERANCE: +/- 0.02 MM.
4. ALL DIMENSIONS APPLY TO PbFREE (+) PKG. CODE ONLY
5. ALL DIMENSIONS IN MM.

-DRAWING NOT TO SCALE-

This document (including dimensions, notes & specs) is a recommendaton based on typical circuit board manufacturing parameters. Since land pattern design depend on many factors unknown to Maxim (eg. user's board manufacturing specs), user must determine suitability for use. This document is subject to change without notice. Contact technical support at http://www.maxim-ic.com/support for further questions.

TITLE:	PACKAGE LAND PATTERN, [F142D5+2] OLGA
APPROVAL	
DOCUMENT CONTROL NO.	90-0461
REV.	A
	1/1

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	9/14	Initial release	—

DATASHEET

Raspberry Pi 4 Model B

Release 1

June 2019

Table 1: Release History

Release	Date	Description
1	21/06/2019	First release

The latest release of this document can be found at https://www.raspberrypi.org

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with siginficantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is avaiable with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

3 Mechanical Specification

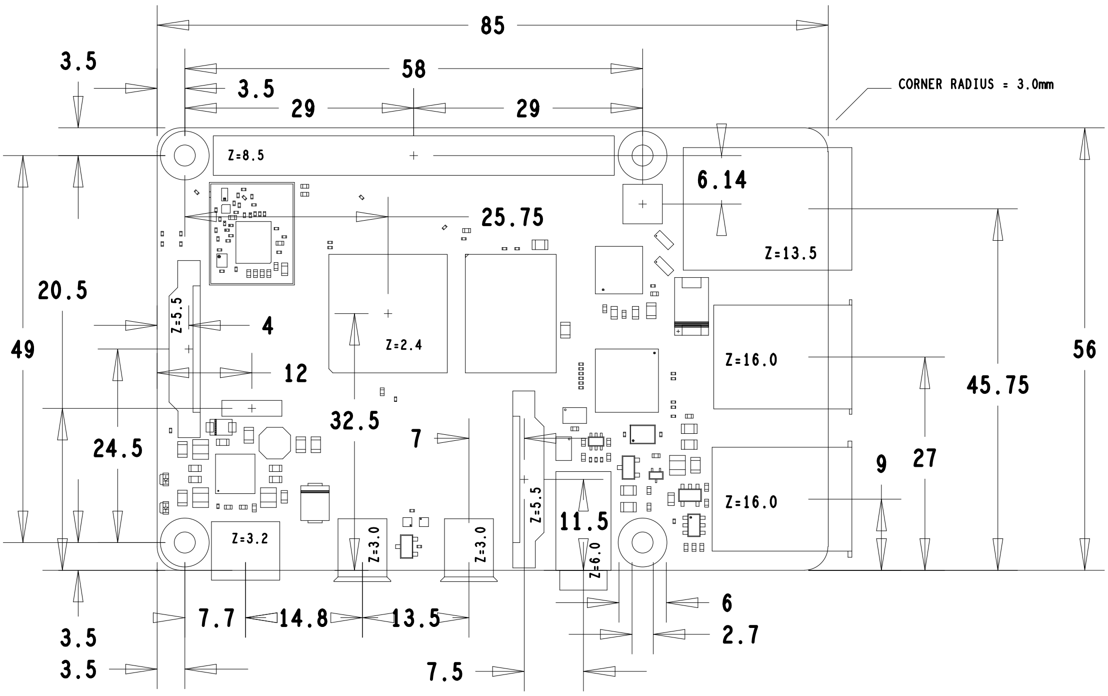


Figure 1: Mechanical Dimensions

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

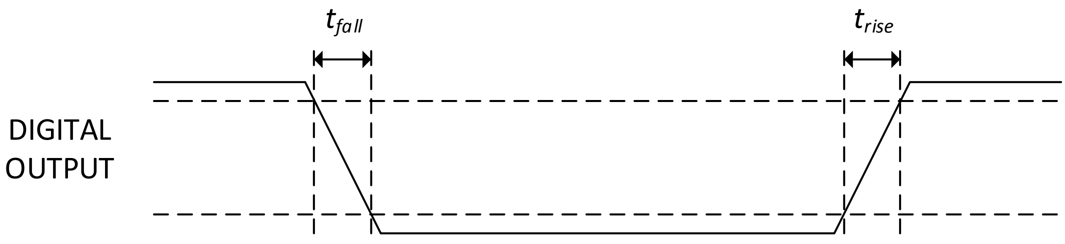


Figure 2: Digital IO Characteristics

4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

5 Peripherals

5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

5.1.1 GPIO Pin Assignments

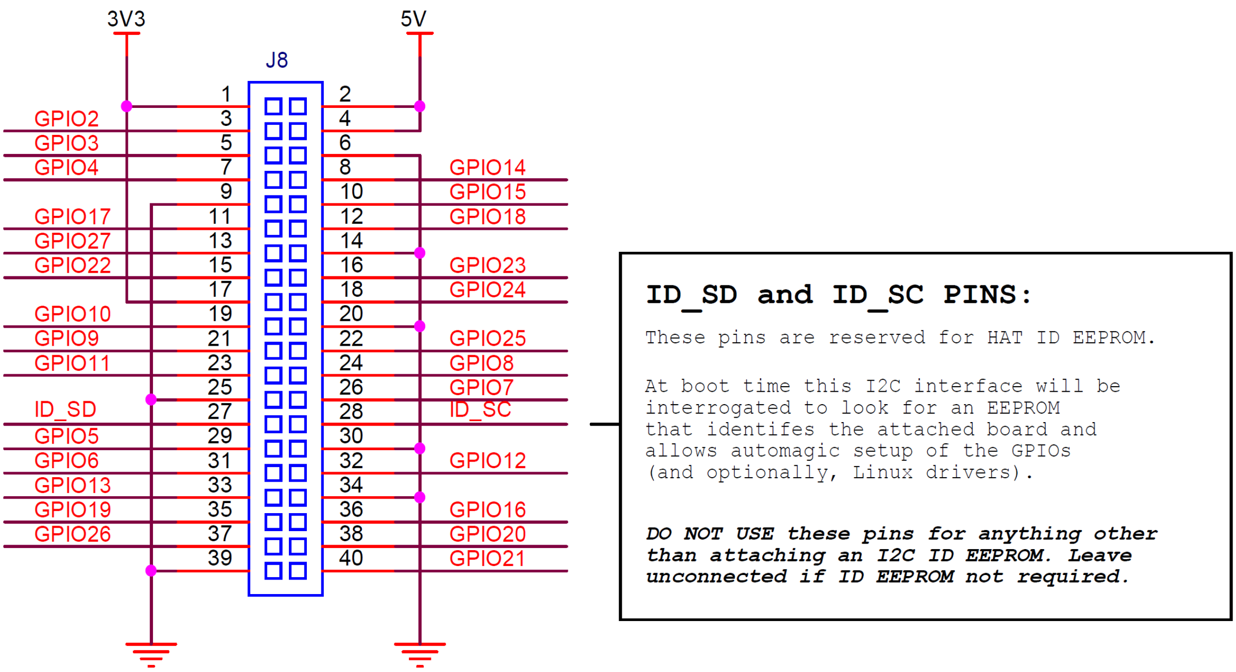


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the hardware documentation section of the website.

5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature at full performance, further cooling may be needed.

6 Availability

Raspberry Pi guarantee availability Pi4B until at least January 2026.

7 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.