

Program Mikrokontroler Atmega 2560

```
#include <SIM800.h>
unsigned long bauds = 9600;
#include <Wire.h>
#include <RtcDS1307.h>
RtcDS1307<TwoWire> Rtc(Wire);
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3f, 20, 4);
#include <Adafruit_Fingerprint.h>
//#include <SoftwareSerial.h>
//SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial2);

int seq, steps;
unsigned long m;

String nim[4];
String nama[4];
String no[4];
String waktu;
String tAbsen;

unsigned char noHp[14];

int btIn = A0;
int btOut = A1;

int bt1[2], bt2[2];
int fp;

void setup()
{
  nim[1] = "1234567890";
  nim[2] = "1234567891";
  nim[3] = "1234567892";

  nama[1] = "Dessy";
  nama[2] = "Shellia";
  nama[3] = "Dasanti";

  no[1] = "\"+6285782221328\""; //6285782221328
  no[2] = "\"+6285782221328\"";
  no[3] = "\"+6285782221328\"";

  Serial.begin(9600);
  while (!Serial);
  delay(100);
```



UNIVERSITAS

MERCU BUANA

```

//-----
-
Serial.println("\n\nAdafruit finger detect test");

finger.begin(57600);

while (1) {
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
    break;
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    delay(1000);
  }
}

finger.getTemplateCount();
Serial.print("Sensor contains ");
Serial.print(finger.templateCount); Serial.println(" templates");
Serial.println("Waiting for valid finger...");

//-----
-

Serial.print("compiled: ");
Serial.print(__DATE__);
Serial.println(__TIME__);

Rtc.Begin();

RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
// printDateTime(compiled);
Serial.println();

if (!Rtc.IsDateTimeValid())
{
  Serial.println("RTC lost confidence in the DateTime!");
  Rtc.SetDateTime(compiled);
}

if (!Rtc.GetIsRunning())
{
  Serial.println("RTC was not actively running, starting now");
  Rtc.SetIsRunning(true);
}

RtcDateTime now = Rtc.GetDateTime();
if (now < compiled)
{
  Serial.println("RTC is older than compile time! (Updating
DateTime)");
}

```

```

    Rtc.SetDateTime(compiled);
}
else if (now > compiled)
{
    Serial.println("RTC is newer than compile time. (this is
expected)");
}
else if (now == compiled)
{
    Serial.println("RTC is the same as compile time! (not expected but
all is fine)");
}

Rtc.SetSquareWavePin(DS1307SquareWaveOut_Low);

if (!Rtc.IsDateTimeValid())
{
    Serial.println("RTC lost confidence in the DateTime!");
}

//-----
-

for (int z = 0; z < 4; z++) {
    SIM.begin(bauds);
    delay(100);

    SIM.setTimeout(3000);
    SIM.cmdBenchmark(true);

    delay(1000);
    SIM.smsDel          (TEST);
    SIM.smsFormat      (SET, "1");
}
//-----
-

homes();
Serial.println();

seq = 0;
steps = 0;

pinMode(btIn, INPUT_PULLUP);
pinMode(btOut, INPUT_PULLUP);

lcd.init();
lcd.backlight();
homes();
}

void loop() // run over and over again

```

```

{
  toggle_steps();
  update_time();
  getFingerprintIDez();
  absen();

  delay(50);          //don't ned to run this at full speed.
}

void homes() {
  lcd.clear();
  RtcDateTime now = Rtc.GetDateTime();

  lcd.setCursor(0, 0);
  lcd.print("  Absensi  Siswa  ");
  lcd.setCursor(0, 1);
  if (steps == 0) lcd.print("Absen : Masuk      ");
  if (steps == 1) lcd.print("Absen : Keluar     ");

  printDateTime(now);
}

void toggle_steps() {
  bt1[0] = digitalRead(btIn);
  bt2[0] = digitalRead(btOut);

  if (bt1[0] != bt1[1]) {
    bt1[1] = bt1[0];

    if (bt1[1] == 0) {
      lcd.setCursor(0, 1);
      lcd.print("Absen : Masuk      ");
      steps = 0;
      delay(100);
    }
  }

  if (bt2[0] != bt2[1]) {
    bt2[1] = bt2[0];

    if (bt2[1] == 0) {
      lcd.setCursor(0, 1);
      lcd.print("Absen : Keluar     ");
      steps = 1;
      delay(100);
    }
  }
}

void update_time() {
  if (millis() - m >= 1000) {
    m = millis();
  }
}

```

```

if (!Rtc.IsDateTimeValid())
{
    Serial.println("RTC lost confidence in the DateTime!");
}

RtcDateTime now = Rtc.GetDateTime();

printDateTime(now);
Serial.println();
}
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
    if (seq == 0) {
        uint8_t p = finger.getImage();
        if (p != FINGERPRINT_OK) return -1;

        p = finger.image2Tz();
        if (p != FINGERPRINT_OK) return -1;

        p = finger.fingerFastSearch();
        if (p != FINGERPRINT_OK) {
            Serial.println("Tidak Ditemukan Dalam Database");
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Finger Print Invalid");
            lcd.setCursor(0, 1);
            lcd.print(" Tidak Ada Dalam Db ");
            lcd.setCursor(0, 3);
            lcd.print("Bersihkan Sidik Jari");
            delay(3000);
            homes();
            seq = 1;
            return -1;
        }
    }

    fp = finger.fingerID;

    // found a match!
    Serial.print("Found ID #"); Serial.print(fp);
    Serial.print(" with confidence of ");
    Serial.println(finger.confidence);
    seq = 1;
    return finger.fingerID;
}

if (seq == 1) {
    uint8_t p = finger.getImage();
    switch (p) {
        case FINGERPRINT_NOFINGER:
            Serial.println("No finger detected");

```

```

        seq = 0;
        fp = 0;
        break;
    }
}
}

void absen() {
    if (seq == 1 && fp > 0 && steps == 0) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("    Selamat Datang    ");

        lcd.setCursor(0, 1);
        lcd.print("Nama : ");
        lcd.print(nama[fp]);

        lcd.setCursor(0, 2);
        lcd.print("Nim : ");
        lcd.print(nim[fp]);

        lcd.setCursor(0, 3);
        lcd.print("Absen : ");
        lcd.print(waktu);

        String msg;
        char m[250];
        char n[250];

        msg = "Sistem Absensi Datang";
        msg += "\n";
        msg += "Nama : ";
        msg += nama[fp];
        msg += "\n";
        msg += "Nim : ";
        msg += nim[fp];
        msg += "\n";
        msg += "Waktu Absen : ";
        msg += tAbsen;

        no[fp].toCharArray(n, 250);
        msg.toCharArray(m, 250);

        SIM.smsSend(n, m);

        delay(3000);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("    Have A Nice Day    ");
        lcd.setCursor(0, 1);
        lcd.print("                ^_^                ");
    }
}
}

```

```

delay(3000);
seq = 0;
fp = 0;
homes();
}

if (seq == 1 && fp > 0 && steps == 1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Selamat Jalan ");

    lcd.setCursor(0, 1);
    lcd.print("Nama : ");
    lcd.print(nama[fp]);

    lcd.setCursor(0, 2);
    lcd.print("Nim : ");
    lcd.print(nim[fp]);

    lcd.setCursor(0, 3);
    lcd.print("Absen : ");
    lcd.print(waktu);

    String msg;
    char m[250];
    char n[250];

    msg = "Sistem Absensi Pulang";
    msg += "\n";
    msg += "Nama : ";
    msg += nama[fp];
    msg += "\n";
    msg += "Nim : ";
    msg += nim[fp];
    msg += "\n";
    msg += "Waktu Absen : ";
    msg += tAbsen;

    no[fp].toCharArray(n, 250);
    msg.toCharArray(m, 250);
    SIM.smsSend(n, m);

    delay(3000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" ThankYou n See You ");
    lcd.setCursor(0, 1);
    lcd.print("      ^ ^      ");
    lcd.setCursor(0, 3);
    lcd.print("      _      ");
    lcd.print(" God Bless You ");
}

```



UNIVERSITAS
MERCU BUANA

```

    delay(3000);
    seq = 0;
    fp = 0;
    homes();
}
}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)
{
    char datestring[25];

    snprintf_P(datestring,
               countof(datestring),
               PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
               dt.Day(),
               dt.Month(),
               dt.Year(),
               dt.Hour(),
               dt.Minute(),
               dt.Second() );
    Serial.println(datestring);

    tAbsen = "";
    for (int i = 0; i < 20; i++) {
        tAbsen += datestring[i];
    }

    lcd.setCursor(0, 3);
    lcd.print(" ");
    lcd.setCursor(0, 3);
    lcd.print(datestring);

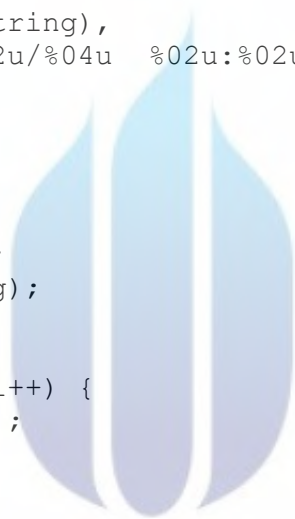
    char datestring2[9];

    snprintf_P(datestring2,
               countof(datestring2),
               PSTR("%02u:%02u:%02u"),
               dt.Hour(),
               dt.Minute(),
               dt.Second() );

    waktu = "";
    for (int i = 0; i < 8; i++) {
        waktu += datestring2[i];
    }

    // Serial.println(waktu);

```



UNIVERSITAS
MERCU BUANA

}



UNIVERSITAS
MERCU BUANA

Fingerprint Sensor Module with Arduino (FPM10A)

Introducing the Fingerprint Sensor Module in the following figure, made fingerprint recognition more accessible and easy to add to your projects.

This means that it is super easy to make fingerprint collection, registration, comparison and search.



These modules come with FLASH memory to store the fingerprints and work with any microcontroller or system with TTL serial. These modules can be added to security systems, door locks, time attendance systems, and much more. Prices for this sensor greatly vary from \$10

Specifications

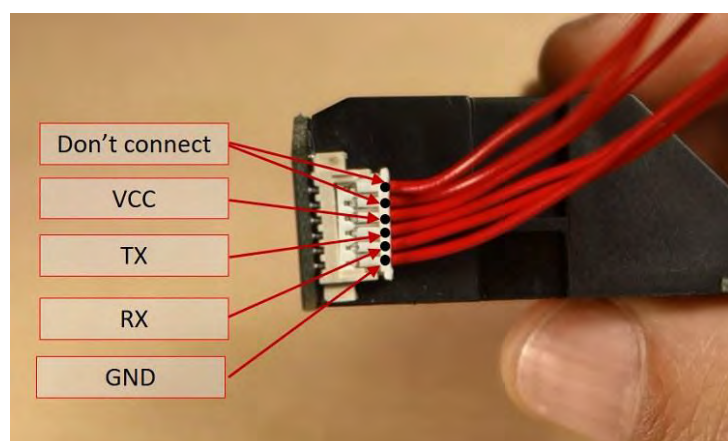
Voltage supply: DC 3.6 to 6.0V, Current supply: <120mA, Backlight color: green, Interface: UART

Bad rate: 9600, Safety level: five (from low to high: 1,2,3,4,5), False Accept Rate (FAR): <0.001% (security level 3)

False Reject Rate (FRR): <1.0% (security level 3), Able to store 127 different fingerprints

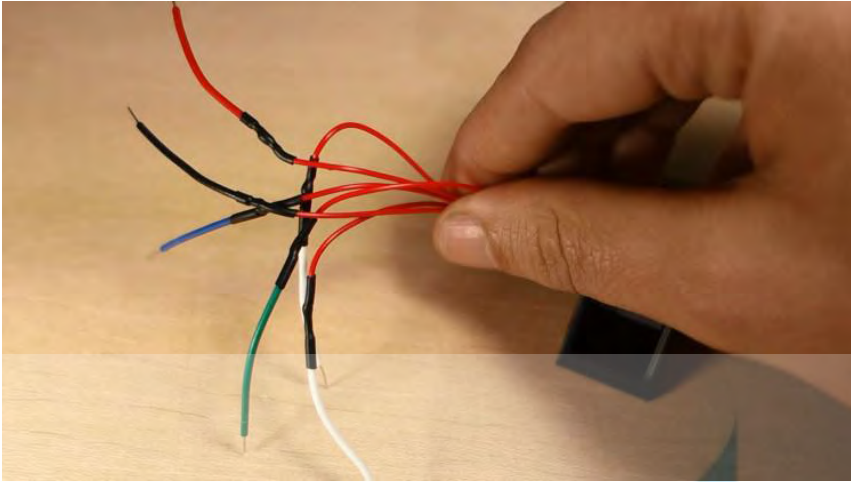
Sensor Pinout

The sensor has six pins that are labeled in the figure below.



The fingerprint sensor module used in this project came with really thin wires, so soldering breadboard-friendly wires was needed. We recommend using different colors according to the pin function. In our case:

DNC - white wires
VCC - red wire
TX - blue wire
RX - green wire
GND - black wire



The following table shows how to wire the sensor to the Arduino.

Fingerprint Sensor	Arduino
VCC	5V (it also works with 3.3V)
TX	RX (digital pin 2, software serial)
RX	TX (digital pin 3, software serial)
GND	GND

Installing the Adafruit Fingerprint Sensor Library

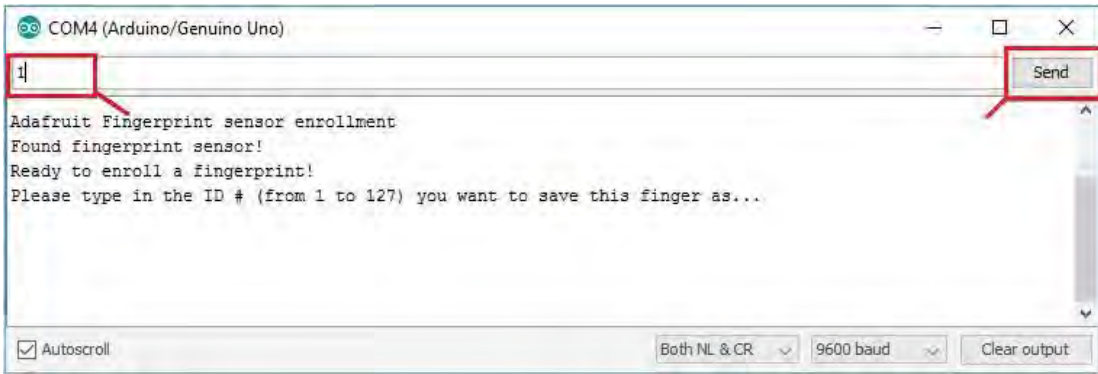
The easiest way to control the fingerprint sensor module with the Arduino is by using the Adafruit library for this sensor. Follow the next instructions to install the library:

1. Click here to download the [Adafruit Fingerprint Sensor library](#).
2. Unzip the .zip folder and you should get Adafruit-Fingerprint-Sensor-Library-master folder
3. Rename your folder from ~~Adafruit-Fingerprint-Sensor-Library-master~~ folder to **Adafruit_Fingerprint_Sensor_Library folder**
4. Move the folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

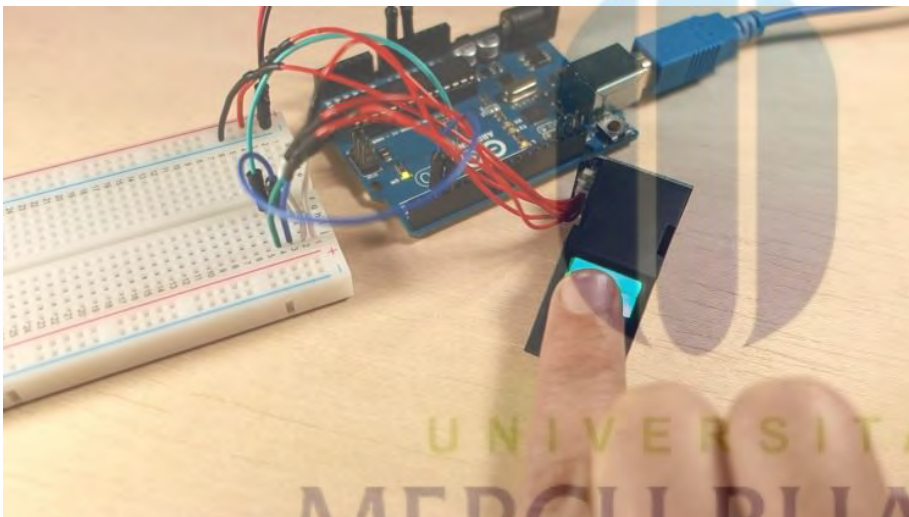
Enroll a New Fingerprint

Having the fingerprint sensor module wired to the Arduino, follow the next steps to enroll a new fingerprint. Make sure you've installed the Adafruit Fingerprint Sensor library previously.

1. In the Arduino IDE, go to **File > Examples > Adafruit Fingerprint Sensor Library > Enroll**.
2. Upload the code, and open the serial monitor at a baud rate of 9600.
3. You should enter an ID for the fingerprint. As this is your first fingerprint, type 1 at the top left corner, and then, click the **Send** button.



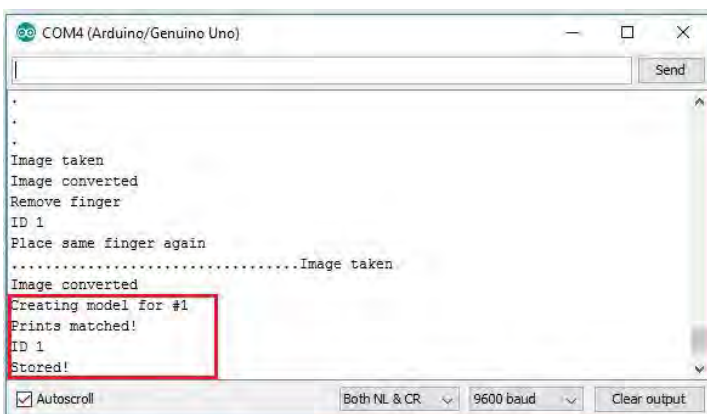
4. Place your finger on the scanner and follow the instructions on the serial monitor.



You'll be asked to place the same finger twice on the scanner.

If you get the "**Prints matched!**" message, as shown below, your fingerprint was successfully stored.

If not, repeat the process, until you succeed.

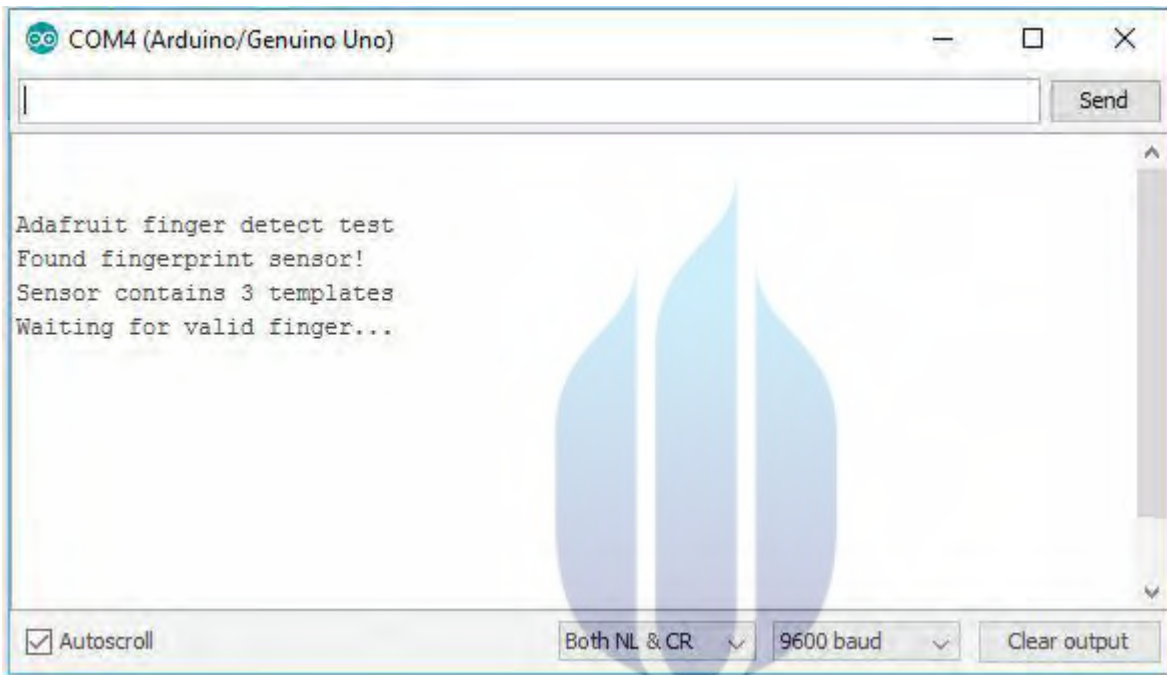


Store as many fingerprints you want using this method.

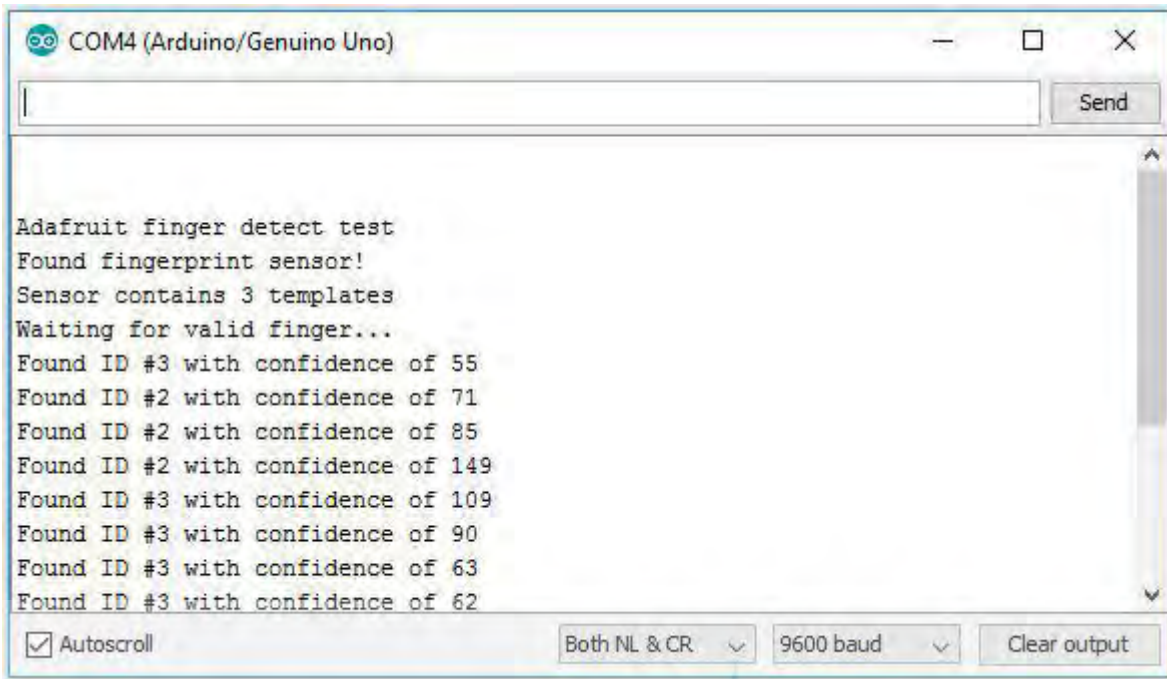
Finding a Match

You now should have several fingerprints saved on different IDs. To find a match with the fingerprint sensor, follow the next instructions.

1. In the Arduino IDE, go to **File > Examples > Adafruit Fingerprint Sensor Library > Fingerprint** and upload the code to your Arduino board.
2. Open the Serial Monitor at a baud rate of 9600. You should see the following message:



3. Place the finger to be identified on the scan.
4. On the serial monitor, you can see the ID that matches the fingerprint. It also shows the confidence - the higher the confidence, the similar the fingerprint is with the stored fingerprint.



```
COM4 (Arduino/Genuino Uno)

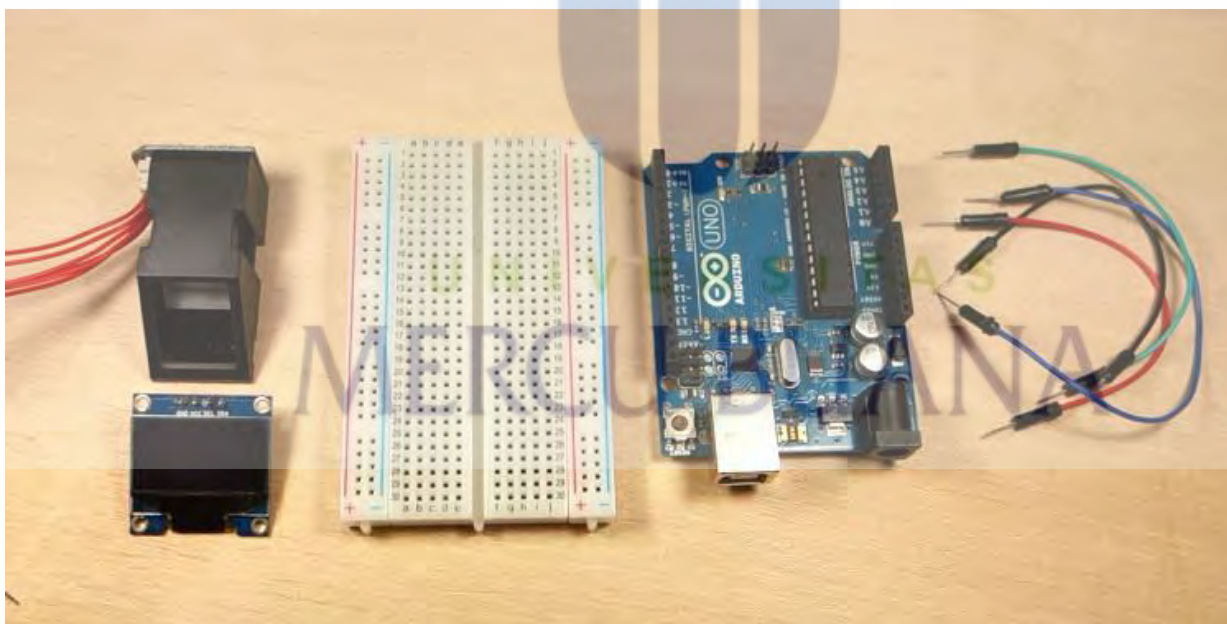
| Send

Adafruit finger detect test
Found fingerprint sensor!
Sensor contains 3 templates
Waiting for valid finger...
Found ID #3 with confidence of 55
Found ID #2 with confidence of 71
Found ID #2 with confidence of 85
Found ID #2 with confidence of 149
Found ID #3 with confidence of 109
Found ID #3 with confidence of 90
Found ID #3 with confidence of 63
Found ID #3 with confidence of 62

 Autoscroll
Both NL & CR
9600 baud
Clear output
```

Project Example - Show Fingerprint Match on OLED display

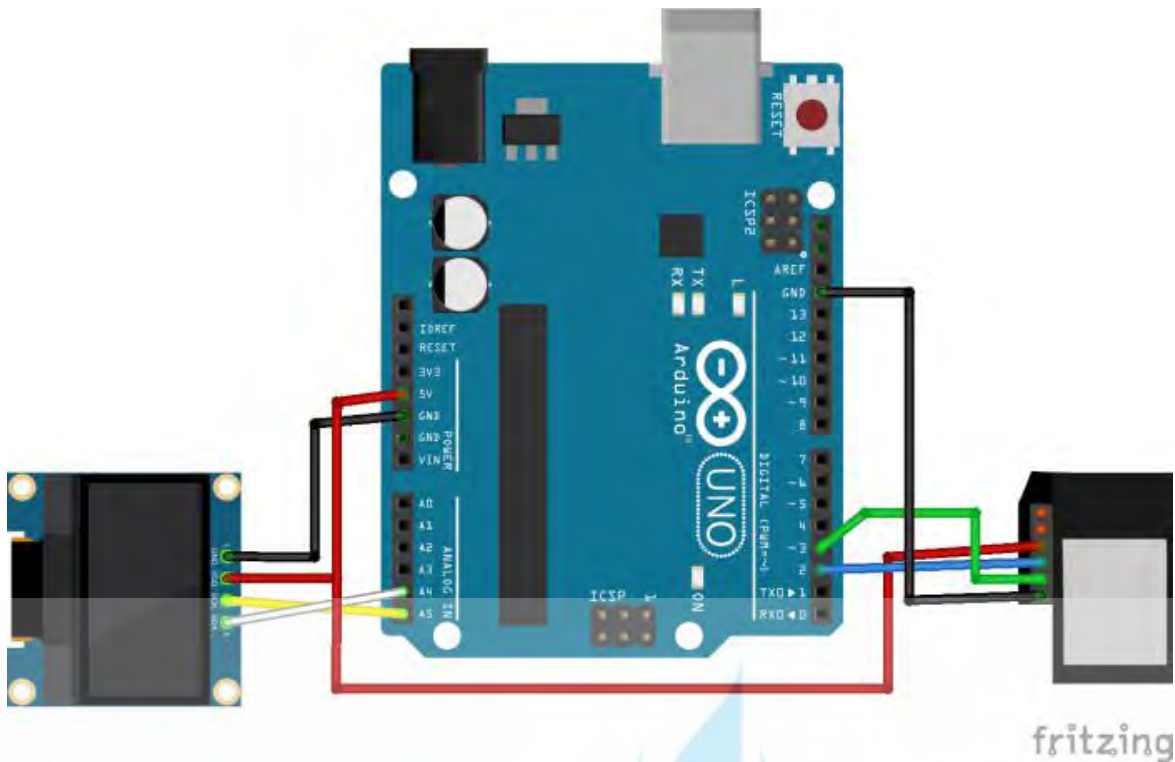
In this project example, we'll enroll two fingerprints from two different persons. Then, we'll display a greeting message accordingly to the match found, on an OLED display. For this example you'll need the following parts:



You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematics

Here's the wiring diagram you should follow to make the circuit for this project.



Installing the 0.96 inch OLED libraries

To control the OLED display you need the "Adafruit_GFX.h" library and "Adafruit_SSD1306.h" library. Follow the next steps to install those libraries:

Installing the Adafruit_GFX library

1. [Click here to download the Adafruit GFX library.](#) You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get **Adafruit-GFX-Library-master** folder
3. Rename your folder from **Adafruit-GFX-Library-master** to **Adafruit_GFX_Library** (you really need to replace those "-" by "_")
4. Move the **Adafruit_GFX_Library** folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

Installing the adafruit_SSD1306 library

1. [Click here to download the Adafruit_SSD1306 library.](#) Y
2. Unzip the .zip folder and you should get **Adafruit-GFX-Library-master** folder
3. Rename your folder from **Adafruit_SSD1306-master** to **Adafruit_SSD1306**
4. Move the **Adafruit_SSD1306** folder to your Arduino IDE installation libraries folder

5. Finally, re-open your Arduino IDE

Code

Before uploading the code, you need to enroll different fingerprints from different persons. Go to **"Enroll a New Fingerprint"** section above, upload the given code and follow the instructions to enroll two fingerprints.

Then, modify the code so that the fingerprint IDs match the name of the persons enrolled – scroll down to page for an explanation of the code.

Finally, you can upload the code provided.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
int fingerprintID = 0;
String IDname;

void setup(){
  //Fingerprint sensor module setup
  Serial.begin(9600);
  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  }
  else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }

  //OLED display setup
  Wire.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  //displays main screen
  displayMainScreen();
}

void loop(){
  displayMainScreen();
  fingerprintID = getFingerprintIDez();
```



UNIVERSITAS
MERCU BUANA


```

delay(50);
if(fingerprintID == 1 || fingerprintID == 3 || fingerprintID == 4 || fingerprintID == 5){
    IDname = "Sara";
    displayUserGreeting(IDname);
}
else if(fingerprintID == 2){
    IDname = "Rui";
    displayUserGreeting(IDname);
}
}

```

// returns -1 if failed, otherwise returns ID #

```

int getFingerprintIDez() {
    uint8_t p = finger.getImage();
    if (p != FINGERPRINT_OK) return -1;

    p = finger.image2Tz();
    if (p != FINGERPRINT_OK) return -1;

    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK) return -1;

    // found a match!
    Serial.print("Found ID #");
    Serial.print(finger.fingerID);
    Serial.print(" with confidence of ");
    Serial.println(finger.confidence);
    return finger.fingerID;
}

```

```

void displayMainScreen(){
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(7,5);
    display.println("Waiting fingerprint");
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(52,20);
    display.println("...");
    display.display();
    delay(2000);
}

```

```

void displayUserGreeting(String Name){
    display.clearDisplay();
    display.setTextColor(WHITE);
    display.setTextSize(2);
    display.setCursor(0,0);

```



UNIVERSITAS
MERCU BUANA

```

display.print("Hello");
display.setCursor(0,15);
display.print(Name);
display.display();
delay(5000);
fingerprintID = 0;
}

```

Importing libraries

The code starts by importing the needed libraries to write in the OLED display, and creates an `Adafruit_SSD1306` object called `display`.

```

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

```

We also need to import the libraries needed for the fingerprint sensor: *Adafruit_Fingerprint.h* and *SoftwareSerial.h*.

```

#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);

```

The following line sets software serial on pins 2 and 3. Pin 2 as RX, and Pin 3 as TX.

```
SoftwareSerial mySerial(2, 3);
```

Then, we create a an *Adafruit_Fingerprint* object called *finger* on the serial pins we've set previously.

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
```

The next two lines create variables to hold the fingerprint ID and the IDname.

```
int fingerprintID = 0;
String IDname;
```

setup()

In the `setup()`, both the fingerprint sensor and the OLED display are initialized. We also print a message on the serial monitor so that we know if the fingerprint sensor was found successfully.

```

void setup(){
  //Fingerprint sensor module setup
  Serial.begin(9600);
  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {

```

```

Serial.println("Found fingerprint sensor!");
}
else {
Serial.println("Did not find fingerprint sensor :(");
while (1) { delay(1); }
}

```

```

//OLED display setup
Wire.begin();
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
//displays main screen
displayMainScreen();
}

```

loop()

In the loop(), the code displays the main screen on the OLED display - this is done in the *displayMainScreen()* function. Then, the code is continuously checking for incoming fingerprints. If the sensor finds a saved fingerprint, the Arduino saves the corresponding ID in the *fingerprintID* variable.

Then, the code has an if/else statement to check the ID the fingerprint corresponds to. You should edit the following lines of code with the corresponding IDs and names.

```

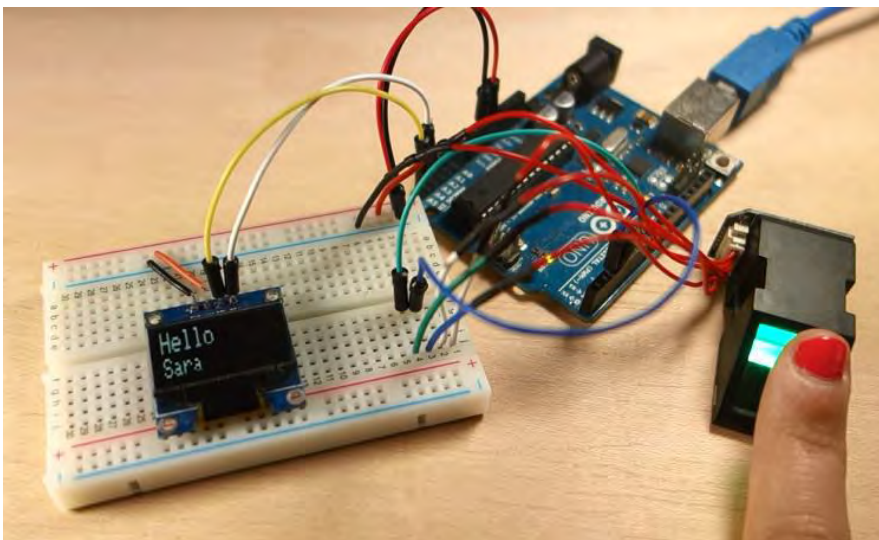
if(fingerprintID == 1 || fingerprintID == 3 || fingerprintID == 4 || fingerprintID == 5){
IDname = "Sara";
displayUserGreeting(IDname);
}
else if(fingerprintID == 2){
IDname = "Rui";

```

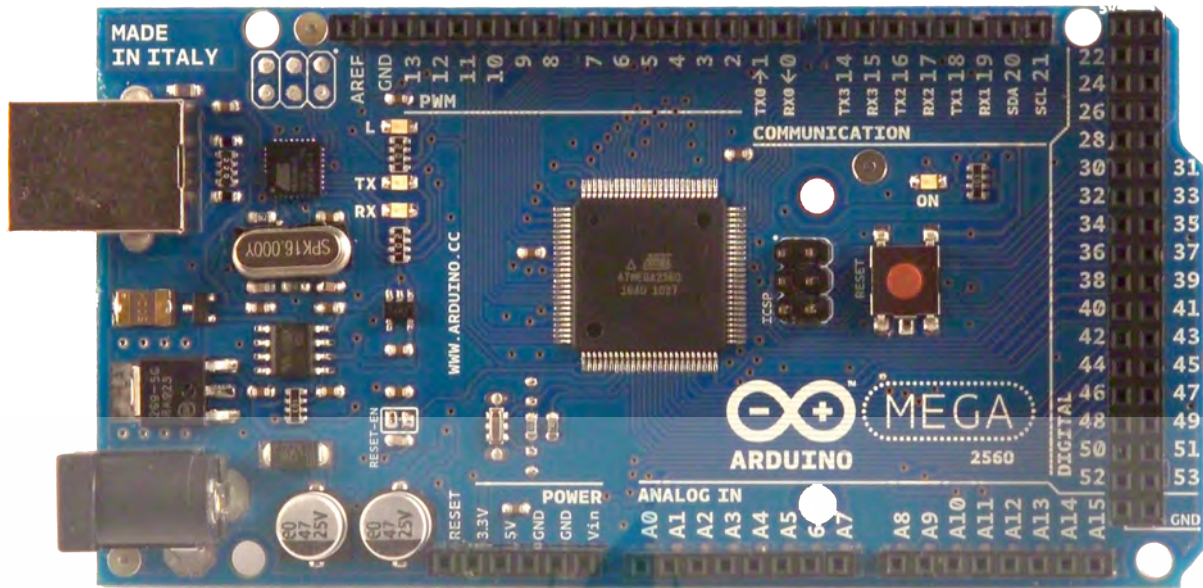
Sometimes, the sensor will recognize a fingerprint better if it is saved several times in different IDs. After identifying the ID name, the OLED displays a greeting - this is done in the *displayUserGreeting()* function,

Demonstration

Now, when a person with a saved fingerprint places the finger on the sensor, it displays a greeting message.



Arduino MEGA 2560



Product 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.

Index

Technical Specifications

Page 2

How to use Arduino
Programming Enviroment, Basic Tutorials

Page 6

Terms & Conditions

Page 7

Enviromental Policies
half sqm of green via Impatto Zero®

Page 7



radiospares

RADIONICS



Technical Specification

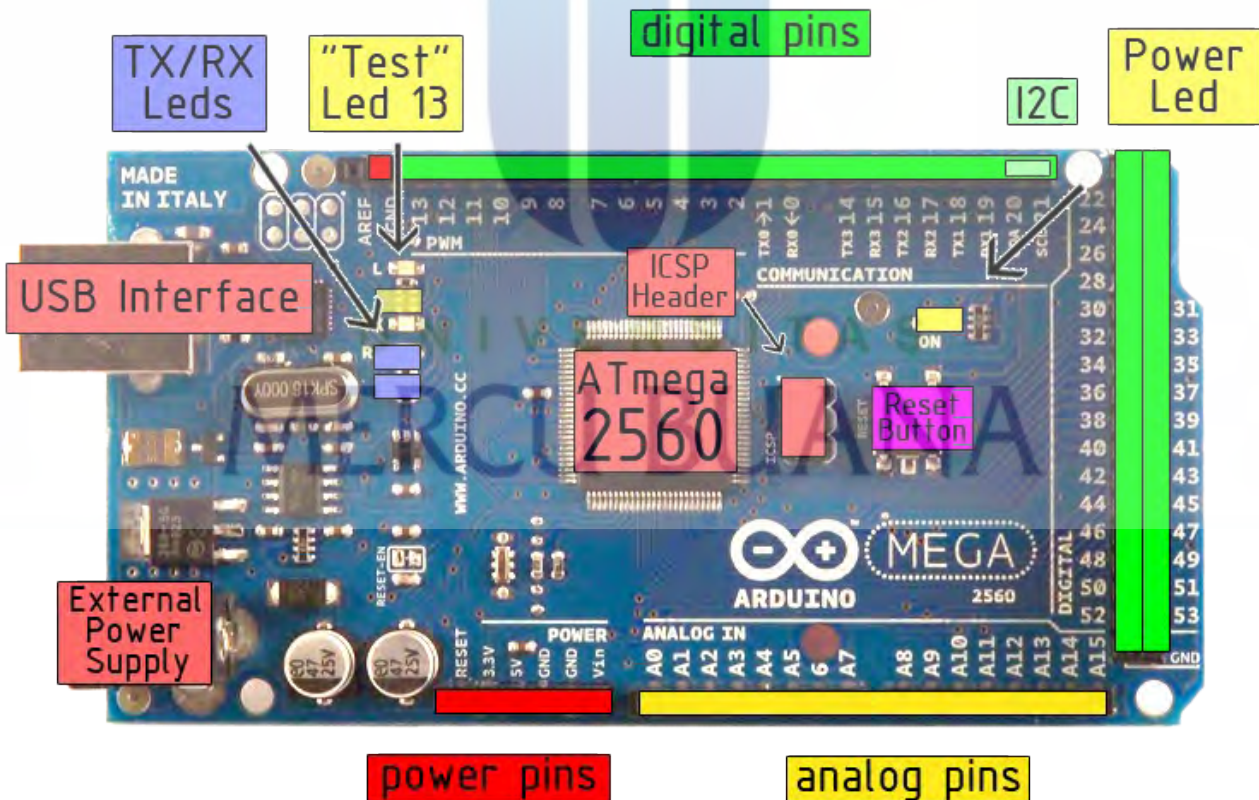


EAGLE files: [arduino-mega2560-reference-design.zip](#) Schematic: [arduino-mega2560-schematic.pdf](#)

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

the board



radiospares

RADIONICS



Power

The Arduino Mega2560 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, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the 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.

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.



radiospares **RADIONICS**



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 Mega's digital pins.

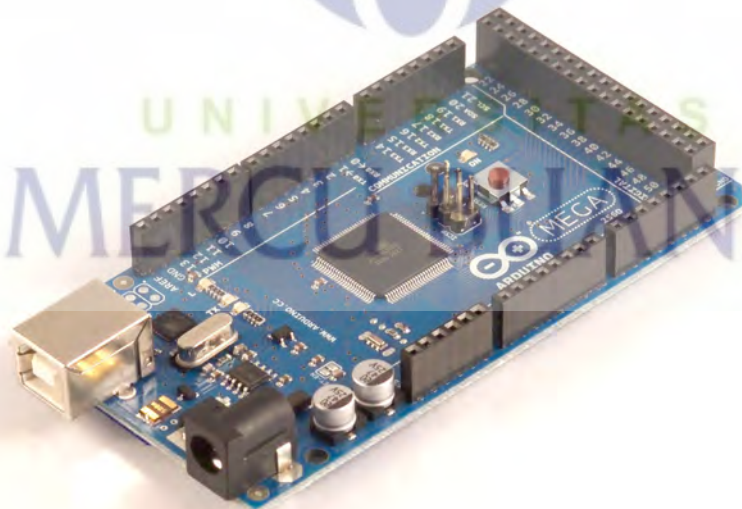
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation on the Wiring website](#) for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 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.



radiospares

RADIONICS



Automatic (Software) Reset

Rather than 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 Mega 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 Mega 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 Mega 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 Mega is designed to be compatible with most shields designed for the 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 Mega and Duemilanove / Diecimila. **Please note that I²C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**



RADIOSPARES RADIONICS

<http://digilib.mercubuana.ac.id/>



How to use Arduino



Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the [Arduino programming language](#) (based on [Wiring](#)) and the Arduino development environment (based on [Processing](#)). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the [Arduino site](#) for the latest instructions. <http://arduino.cc/en/Guide/HomePage>

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

**File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink**

Once you have your sketch you'll see something very close to the screenshot on the right.

In **Tools>Board** select MEGA

Now you have to go to **Tools>SerialPort** and select the right serial port, the one arduino is attached to.

```
Blink | Arduino 0017
File Edit Sketch Tools Help
Blink $
int ledPin = 13; // LED connected to digital pin 13
// The setup() method runs once, when the sketch starts
void setup() {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}
// the loop() method runs over and over again,
// as long as the Arduino has power
void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on
  delay(1000); // wait for a second
  digitalWrite(ledPin, LOW); // set the LED off
  delay(1000); // wait for a second
}
```



Done compiling.

Press Compile button
(to check for errors)



Upload



TX RX Flashing



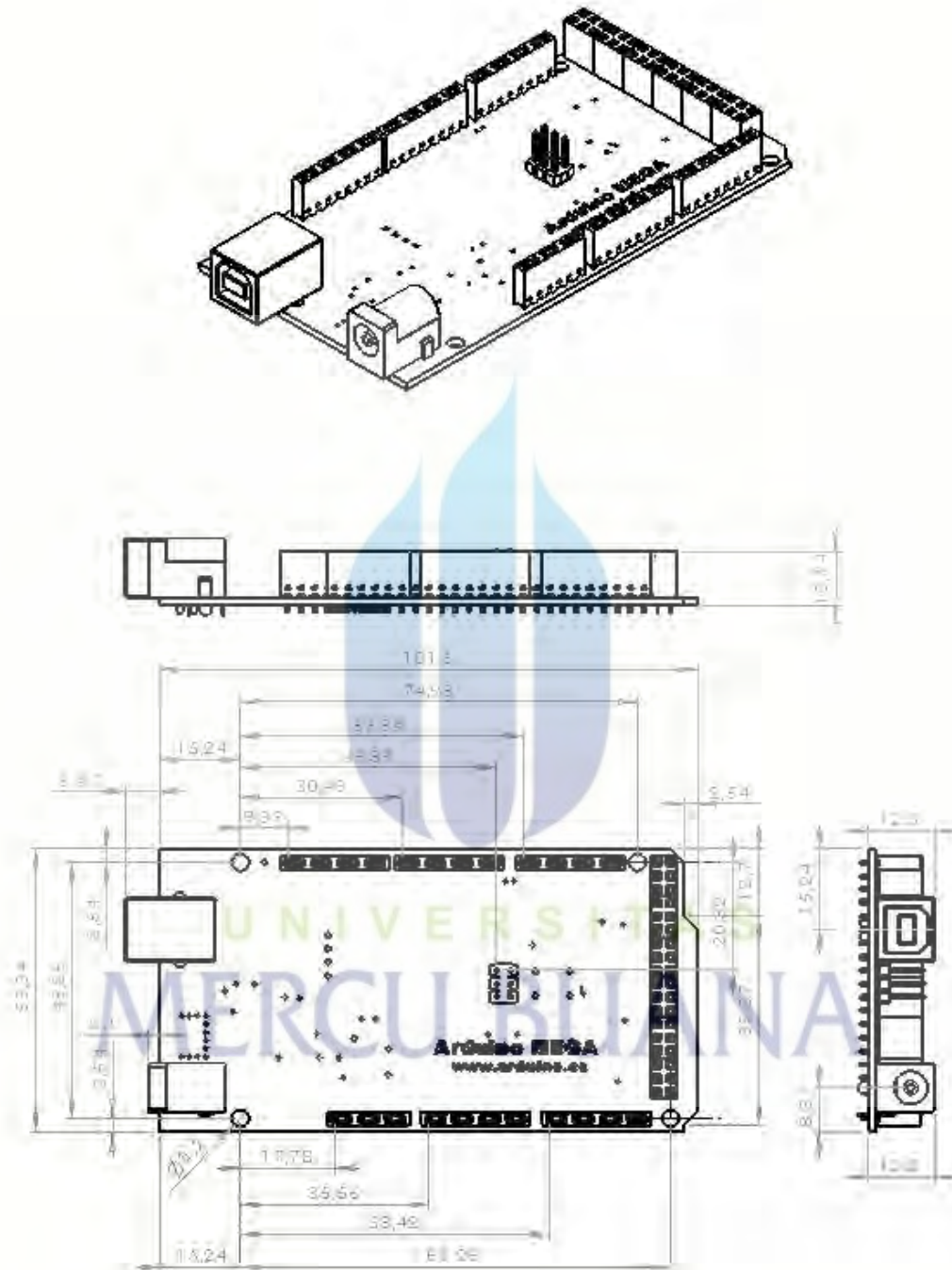
Blinking Led!



radiospares

RADIONICS





Terms & Conditions



1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.



Environmental Policies



The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.



RADIOSPARES RADIONICS

<http://digilib.mercubuana.ac.id/>



Systronix 20x4 LCD Brief Technical Data

July 31, 2000

Here is brief data for the Systronix 20x4 character LCD. It is a DataVision part and uses the Samsung KS0066 LCD controller. It's a clone of the Hitachi HD44780. We're not aware of any incompatibilities between the two - at least we have never seen any in all the code and custom applications we have done.

This 20x4 LCD is electrically and mechanically interchangeable with 20x4 LCDs from several other vendors. The only differences we've seen among different 20x4 LCDs are:

- 1) LED backlight brightness, voltage and current vary widely, as does the quality of the display
- 2) There is a resistor "Rf" which sets the speed of the LCD interface by controlling the internal oscillator frequency. Several displays we have evaluated have a low resistor value. This makes the display too slow. Looking at the Hitachi data sheet page 56, it appears that perhaps the "incorrect" resistor is really intended for 3V use of the displays.

At 5V the resistor Rf should be 91 Kohms. At 3V it should be 75 Kohms. Using a 3V display at 5V is acceptable from a voltage standpoint (the display can operate on 3-5V) but the oscillator will then be running too slowly. One fix is to always check the busy flag and not use a fixed time delay in your code, then it will work regardless of the LCD speed. The other option is to always allow enough delay for the slower display.

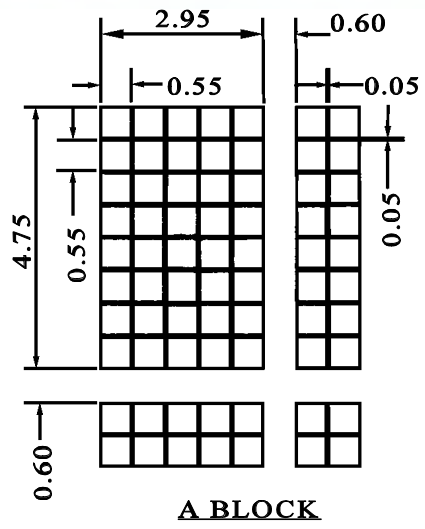
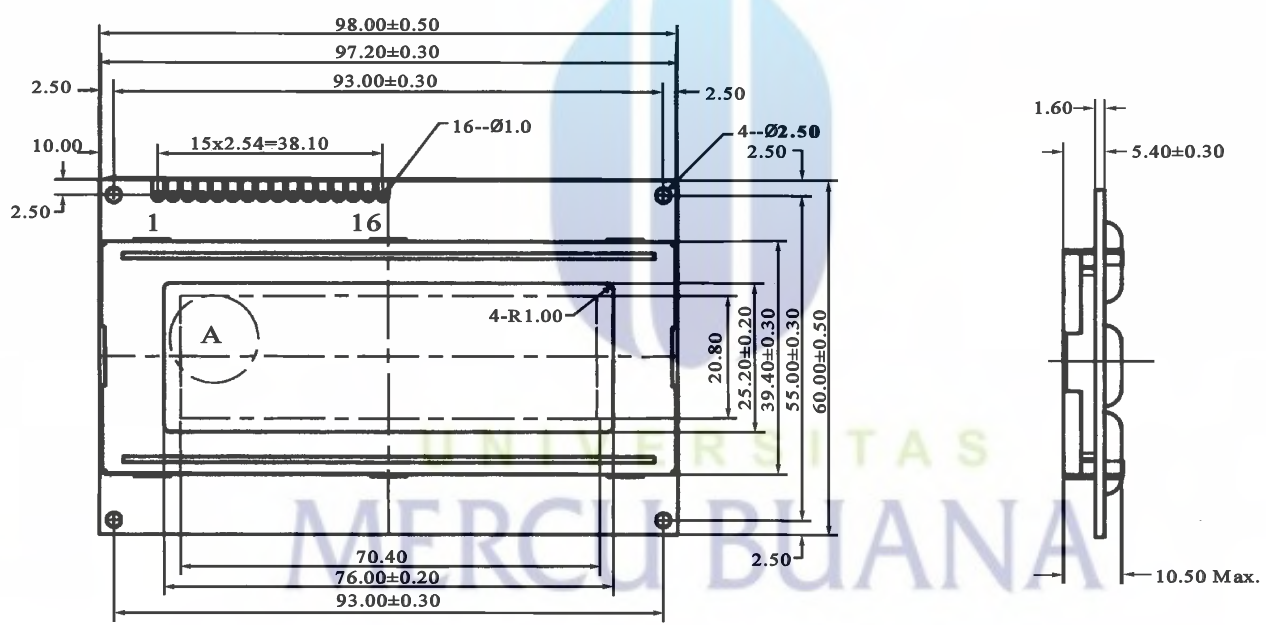
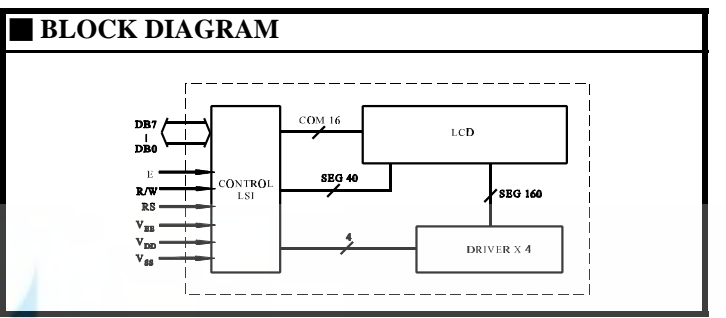
All Systronix 20x4 LCDs have the 91 Kohm resistor and are intended for 5V operation.

Thank you for purchasing Systronix embedded control products and accessories. If you have any other questions please email to support@systronix.com or phone +1-801-534-1017, fax +1-801-534-1019.

ABSOLUTE MAXIMUM RATINGS					
Item	Symbol	Standard Value			Unit
		Min.	Typ.	Max.	
Supply Voltage for Logic	V _{DD}	0	—	7.0	V
Supply Voltage for LCD Driver	V _{DD} -V _{EE}	—	—	13.5	V
Input Voltage	V _I	V _{SS}	—	V _{DD}	V
Operature Temp.	T _{opr}	0	—	50	°C
Storage Temp.	T _{stg}	-20	—	70	°C

ELECTRICAL CHARACTERISTICS (REFLECTIVE TYPE)						
Item	Symbol	Test Condition	Standard Value			Unit
			Min.	Typ.	Max.	
Input "High" Voltage	V _{IH}	—	2.2	—	V _{EE}	V
Input "Low" Voltage	V _{IL}	—	—	—	0.6	V
Output "High" Voltage	V _{OH}	I _{OH} =0.2mA	2.2	—	—	V
Output "Low" Voltage	V _{OL}	I _{OL} =1.2mA	—	—	0.4	V
Supply Current	I _{DD}	V _{DD} =5.0A	—	2.5	4.0	mA

PIN FUNCTIONS					
No	Symbol	Function	No	Symbol	Function
1	V _{SS}	GND, 0V	10	DB3	Data Bus
2	V _{DD}	+5V	11	DB4	—
3	V _{EE}	for LCD Drive	12	DB5	—
4	RS	Function Select	13	DB6	—
5	R/W	Read/Write	14	DB7	—
6	E	Enable Signal	15	LEDA	LED Power Supply
7-9	DB0-DB2	Data Bus Line	16	LEDA	



HD44780U

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A00)

Lower 4 Bits	Upper 4 Bits				1000	1001	1010	1011	1100	1101	1110	1111	
	0000	0001	0010	0011									
xxxx0000	CG RAM (1)			0aP`P				-	9	3	o	p	
xxxx0001	(2)			!1A0a*				u	7	7	4	ä	q
xxxx0010	(3)			"2BRbr				「	イ	ウ	×	þ	ø
xxxx0011	(4)			#3CScs				」	ウ	テ	エ	š	o
xxxx0100	(5)			\$4DTdt				、	エ	ト	ト	μ	o
xxxx0101	(6)			%5EUeu				・	オ	ナ	1	o	ü
xxxx0110	(7)			&6FUfv				ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)			'7GWgw				フ	キ	ヌ	ヲ	g	π
xxxx1000	(1)			(8HXhx				イ	ウ	ネ	リ	ſ	×
xxxx1001	(2))9IYiy				o	ウ	ル		ı	ı
xxxx1010	(3)			*:JZJz				エ	コ	ン	レ	j	ſ
xxxx1011	(4)			+;Klk<				オ	ウ	ヒ	ロ	°	π
xxxx1100	(5)			,<L#ll				オ	シ	フ	ワ	o	π
xxxx1101	(6)			-=Mln>				ユ	ズ	ン	ト	÷	
xxxx1110	(7)			.>N^n÷				ヨ	セ	ホ	°	ñ	
xxxx1111	(8)			/?O_oe				ウ	リ	ヌ	°	o	■

Note: The user can specify any pattern for character-generator RAM.

Initializing by Instruction

If the power supply conditions for correctly operating the internal reset circuit are not met, initialization by instructions becomes necessary.

Refer to Figures 25 and 26 for the procedures on 8-bit and 4-bit initializations, respectively.

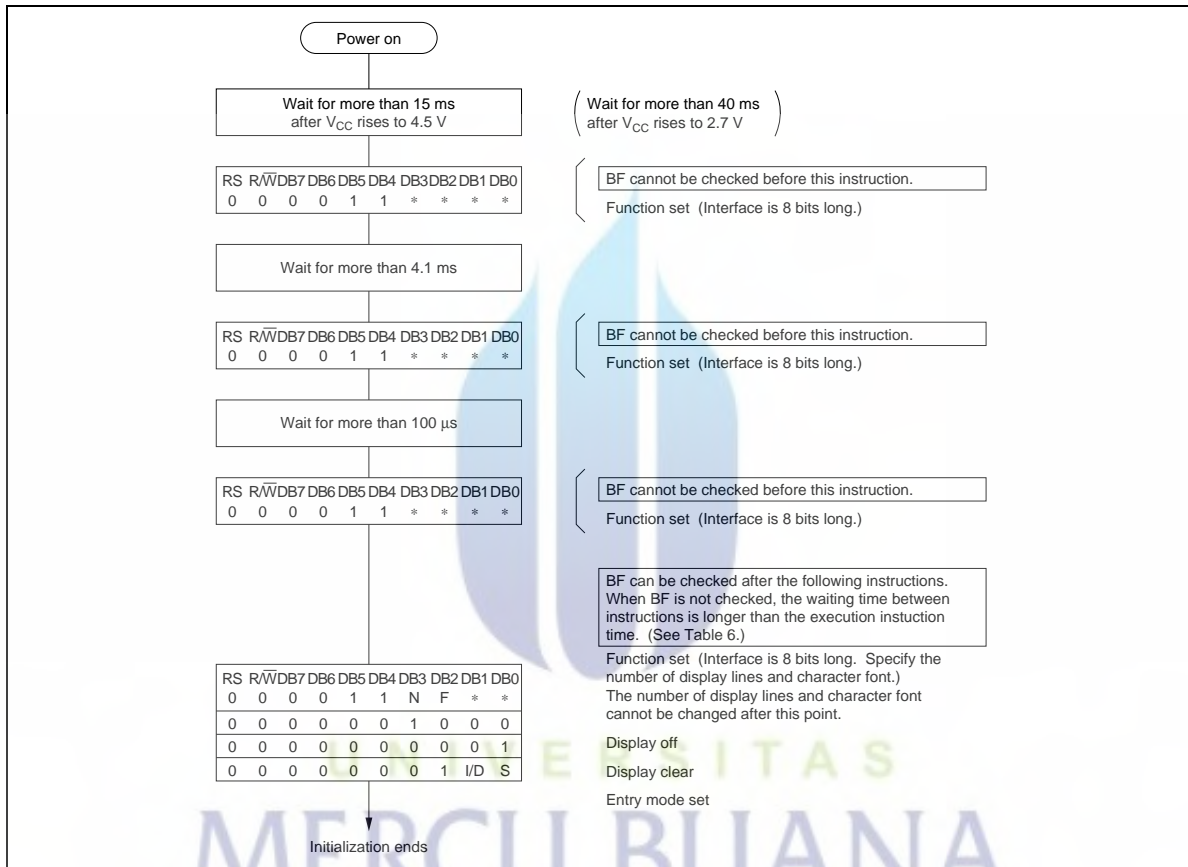


Figure 25 8-Bit Interface

Reset Function

Initializing by Internal Reset Circuit

An internal reset circuit automatically initializes the HD44780U when the power is turned on. The following instructions are executed during the initialization. The busy flag (BF) is kept in the busy state until the initialization ends (BF = 1). The busy state lasts for 10 ms after V_{CC} rises to 4.5 V.

1. Display clear
2. Function set:
DL = 1; 8-bit interface data
N = 0; 1-line display
F = 0; 5 × 8 dot character font
3. Display on/off control:
D = 0; Display off
C = 0; Cursor off
B = 0; Blinking off
4. Entry mode set:
I/D = 1; Increment by 1
S = 0; No shift

Note: If the electrical characteristics conditions listed under the table Power Supply Conditions Using Internal Reset Circuit are not met, the internal reset circuit will not operate normally and will fail to initialize the HD44780U. For such a case, initialization must be performed by the MPU as explained in the section, Initializing by Instruction.

Instructions

Outline

Only the instruction register (IR) and the data register (DR) of the HD44780U can be controlled by the MPU. Before starting the internal operation of the HD44780U, control information is temporarily stored into these registers to allow interfacing with various MPUs, which operate at different speeds, or various peripheral control devices. The internal operation of the HD44780U is determined by signals sent from the MPU. These signals, which include register selection signal (RS), read/

write signal (R/\overline{W}), and the data bus (DB0 to DB7), make up the HD44780U instructions (Table 6). There are four categories of instructions that:

- Designate HD44780U functions, such as display format, data length, etc.
- Set internal RAM addresses
- Perform data transfer with internal RAM
- Perform miscellaneous functions

Normally, instructions that perform data transfer with internal RAM are used the most. However, auto-incrementation by 1 (or auto-decrementation by 1) of internal HD44780U RAM addresses after each data write can lighten the program load of the MPU. Since the display shift instruction (Table 11) can perform concurrently with display data write, the user can minimize system development time with maximum programming efficiency.

When an instruction is being executed for internal operation, no instruction other than the busy flag/address read instruction can be executed.

Because the busy flag is set to 1 while an instruction is being executed, check it to make sure it is 0 before sending another instruction from the MPU.

Note: Be sure the HD44780U is not in the busy state (BF = 0) before sending an instruction from the MPU to the HD44780U. If an instruction is sent without checking the busy flag, the time between the first instruction and next instruction will take much longer than the instruction time itself. Refer to Table 6 for the list of each instruction execution time.

Table 6 Instructions

Instruction	Code										Description	Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 μ s
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 μ s
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 μ s
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 μ s
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 μ s
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 μ s
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μ s

HD44780U

Table 6 Instructions (cont)

Instruction	Code										Description	Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)		
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		$t_{ADD} = 4 \mu s^*$		
Write data to CG or DDRAM	1	0	Write data										Writes data into DDRAM or CGRAM.	$37 \mu s$ $t_{ADD} = 4 \mu s^*$
Read data from CG or DDRAM	1	1	Read data										Reads data from DDRAM or CGRAM.	$37 \mu s$ $t_{ADD} = 4 \mu s^*$
	I/D = 1: Increment I/D = 0: Decrement										DDRAM: Display data RAM CGRAM: Character generator RAM	Execution time changes when frequency changes		
	S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move										ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address)	Example: When f_{cp} or f_{osc} is 250 kHz, $37 \mu s \times \frac{270}{250} = 40 \mu s$		
	R/L = 1: Shift to the right R/L = 0: Shift to the left										AC: Address counter used for both DD and CGRAM addresses			
	DL = 1: 8 bits, DL = 0: 4 bits													
	N = 1: 2 lines, N = 0: 1 line													
	F = 1: 5×10 dots, F = 0: 5×8 dots													
	BF = 1: Internally operating BF = 0: Instructions acceptable													

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.

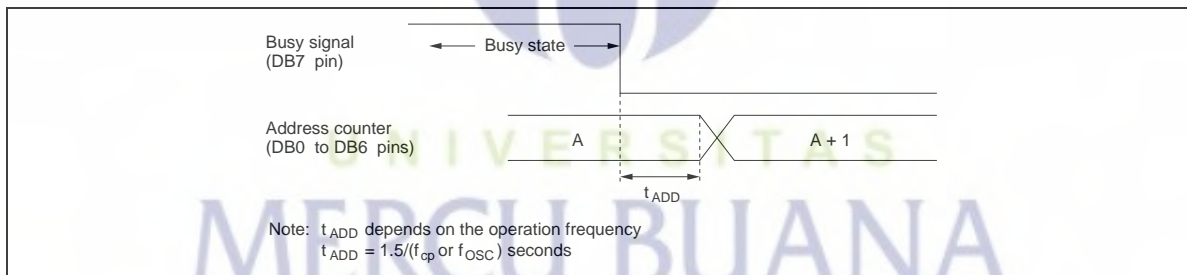
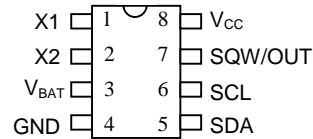


Figure 10 Address Counter Update

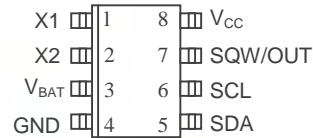
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

PIN DESCRIPTION

V _{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V _{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

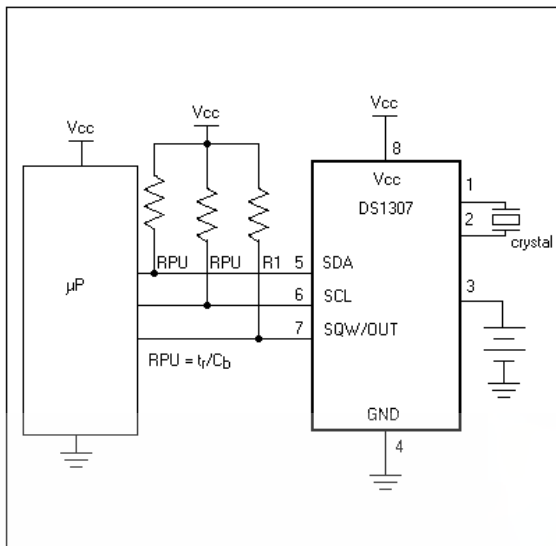
ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

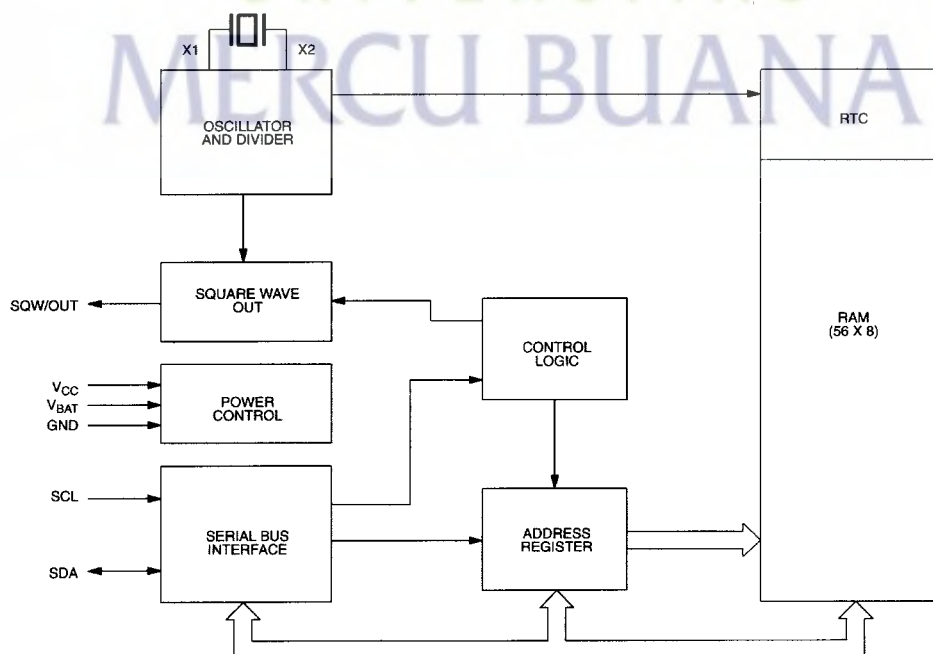
TYPICAL OPERATING CIRCUIT



OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below $1.25 \times V_{BAT}$ the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low-current battery backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than $V_{BAT} + 0.2V$ and recognizes inputs when V_{CC} is greater than $1.25 \times V_{BAT}$. The block diagram in Figure 1 shows the main elements of the serial RTC.

DS1307 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, GND – DC power is provided to the device on these pins. V_{CC} is the +5V input. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} – Battery input for any standard 3V lithium cell or other energy source. Battery voltage must be held between 2.0V and 3.5V for proper operation. The nominal write protect trip point voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAh or greater will back up the DS1307 for more than 10 years in the absence of power at 25°C. UL recognized to ensure against reverse charging current when used in conjunction with a lithium battery.

See “Conditions of Acceptability” at <http://www.maxim-ic.com/TechSupport/QA/ntrl.htm>.

SCL (Serial Clock Input) – SCL is used to synchronize data movement on the serial interface.

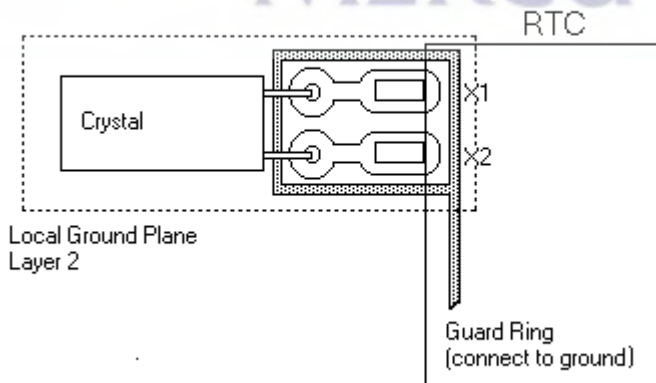
SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pull-up resistor. SQW/OUT will operate with either V_{cc} or V_{bat} applied.

X1, X2 – Connections for a standard 32.768kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5pF.

For more information on crystal selection and crystal layout considerations, please consult Application Note 58, “Crystal Considerations with Dallas Real-Time Clocks.” The DS1307 can also be driven by an external 32.768kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

RECOMMENDED LAYOUT FOR CRYSTAL



CLOCK ACCURACY

The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. See Application Note 58, “Crystal Considerations with Dallas Real-Time Clocks” for detailed information.

Please review Application Note 95, “Interfacing the DS1307 with a 8051-Compatible Microcontroller” for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307 is shown in Figure 2. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2

00H	SECONDS
	MINUTES
	HOURS
	DAY
	DATE
	MONTH
	YEAR
07H	CONTROL
08H	RAM
3FH	56 x 8

CLOCK AND CALENDAR

The time and calendar information is obtained by reading the appropriate register bytes. The RTC registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. Bit 7 of register 0 is the clock halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power-on state of all registers is not defined. Therefore, it is important to enable the oscillator (CH bit = 0) during initial configuration.

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

On a 2-wire START, the current time is transferred to a second set of registers. The time information is read from these secondary registers, while the clock may continue to run. This eliminates the need to re-read the registers in case of an update of the main registers during a read.

DS1307 TIMEKEEPER REGISTERS Figure 3

BIT7										BIT0		
00H	CH	10 SECONDS				SECONDS						00-59
	0	10 MINUTES				MINUTES						00-59
	0	12 24	10 HR A/P	10 HR		HOURS						01-12 00-23
	0	0	0	0	0	DAY						1-7
	0	0	10 DATE		DATE						01-28/29 01-30 01-31	
	0	0	0	10 MONTH	MONTH						01-12	
	10 YEAR				YEAR						00-99	
07H	OUT	0	0	SQWE	0	0	RS1	RS0				

CONTROL REGISTER

The DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

OUT (Output control): This bit controls the output level of the SQW/OUT pin when the square wave output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0.

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits. With the square wave output set to 1Hz, the clock registers update on the falling edge of the square wave.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

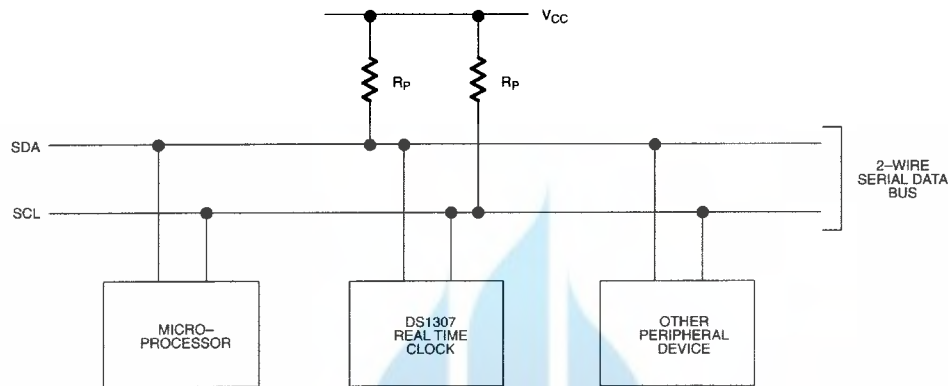
SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

2-WIRE SERIAL DATA BUS

The DS1307 supports a bi-directional, 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307 operates as a slave on the 2-wire bus. A typical bus configuration using this 2-wire protocol is shown in Figure 4.

TYPICAL 2-WIRE BUS CONFIGURATION Figure 4



Figures 5, 6, and 7 detail how data is transferred on the 2-wire bus.

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

Start data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

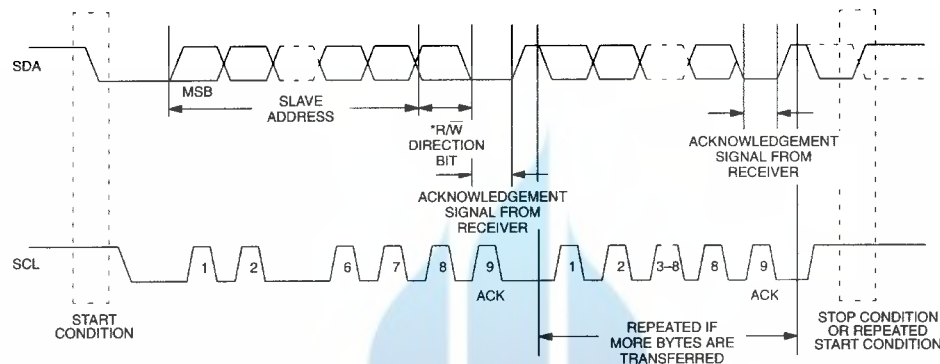
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the 2-wire bus specifications a regular mode (100kHz clock rate) and a fast mode (400kHz clock rate) are defined. The DS1307 operates in the regular mode (100kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

DATA TRANSFER ON 2-WIRE SERIAL BUS Figure 5



Depending upon the state of the $\overline{R/\overline{W}}$ bit, two types of data transfer are possible:

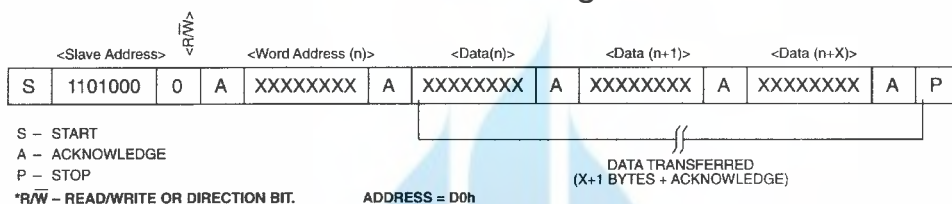
1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.
2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS1307 may operate in the following two modes:

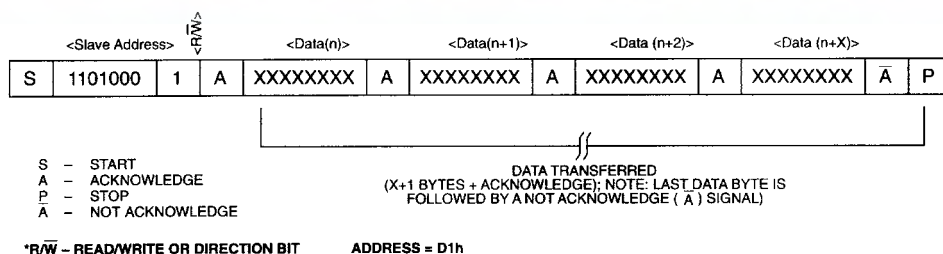
1. **Slave receiver mode (DS1307 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and *direction bit (See Figure 6). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7 bit DS1307 address, which is 1101000, followed by the *direction bit (R/\bar{W}) which, for a write, is a 0. After receiving and decoding the address byte the device outputs an acknowledge on the SDA line. After the DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307. This will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

DATA WRITE – SLAVE RECEIVER MODE Figure 6



2. **Slave transmitter mode (DS1307 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the *direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 7). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307 address, which is 1101000, followed by the *direction bit (R/\bar{W}) which, for a read, is a 1. After receiving and decoding the address byte the device inputs an acknowledge on the SDA line. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The DS1307 must receive a “not acknowledge” to end a read.

DATA READ – SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	-0.5V to +7.0V
Storage Temperature	-55°C to +125°C
Soldering Temperature	260°C for 10 seconds DIP See JPC/JEDEC Standard J-STD-020A for Surface Mount Devices

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

Range	Temperature	V _{CC}
Commercial	0°C to +70°C	4.5V to 5.5V V _{CC1}
Industrial	-40°C to +85°C	4.5V to 5.5V V _{CC1}

RECOMMENDED DC OPERATING CONDITIONS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V _{CC}	4.5	5.0	5.5	V	
Logic 1	V _{IH}	2.2		V _{CC} + 0.3	V	
Logic 0	V _{IL}	-0.5		+0.8	V	
V _{BAT} Battery Voltage	V _{BAT}	2.0		3.5	V	

*Unless otherwise specified.

DC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage (SCL)	I _{LI}			1	μA	
I/O Leakage (SDA & SQW/OUT)	I _{LO}			1	μA	
Logic 0 Output (I _{OL} = 5mA)	V _{OL}			0.4	V	
Active Supply Current	I _{CCA}			1.5	mA	7
Standby Current	I _{CCS}			200	μA	1
Battery Current (OSC ON); SQW/OUT OFF	I _{BAT1}		300	500	nA	2
Battery Current (OSC ON); SQW/OUT ON (32kHz)	I _{BAT2}		480	800	nA	
Power-Fail Voltage	V _{PF}	1.216 x V _{BAT}	1.25 x V _{BAT}	1.284 x V _{BAT}	V	8

*Unless otherwise specified.

AC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

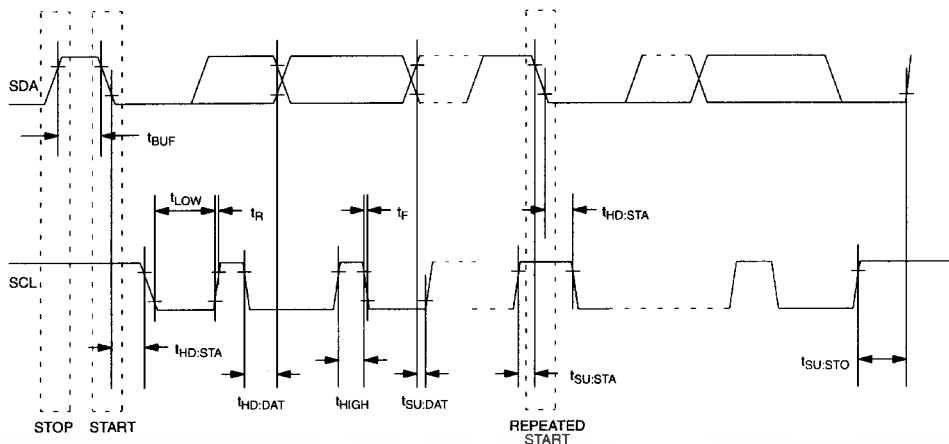
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			μs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			μs	3
LOW Period of SCL Clock	t_{LOW}	4.7			μs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			μs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			μs	
Data Hold Time	$t_{HD:DAT}$	0			μs	4,5
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			μs	
Capacitive Load for each Bus Line	C_B			400	pF	6
I/O Capacitance ($T_A = 25^\circ C$)	C_{IO}		10		pF	
Crystal Specified Load Capacitance ($T_A = 25^\circ C$)			12.5		pF	

*Unless otherwise specified.

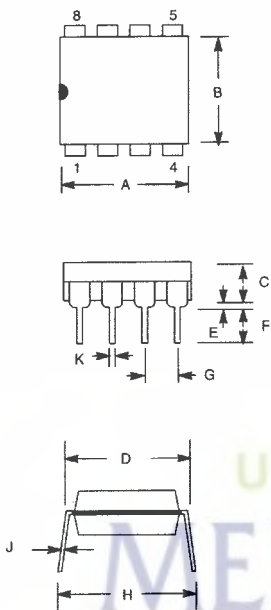
NOTES:

1. I_{CCS} specified with $V_{CC} = 5.0V$ and SDA, SCL = 5.0V.
2. $V_{CC} = 0V$, $V_{BAT} = 3V$.
3. After this period, the first clock pulse is generated.
4. A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
5. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
6. C_B – Total capacitance of one bus line in pF.
7. I_{CCA} – SCL clocking at max frequency = 100kHz.
8. V_{PF} measured at $V_{BAT} = 3.0V$.

TIMING DIAGRAM Figure 8

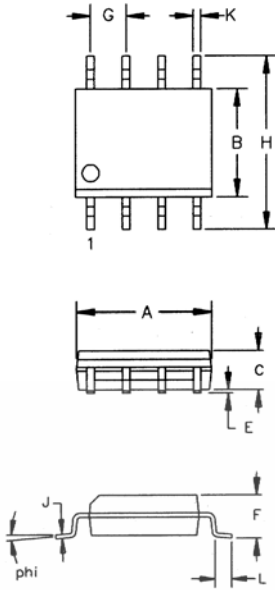


DS1307 64 X 8 SERIAL REAL-TIME CLOCK 8-PIN DIP MECHANICAL DIMENSIONS



PKG DIM	8-PIN	
	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

DS1307Z 64 X 8 SERIAL REAL-TIME CLOCK 8-PIN SOIC (150-MIL) MECHANICAL DIMENSIONS



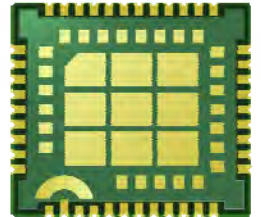
PKG	8-PIN (150 MIL)	
	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM	0.30	0.51
L IN.	0.016	0.050
MM	0.41	1.27
phi	0°	8°

56-G2008-001

UNIVERSITAS
MERCU BUANA



GSM/GPRS Module



SIM800C-DS

SIM800C-DS is a complete Quad-band GSM/GPRS solution with LCC and LGA pads, support Dual-SIM, which allows customers to use two SIM cards in one device simultaneously.

SIM800C-DS supports Quad-band 850/900/1800/1900MHz, it can transmit Voice, SMS and data information with low power consumption. With tiny size of 17.6*15.7*2.3mm, it can smoothly fit into slim and compact demands of customer design.

Smart Machine Smart Decision

General features

- Quad-band 850/900/1800/1900MHz
- GPRS multi-slot class 12/10
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
 - Class 4 (2 W @ 850/900MHz)
 - Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 17.6*15.7*2.3mm
- Weight: 1.3g
- Control via AT commands (3GPP TS 27.007, 27.005 and SIMCom enhanced AT Commands)
- Supply voltage range 3.4 ~ 4.4V
- Low power consumption
- Operation temperature:-40°C ~85°C

Specifications for GPRS Data

- GPRS class 12: max. 85.6 kbps (downlink/uplink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- PPP-stack
- USSD

Specifications for SMS via GSM/GPRS

- Point to point MO and MT
- SMS cell broadcast
- Text and PDU mode

Software features

- 0710 MUX protocol
- Embedded TCP/UDP protocol
- FTP/HTTP
- MMS
- POP3/SMTP
- DTMF
- Jamming Detection
- Audio Record
- SSL
- Bluetooth 3.0 (optional)
- EAT (optional)
- TTS_CN (optional)

Specifications for voice

- Tricodex
 - Half rate (HR)
 - Full rate (FR)
 - Enhanced Full rate (EFR)
- AMR
 - Half rate (HR)
 - Full rate (FR)
- Hands-free operation (Echo suppression)

Interfaces

- 77 SMT pins including
- SIM Card Interfaces (Dual Standby)
 - SIM card 1: 3V/ 1.8V
 - SIM card 2: 3V/ 1.8V
- Analog audio interface
- RTC backup
- I2C interface
- USB interface
- Serial interface
- PCM
- SD
- GPIO
- ADC
- GSM Antenna pad
- Bluetooth Antenna pad

Compatibility

- AT cellular command interface

More about SIMCom SIM800C-DS
Please contact:
Tel: 86-21-32523300
Fax: 86-21-32523301
Email: simcom@sim.com
Website: www.sim.com/wm

All specifications are subject to change without prior notice.