



SSG EMBEDDED SOLUTIONS

SSG

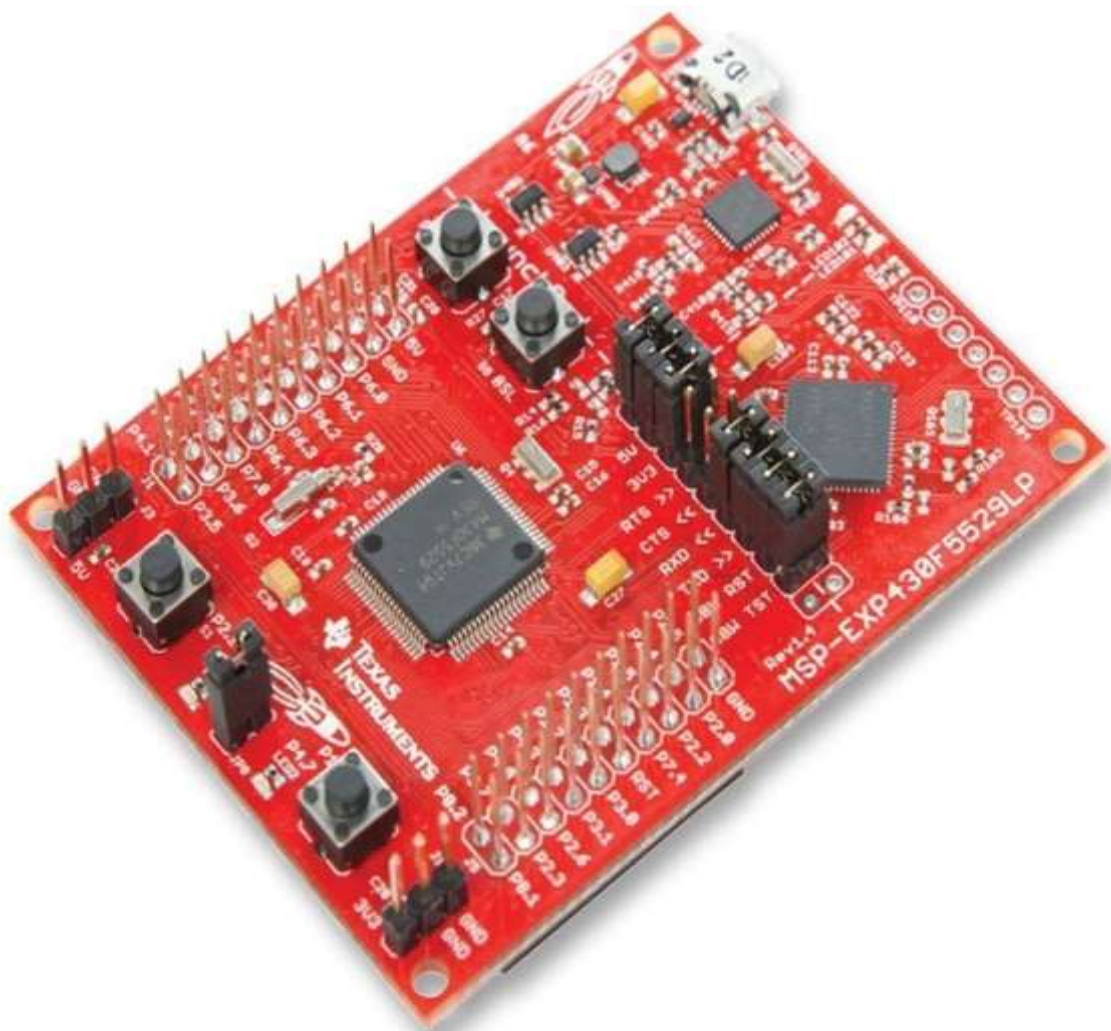
info@ssges.co.in



Microcontroller Development Kit

MSP-EXP430F5529LP

Documentation





List of Contents

1. Description of MSP430F5529LP

- MSP-EXP430F5529LP Launchpad board
- Pin diagram
- Feature
- Specification

2. Downloading and launching the Energia IDE

- Installation

3. User guide

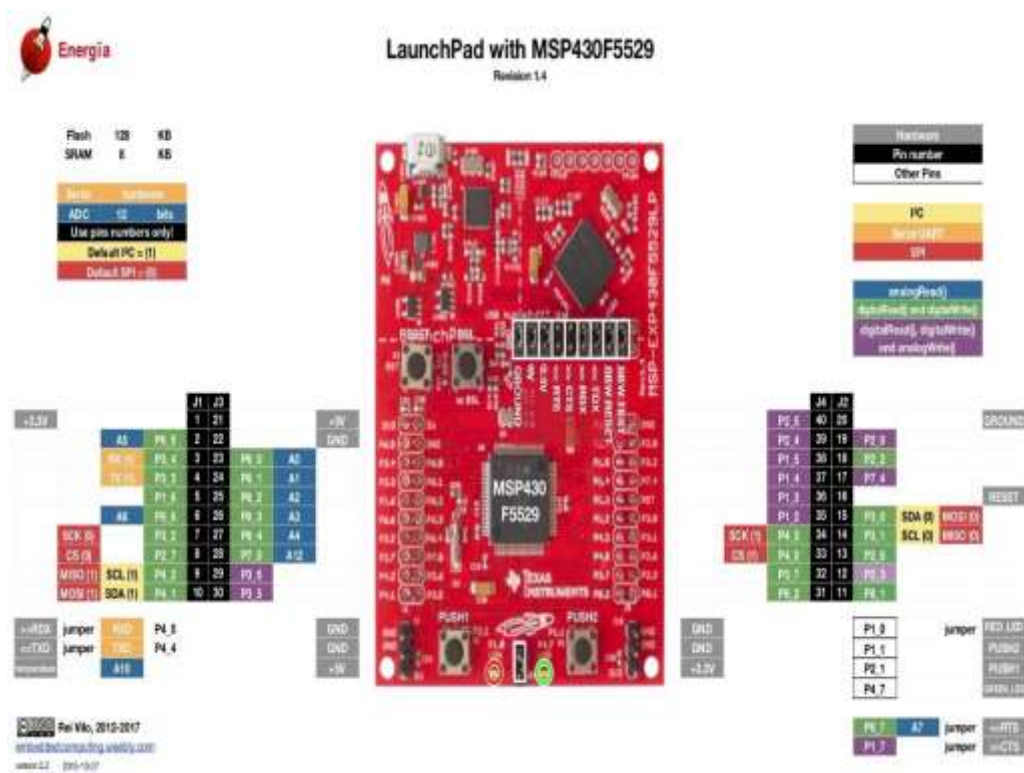
- Understanding the example Blink

4. Examples

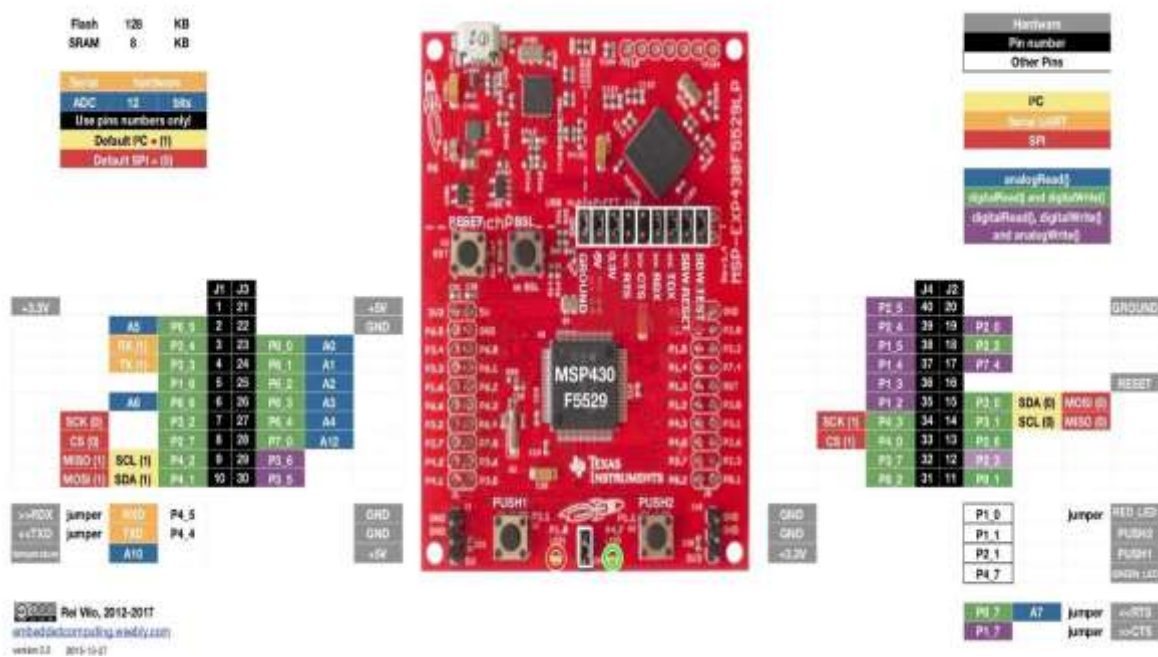
- LED Blinking
- LED Blinking with Toggling Switch Code
- LCD Display
- OLED Display
- Multiple Keypad
- DC Motor
- Stepper Motor
- RTC
- RS232 Serial Communication

MSP-EXP430F5529LP Launchpad board

The gorgeous red colour board is the MSP-EXP430F5529LP development Board. This board can program TI Microcontrollers that fall under the MSP430 series. The main purpose of this board is to upload code from the computer to the MCU and read serial data from the MCU for debugging purpose. It also provides the pin-out for each pin of the MCU and also two LEDs and a push button to make development easy.



Pin diagram



Features

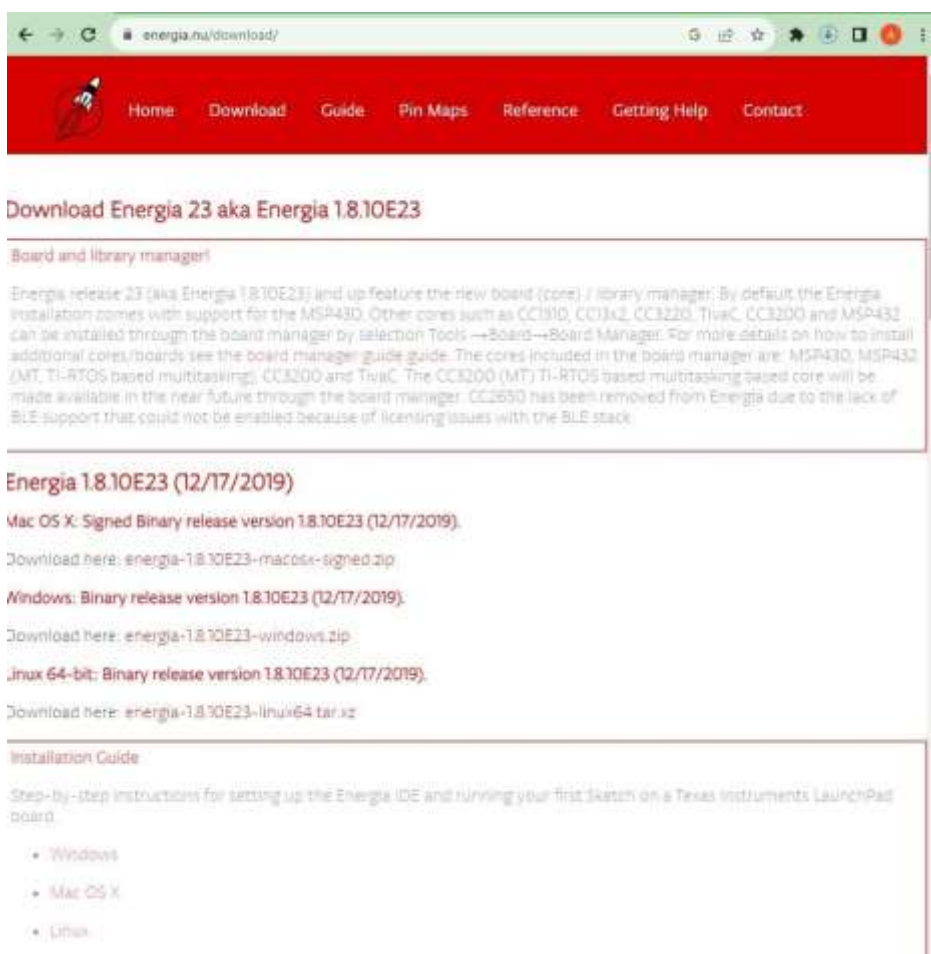
- 16-bit microcontroller, 1.8-V to 3.6-V operation Up to 25-MHz System Clock
- 128KB of flash, 8KB of RAM, Full-Speed USB 2.0
- Up to four serial interfaces (SPI, UART, I 2C), 12-bit analog-to-digital converter.
- Five timers
- USB Development Platform
- 4 push buttons (2x User Configured Pushbuttons, 1x Reset Pushbutton, 1x USB Bootstrap Pushbutton)
- 3 general purpose LEDs and 1 LED Power indicator.

Downloading and Launching the Energia IDE

Installation

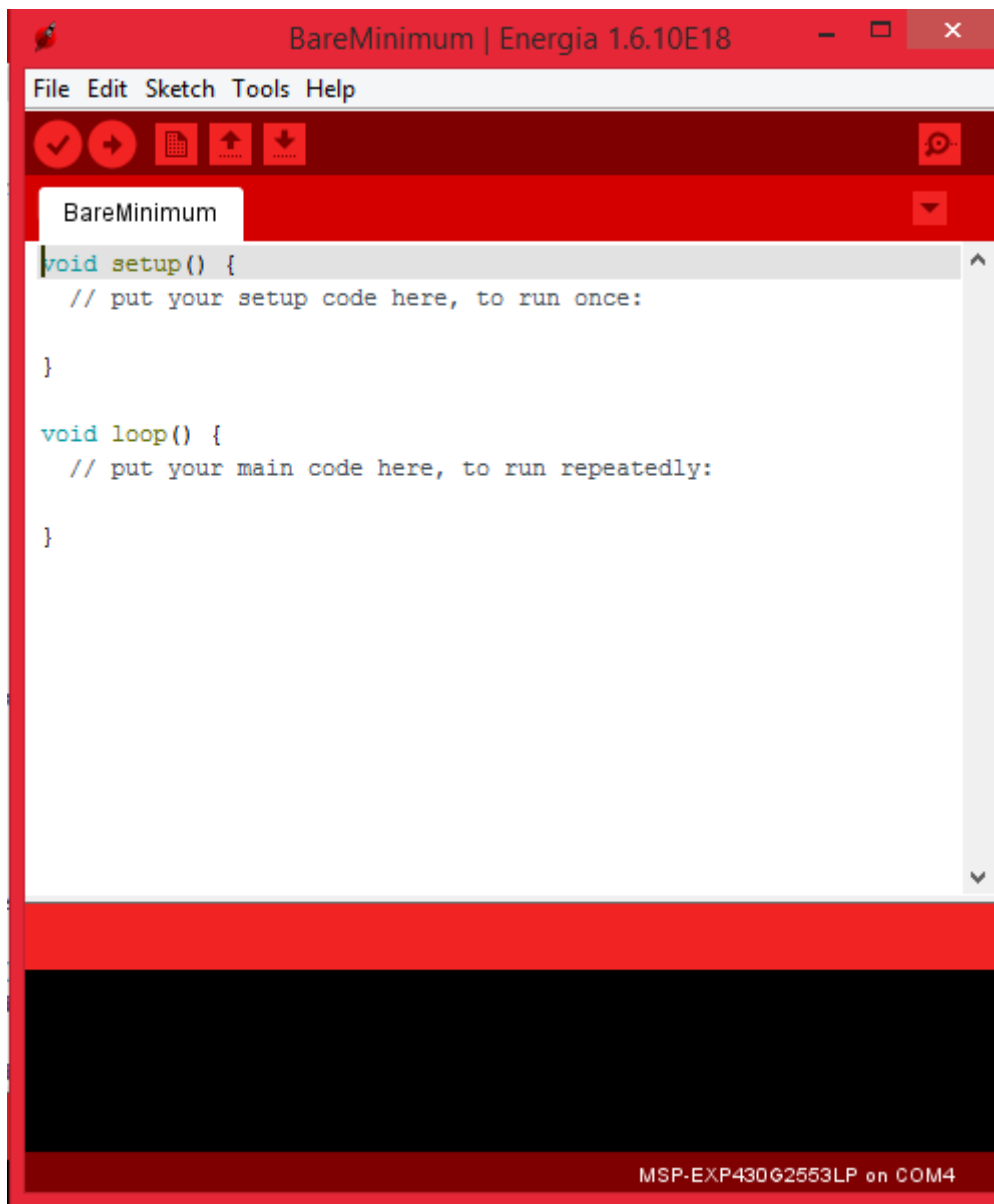
As said earlier the Energia is an open source and free Development environment, and it can be downloaded from this [Download link](#).

Select the Version based on your operating system, for windows you should notice a ZIP file being downloaded. After downloading the ZIP just extract it on your preferred location and open the folder. You find the application named Energia. Launch it and it should Look something like below.



Starting and Configuring Energia

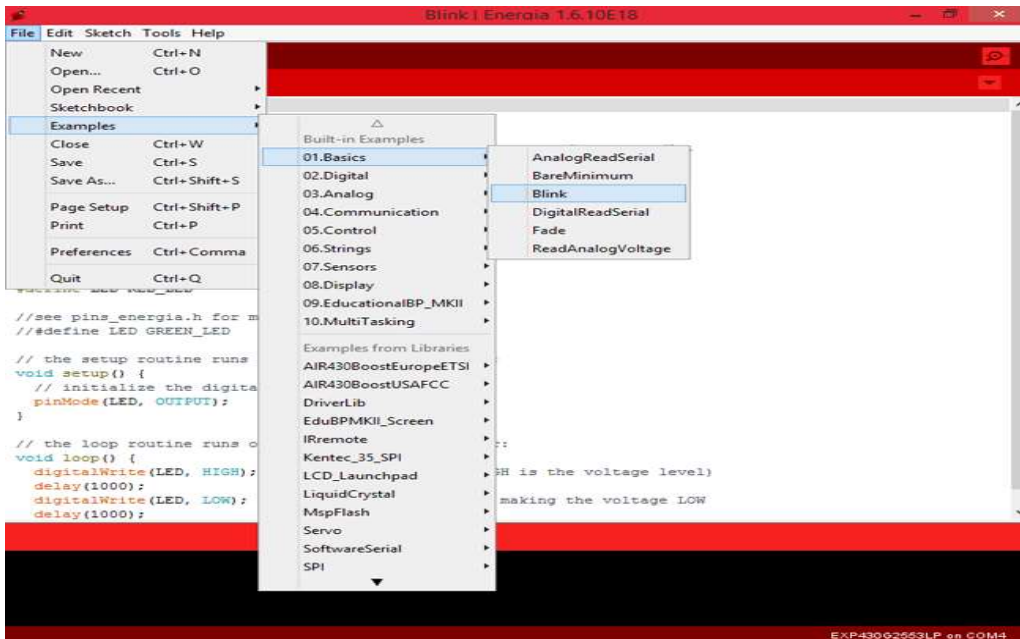
Double click Energia.exe (Windows users)
You find the application named Energia. Launch it and it should Look something like below.



The resemblance of both Arduino and Energia are same because they both are built on top of a platform called **Proce**

Understanding the example of Blink LED

To open this example program, navigate to *File->Example->Basics-> Blink* as shown below



The following program will appear on your IDE

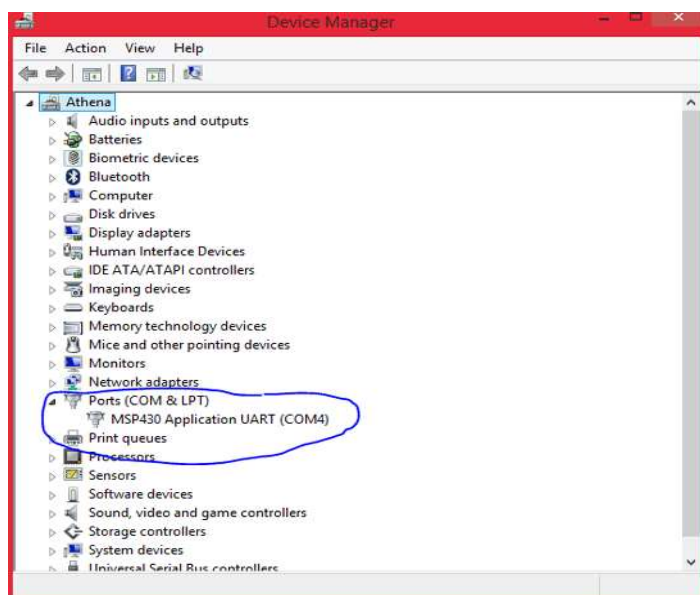
```

#define LED RED_LED
// the setup routine runs when you press reset:
void setup()
{
    pinMode(LED, OUTPUT); // Initialize the digital pin as an output
}
// the loop routine runs over and over again forever
void loop()
{
    digitalWrite(LED, HIGH); // turn the LED on
    delay (1000);           // wait for a second
    digitalWrite(LED, LOW); // turn the LED off
    delay (1000);           // wait for a second
}

```


Compiling and Uploading Blink Program

The next step would be to upload this program to our MSP board. To do this simply connect your board to the computer using the USB mini cable provided and wait for some time. The drivers for the board should start installing automatically. Then Open your device manager and under the COM ports option you should see your Board name like shown below



If you do not find the board discovered, go to the below Energia driver's link and install the appropriate drivers.

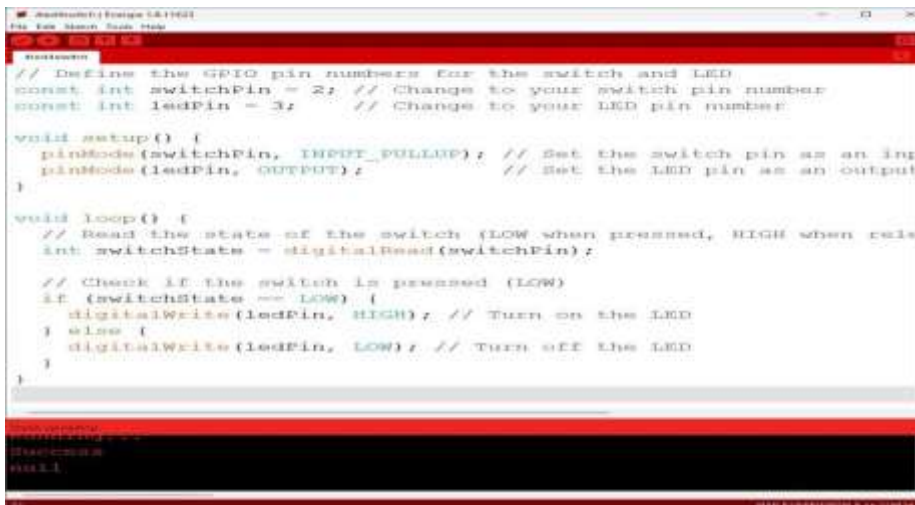
- [Windows](#)
- [Mac OS X](#)
- [Linux](#)

Once your board is discovered, make a note to which COM port it is connected to. Mine is connected to Port 4 here. Then go back to the Energia IDE and select *Tools* - > *Port* and select the same port I have select COM4 for me. Then again go to *Tools* - > *Boards* and select the MSP-EXP430G2553LP. Once done you should notice the following at the bottom right corner of your Energia IDE.



Now click on the upload icon on the top left corner (use the **video** at the bottom of this page if you are an absolute beginner) and your program should start uploading

to your board. If everything works fine you should notice the “Done uploading” message appearing on your IDE as shown below



```
const int switchPin = 2; // Change to your switch pin number
const int ledPin = 3; // Change to your LED pin number

void setup() {
  pinMode(switchPin, INPUT_PULLUP); // Set the switch pin as an input
  pinMode(ledPin, OUTPUT); // Set the LED pin as an output
}

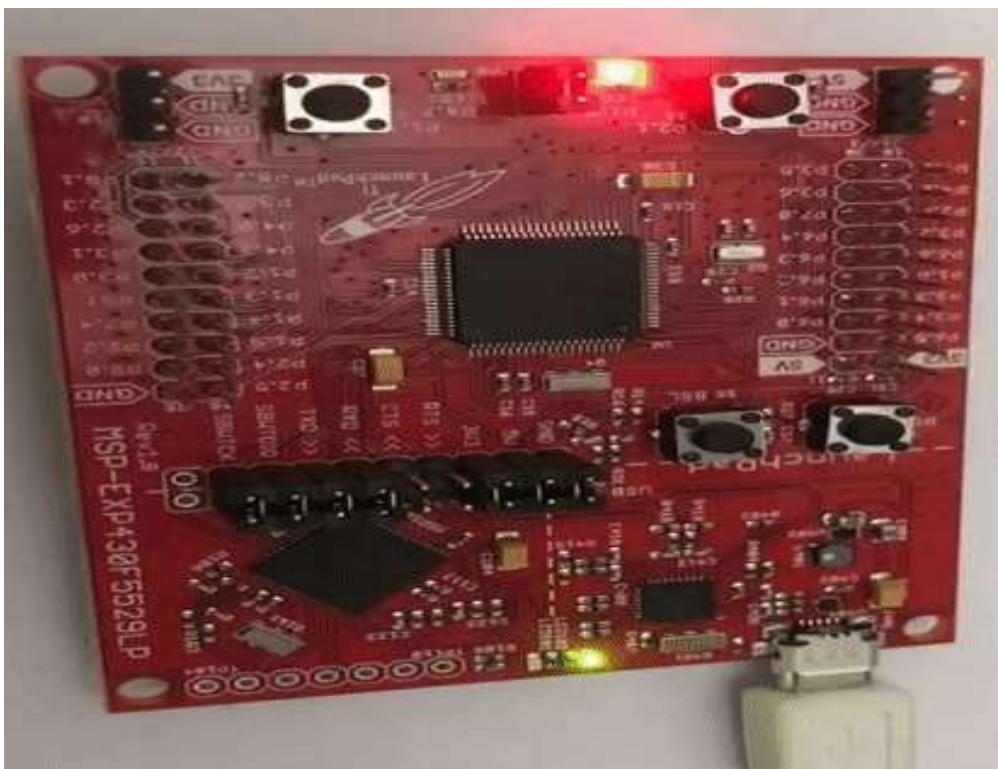
void loop() {
  // Read the state of the switch (LOW when pressed, HIGH when released)
  int switchState = digitalRead(switchPin);

  // Check if the switch is pressed (LOW)
  if (switchState == LOW) {
    digitalWrite(ledPin, HIGH); // Turn on the LED
  } else {
    digitalWrite(ledPin, LOW); // Turn off the LED
  }
}
```

uploading...

Verifying our Blink Output

Once the program is uploaded you can notice the LED on your Board blinking with a delay of 1000ms (1 sec) as programmed from our Energia IDE as shown below.

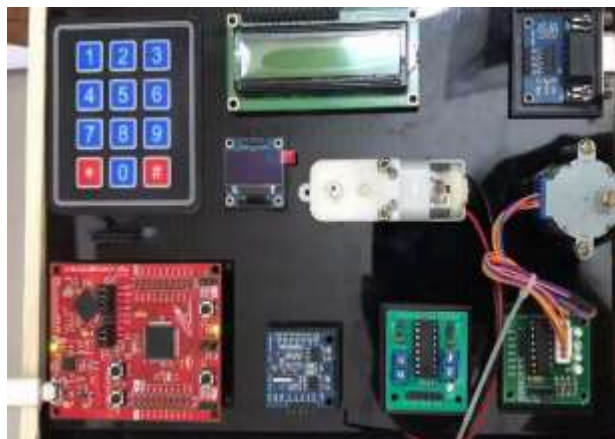


LED Blinking Code

```
#define LED RED_LED
#define LED GREEN_LED
//see pins_energia.h for more LED definitions
//#define LED GREEN_LED
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {

  digitalWrite(RED_LED, HIGH);
  delay(1000);
  digitalWrite(RED_LED, LOW);
  delay(1000);

  digitalWrite(GREEN_LED, HIGH);
  delay(1000);
  digitalWrite(GREEN_LED, LOW);
  delay(1000);
}
```



LED Blinking with Toggling Switch Code

```
#define LED1 RED_LED
#define LED2 GREEN_LED
#define BUTTON1 P2_1
#define BUTTON2 P1_1

int buttonState1 = 0;
int lastButtonState1 = 0;
int buttonState2 = 0;
int lastButtonState2 = 0;
bool ledState1 = LOW;
bool ledState2 = LOW;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(BUTTON1, INPUT_PULLUP);
  pinMode(LED2, OUTPUT);
  pinMode(BUTTON2, INPUT_PULLUP);
}

void loop() {
  buttonState1 = digitalRead(BUTTON1);
  buttonState2 = digitalRead(BUTTON2);

  if (buttonState1 != lastButtonState1) {
    // Check if the button state has changed
    if (buttonState1 == LOW) {
      // Button 1 is pressed
      ledState1 = !ledState1; // Toggle LED1 state
    }
  }
}
```

```
digitalWrite(LED1, ledState1);
}
delay(50); // Debounce delay
}

if (buttonState2 != lastButtonState2) {
  // Check if the button state has changed
  if (buttonState2 == LOW) {
    // Button 2 is pressed
    ledState2 = !ledState2; // Toggle LED2 state
    digitalWrite(LED2, ledState2);
  }
  delay(50); // Debounce delay
}

lastButtonState1 = buttonState1;
lastButtonState2 = buttonState2;
}
```



LCD Display

A liquid crystal display, better known as an LCD, is an excellent way for a microcontroller to present visible information. LCDs can display output from the μC such as time, date, and temperature; they can also be used to display the contents of memory, and aid in debugging programs.

LCDs are commonly used for portable electronic games, as viewfinders for digital cameras and camcorders, in video projection systems, for electronic billboards, as monitors for computers, and in flat-panel televisions.

Code

/*

LiquidCrystal Library - Hello World

Demonstrates the use a 16x2 LCD display. The LiquidCrystal library works with all LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This sketch prints "Hello World!" to the LCD and shows the time.

The circuit:

```
=====
```

LCD pin	Connect to
01 - GND	GND, pot
02 - VCC	+5V, pot
03 - Contrast	Pot wiper
04 - RS	Pin8 (P2.0)
05 - R/W	GND
06 - EN	Pin9 (P2.1)
07 - DB0	GND
08 - DB1	GND
09 - DB2	GND
10 - DB3	GND
11 - DB4	Pin10 (P2.2)
12 - DB5	Pin11 (P2.3)
13 - DB6	Pin12 (P2.4)
14 - DB7	Pin13 (P2.5)
15 - BL+	+5V
16 - BL-	GND

```
=====
```

Library originally added 18 Apr 2008

by David A. Mellis
library modified 5 Jul 2009
by Limor Fried (<http://www.ladyada.net>)
example added 9 Jul 2009
by Tom Igoe
modified 22 Nov 2010
by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/LiquidCrystal>
*/

```
// include the library code:  
#include <LiquidCrystal.h>
```

```
// initialize the library with the numbers of the interface pins  
LiquidCrystal lcd(P3_0, P3_1, P3_2, P3_3, P3_4, P3_5);
```

```
void setup() {  
  // set up the LCD's number of columns and rows:  
  lcd.begin(16, 2);  
  // Print a message to the LCD.  
  lcd.print("hello, world!");  
}
```

```
void loop() {  
  // set the cursor to column 0, line 1  
  // (note: line 1 is the second row, since counting begins with 0):  
  lcd.setCursor(0, 1);  
  // print the number of seconds since reset:  
  lcd.print(millis()/1000);  
}
```



Multiple Keypad

Matrix keypads are very common input devices in embedded systems. They have simple architecture and are easy to interface. One good thing about them is that they allow you to interface a large number of input keys to a microcontroller with minimum usage of I/O resources.

The big advantage of using a matrix keypad is that it allows to interface a large number of keys with a relatively small number of microcontroller pins.

Code

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {8, 7, 6}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins,
ROWS, COLS);

void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();

  if (customKey){
    Serial.println(customKey);
  }
}
```




DC Motor

DC motor usually refers to the particular type of motor through which the mechanical energy can be gained from the transformation of the electrical energy. A constant torque can be developed by DC motor; Rotor is a moving part of the DC motor. The mechanical movement of DC motor is generally created by it.

Code

```
byte M1A = 2;  
byte M1B = 3;  
byte M2A = 4;  
byte M2B = 5;  
  
void setup()  
{  
  pinMode(M1A, OUTPUT);  
  pinMode(M1B, OUTPUT);  
  pinMode(M2A, OUTPUT);  
  pinMode(M2B, OUTPUT);  
}
```

```
}  
  
void loop()  
{  
  digitalWrite(M1A, LOW);  
  digitalWrite(M1B, HIGH);  
  delay(2000);  
  digitalWrite(M1A, HIGH);  
  digitalWrite(M1B, HIGH);  
  delay(2000);  
  digitalWrite(M1A, HIGH);  
  digitalWrite(M1B, LOW);  
  delay(2000);  
  
  digitalWrite(M2A, LOW);  
  digitalWrite(M2B, HIGH);  
  delay(2000);  
  digitalWrite(M2A, HIGH);  
  digitalWrite(M2B, HIGH);  
  delay(2000);  
  digitalWrite(M2A, HIGH);  
  digitalWrite(M2B, LOW);  
  delay(2000);  
}
```



Stepper Motor

A Stepper Motor or a step motor is a brushless, synchronous motor, which divides a full rotation into a number of steps. Unlike a brushless DC motor, which rotates continuously when a fixed DC voltage is applied to it, a step motor rotates in discrete step angles.

The Stepper Motors therefore are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in stepping angles of 30, 15, 5, 2.5, 2, and 1.8 degrees per step.

Code

```
#include <Stepper.h>

const int stepsPerRevolution = 200; // change this to fit the number of steps per
revolution

// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8,9,10,11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(10);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one revolution in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
```

```
delay(500);
```

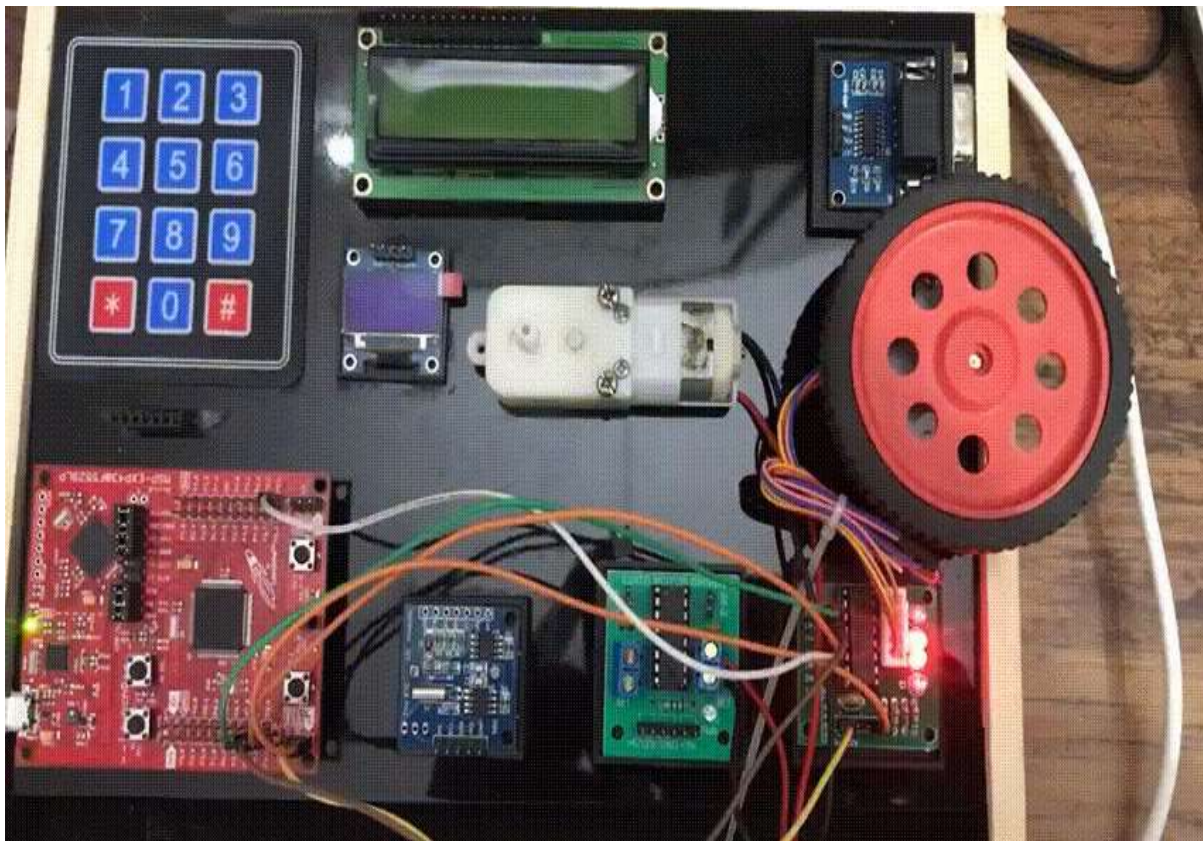
```
// step one revolution in the other direction:
```

```
Serial.println("counterclockwise");
```

```
myStepper.step(-stepsPerRevolution);
```

```
delay(500);
```

```
}
```



RTC

Real-Time Communications (RTC) refers to the ability to communicate and exchange information in real time over the internet. RTC includes technologies and protocols that enable synchronous communication of audio, video, and other types of data between devices.

RTC technologies are used in a wide range of applications, including voice and video calls, video conferencing, instant messaging, and live streaming

RTC has become increasingly important in recent years with the proliferation of smartphones and other devices that are capable of accessing the internet and running real-time communication applications. It has enabled new forms of communication and collaboration, such as remote work and online education, and it has also become an important part of many social and business interactions.

Code

```
/* Pin connection
SCL
SDA
*/
#include <Wire.h>
#include <DS1307.h>
DS1307 rtc;

void setup()
{
  //init Serial port
  Serial.begin(9600);
  while(!Serial); //wait for serial port to connect - needed for Leonardo only

  //init RTC
  Serial.println("Init RTC...");

  //only set the date+time one time
  // rtc.set(0, 0, 8, 24, 12, 2014); //08:00:00 24.12.2014 //sec, min, hour, day, month,
  year
```

```
//stop/pause RTC
// rtc.stop();

//start RTC
rtc.start();
}

void loop()
{
  uint8_t sec, min, hour, day, month;
  uint16_t year;

  //get time from RTC
  rtc.get(&sec, &min, &hour, &day, &month, &year);

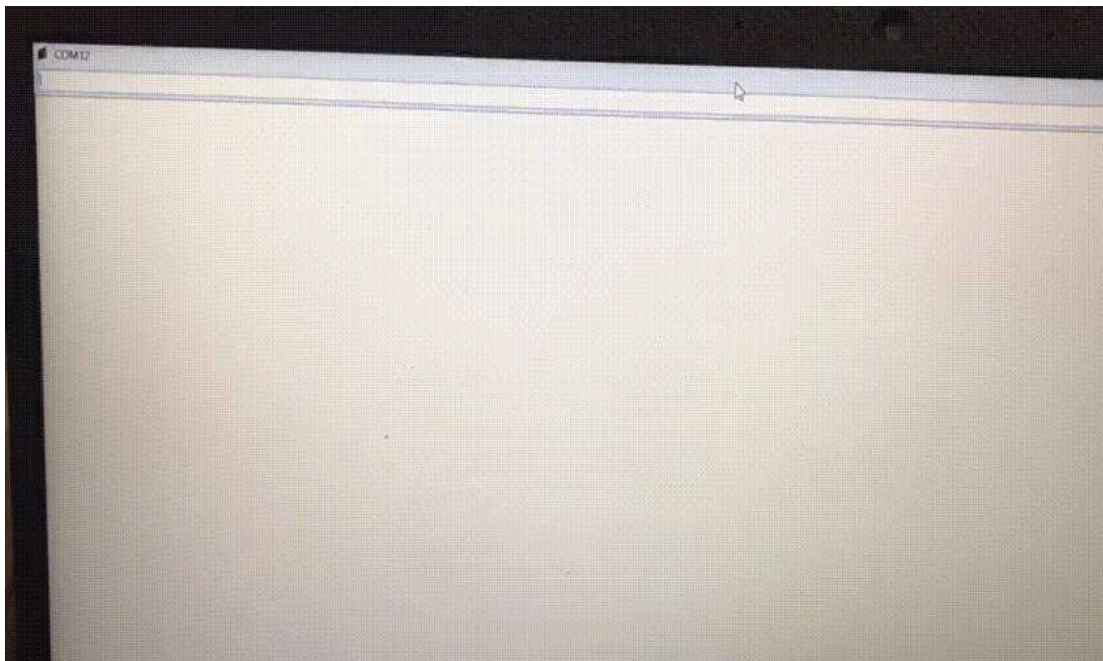
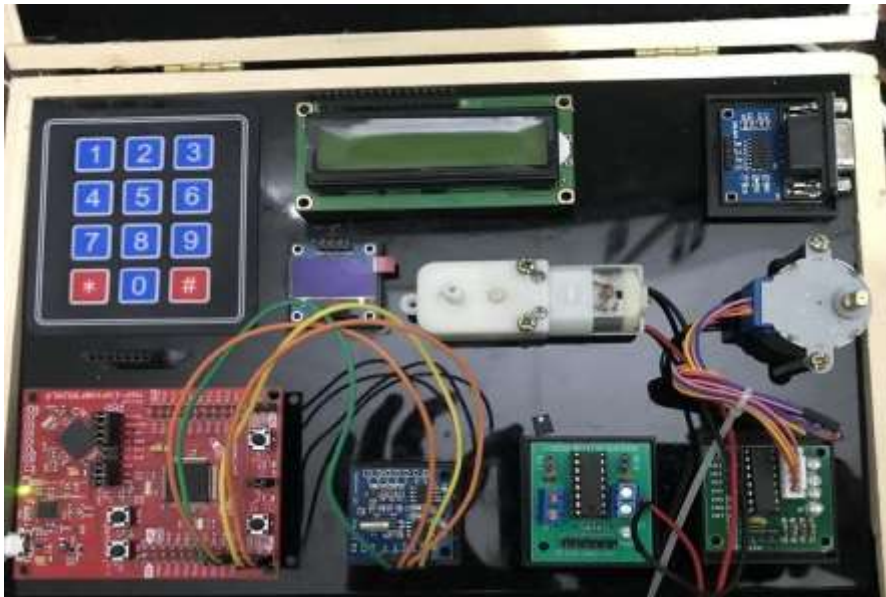
  //serial output
  Serial.print("\nTime: ");
  Serial.print(hour, DEC);
  Serial.print(":");
  Serial.print(min, DEC);
  Serial.print(":");
  Serial.print(sec, DEC);

  Serial.print("\nDate: ");
  Serial.print(day, DEC);
  Serial.print(".");
  Serial.print(month, DEC);
  Serial.print(".");
  Serial.print(year, DEC);
```

```
//wait a second
```

```
delay(1000);
```

```
}
```



RS232 Serial Communication

RS-232, commonly known as EIA-232, is a serial communication standard for data transmission between devices. It was first used in computer serial ports, modems, and other communication devices in the 1960s. RS-232 communicates over relatively small distances (usually up to 50 feet) at relatively moderate speeds (up to 115.2 kbps) using a single-ended signal. RS-232 additionally features handshaking signals for controlling data flow between devices.

RS-232 is often used for short-distance, low-speed communication

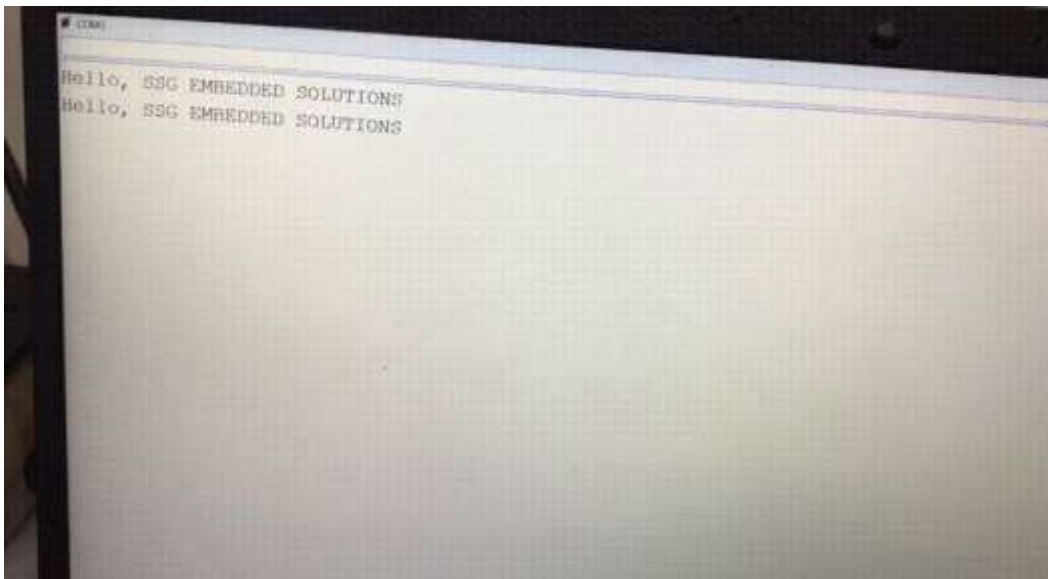
Code

```
/* Pin connection
RXD
TXD
*/
#include <Energia.h>

// Define the UART configuration
#define BAUD_RATE 9600 // Baud rate for RS232 communication

void setup() {
  // Initialize the UART module
  Serial.begin(BAUD_RATE);
}

void loop() {
  // Send a message over RS232
  Serial.println("Hello, SSG EMBEDDED SOLUTIONS");
  delay(1000); // Wait for 1 second before sending the next message
```

OLED Display

OLED is the acronym for **Organic Light Emitting Diode**. OLED is a modern display technology used in a wide range of electronic display devices, such as TVs, monitors, laptops, smartphones, bulletin boards, stadium screens, etc.

OLED displays consist of organic semiconductor compounds that emit a bright light on the passage of electric current through them, and hence it is termed as OLED. Since, OLED displays can emit light on their own, thus they are considered as self-emissive types of display. There is no need of backlight panel with LEDs to illuminate the screen.

The primary advantages of OLED displays include better picture quality, relatively wider viewing angles, greater flexibility in design, compact size, faster response time, and low power consumption.

Code

```
/* Pin connection
SCL
SDA
*/

#include <Wire.h>
#include "Font.h"
#include <string.h>
#include "images.h"

#define OLED_Write_Address 0x3C

void OLED_Data(char *DATA) /* Function for sending data to OLED */
{
    int len = strlen(DATA);
    for (int g=0; g<len; g++)
```

```

{
  for (int index=0; index<5; index++)
  {
    Wire.beginTransmission(OLED_Write_Address); /* Begin transmission to slave
device */

    /* Queue data to be transmitted */

    Wire.write(0x40); /* For Data Transmission, C = 0 and D/C = 1 */
    Wire.write(ASCII[DATA[g] - 0x20][index]);

    Wire.endTransmission(); /* Transmit the queued bytes and end transmission to
slave device */

  }
}
}

```

```

void OLED_Command(char DATA) /* Function for sending command to OLED */
{
  Wire.beginTransmission(OLED_Write_Address); /* Begin transmission to slave
device */

  /* Queue data to be transmitted */

  Wire.write(0x00); /* For Data Transmission, C = 0 and D/C = 0 */
  Wire.write(DATA);

  Wire.endTransmission(); /* Transmit the queued bytes and end transmission to
slave device */

}

```

```

void OLED_clear(void) /* Function for clearing OLED */
{
  OLED_setXY(0x00, 0x7F, 0x00, 0x07); /* Column Start Address 0, Column End
Address 127, Page Start Address 0, Page End Address 7 */

  for (int k=0; k<=1023; k++)
  {

```

```
Wire.beginTransmission(OLED_Write_Address); /* Begin transmission to slave device */
```

```
/* Queue data to be transmitted */
```

```
Wire.write(0x40); /* For Data Transmission, C = 0 and D/C = 1 */
```

```
Wire.write(0x00);
```

```
Wire.endTransmission(); /* Transmit the queued bytes and end transmission to slave device */
```

```
}
```

```
}
```

```
void OLED_setXY(char col_start, char col_end, char page_start, char page_end) /* Function for setting cursor for writing data */
```

```
{
```

```
Wire.beginTransmission(OLED_Write_Address); /* Begin transmission to slave device */
```

```
/* Queue data to be transmitted */
```

```
Wire.write(0x00); /* For Data Transmission, C = 0 and D/C = 0 */
```

```
Wire.write(0x21); /* Set Column Start and End Address */
```

```
Wire.write(col_start); /* Column Start Address col_start */
```

```
Wire.write(col_end); /* Column End Address col_end */
```

```
Wire.write(0x22); /* Set Page Start and End Address */
```

```
Wire.write(page_start); /* Page Start Address page_start */
```

```
Wire.write(page_end); /* Page End Address page_end */
```

```
Wire.endTransmission(); /* Transmit the queued bytes and end transmission to slave device */
```

```
}
```

```
void OLED_init(void) /* Function for initializing OLED */
```

```
{
```

```
OLED_Command(0xAE); /* Entire Display OFF */
```

```
OLED_Command(0xD5); /* Set Display Clock Divide Ratio and Oscillator Frequency */
```

```

    OLED_Command(0x80); /* Default Setting for Display Clock Divide Ratio and
    Oscillator Frequency that is recommended */
    OLED_Command(0xA8); /* Set Multiplex Ratio */
    OLED_Command(0x3F); /* 64 COM lines */
    OLED_Command(0xD3); /* Set display offset */
    OLED_Command(0x00); /* 0 offset */
    OLED_Command(0x40); /* Set first line as the start line of the display */
    OLED_Command(0x8D); /* Charge pump */
    OLED_Command(0x14); /* Enable charge dump during display on */
    OLED_Command(0x20); /* Set memory addressing mode */
    OLED_Command(0x00); /* Horizontal addressing mode */
    OLED_Command(0xA1); /* Set segment remap with column address 127 mapped
    to segment 0 */
    OLED_Command(0xC8); /* Set com output scan direction, scan from com63 to
    com 0 */
    OLED_Command(0xDA); /* Set com pins hardware configuration */
    OLED_Command(0x12); /* Alternative com pin configuration, disable com left/right
    remap */
    OLED_Command(0x81); /* Set contrast control */
    OLED_Command(0x80); /* Set Contrast to 128 */
    OLED_Command(0xD9); /* Set pre-charge period */
    OLED_Command(0xF1); /* Phase 1 period of 15 DCLK, Phase 2 period of 1 DCLK
    */
    OLED_Command(0xDB); /* Set Vcomh deselect level */
    OLED_Command(0x20); /* Vcomh deselect level ~ 0.77 Vcc */
    OLED_Command(0xA4); /* Entire display ON, resume to RAM content display */
    OLED_Command(0xA6); /* Set Display in Normal Mode, 1 = ON, 0 = OFF */
    OLED_Command(0x2E); /* Deactivate scroll */
    OLED_Command(0xAF); /* Display on in normal mode */
}

```

```

void OLED_image(const unsigned char *image_data) /* Function for sending image
data to OLED */
{
    OLED_setXY(0x00, 0x7F, 0x00, 0x07);
    for (int k=0; k<=1023; k++)
    {
        Wire.beginTransmission(OLED_Write_Address); /* Begin transmission to slave
device */

        /* Queue data to be transmitted */

        Wire.write(0x40); /* For Data Transmission, C = 0 and D/C = 1 */
        Wire.write(image_data[k]);

        Wire.endTransmission(); /* Transmit the queued bytes and end transmission to
slave device */
    }
}

```

```

void setup() {
    Wire.begin(); /* Initiate wire library and join I2C bus as a master */
    OLED_init(); /* Initialize OLED */
    delay(100);
    OLED_clear(); /* Clear OLED */
    delay(1000);
    OLED_image(Launchpad_Logo);
    delay(2000);
    OLED_clear();
    delay(200);
    OLED_setXY(0x31, 0x7F, 0x03, 0x02);
    OLED_Data("Smiley");
    OLED_setXY(0x36, 0x7F, 0x04, 0x03);
    OLED_Data("Demo");
}

```

```
OLED_setXY(0x00, 0x7F, 0x00, 0x07);
delay(2000);
}
```

```
void loop() {
  OLED_image(Smile_1);
  delay(200);
  OLED_image(Smile_2);
  delay(200);
  OLED_image(Smile_3);
  delay(200);
  OLED_image(Smile_4);
  delay(200);
}
```

Font.h

```
static const char ASCII[][5] =
{
  {0x00, 0x00, 0x00, 0x00, 0x00} // 20 (Space)
  ,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
  ,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
  ,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
  ,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
  ,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
  ,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
  ,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
  ,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
  ,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
  ,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
```

,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f /
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H

,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x02, 0x04, 0x08, 0x10, 0x20} // 5c ¥
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x01, 0x02, 0x04, 0x00} // 60 `~
,{0x20, 0x54, 0x54, 0x54, 0x78} // 61 a
,{0x7f, 0x48, 0x44, 0x44, 0x38} // 62 b
,{0x38, 0x44, 0x44, 0x44, 0x20} // 63 c
,{0x38, 0x44, 0x44, 0x48, 0x7f} // 64 d
,{0x38, 0x54, 0x54, 0x54, 0x18} // 65 e
,{0x08, 0x7e, 0x09, 0x01, 0x02} // 66 f

```

,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 67 g
,{0x7f, 0x08, 0x04, 0x04, 0x78} // 68 h
,{0x00, 0x44, 0x7d, 0x40, 0x00} // 69 i
,{0x20, 0x40, 0x44, 0x3d, 0x00} // 6a j
,{0x7f, 0x10, 0x28, 0x44, 0x00} // 6b k
,{0x00, 0x41, 0x7f, 0x40, 0x00} // 6c l
,{0x7c, 0x04, 0x18, 0x04, 0x78} // 6d m
,{0x7c, 0x08, 0x04, 0x04, 0x78} // 6e n
,{0x38, 0x44, 0x44, 0x44, 0x38} // 6f o
,{0x7c, 0x14, 0x14, 0x14, 0x08} // 70 p
,{0x08, 0x14, 0x14, 0x18, 0x7c} // 71 q
,{0x7c, 0x08, 0x04, 0x04, 0x08} // 72 r
,{0x48, 0x54, 0x54, 0x54, 0x20} // 73 s
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 77 w
,{0x44, 0x28, 0x10, 0x28, 0x44} // 78 x
,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 79 y
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ->
,{0x78, 0x46, 0x41, 0x46, 0x78} // 7f <-
};

```

Image.h

```

const unsigned char Launchpad_Logo[1024] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0,
0xC0, 0xE0, 0xE0,
0xE0, 0xF0, 0xD0, 0x98, 0x18, 0x0C, 0x0C, 0x04, 0x06, 0x06, 0x02, 0x02, 0x03,
0x03, 0x03, 0x03,
0x03, 0x03, 0x83, 0xE6, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0x80, 0xC0, 0x40, 0x40, 0x60, 0x20, 0x20, 0x20, 0x20, 0x30,
0x20, 0xA0, 0xE0,
0xE0, 0x60, 0x30, 0x30, 0x18, 0x0C, 0x0C, 0x1E, 0x3E, 0x7F, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFC, 0xF0, 0xC0, 0x00, 0x00, 0x80, 0xC0,
0x60, 0x30, 0x18,
0x0C, 0x07, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0x30, 0x38,
0x28, 0x24, 0x16,
0x12, 0x11, 0x11, 0x30, 0x60, 0xC0, 0x80, 0xE0, 0x70, 0x38, 0x1C, 0x0C, 0x06,
0x03, 0x01, 0x80,
0xC0, 0xC0, 0xE0, 0xE0, 0x70, 0x78, 0x38, 0x00, 0x00, 0x00, 0x00, 0x01, 0x07,
0x1F, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x03, 0x01, 0x00, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0x80,
0xC0, 0x40, 0x60, 0x20, 0x20, 0x30, 0x10, 0x10, 0x18, 0x98, 0x88, 0x80, 0xC0,
0x40, 0x60, 0x20,
0x20, 0x30, 0x90, 0x80, 0xC0, 0xE0, 0x63, 0x76, 0x38, 0x38, 0xDC, 0x9C, 0x0E,
0x07, 0x07, 0x83,
0x83, 0x81, 0xC0, 0xC0, 0x40, 0x60, 0x20, 0x30, 0x90, 0xF8, 0x08, 0x0C, 0x06,
0x06, 0x03, 0x01,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x80, 0xC0, 0x40, 0x60, 0x30, 0x10, 0x18, 0x0C, 0x04, 0x86, 0x82, 0xC3,
0x41, 0x61, 0x30,
0x10, 0x18, 0x08, 0x0C, 0x04, 0x06, 0x02, 0x03, 0x01, 0x01, 0x00, 0x00, 0x80,
0xC0, 0x60, 0x30,
0x00, 0x03, 0x03, 0x81, 0xC1, 0x60, 0x30, 0x00, 0x00, 0x00, 0x00, 0x03, 0xC7,
0x7B, 0x01, 0x01,

0x01, 0x80, 0x80, 0x40, 0x20, 0x30, 0x18, 0x06, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x18,
0x0C, 0x06, 0x02,

0x83, 0xC1, 0x60, 0x20, 0x10, 0x18, 0x0C, 0x04, 0x02, 0x03, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x80, 0x80, 0x40, 0x60, 0x20, 0x30, 0x18, 0x0C, 0x06, 0x02, 0x83, 0xC1,
0x60, 0x30, 0x18,

0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x18, 0x1C, 0x0E, 0x0B, 0x0D,
0x04, 0x02, 0x02,

0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0xC0, 0x60, 0x30, 0x18, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x60, 0x30, 0x18,
0x0C, 0x06, 0x03,

0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0x18, 0x08, 0x0C, 0x04,
0x06, 0x02, 0x83,

0x81, 0xC1, 0x40, 0x60, 0x30, 0x10, 0x18, 0x0C, 0x04, 0x02, 0x03, 0x01, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x60,
0x70, 0x5C, 0x46,

0x63, 0x61, 0x20, 0x20, 0x30, 0x30, 0x10, 0x10, 0x18, 0x08, 0x08, 0x0C, 0xE4,
0xFE, 0x4E, 0x42,

0x60, 0x60, 0x20, 0x30, 0x30, 0x10, 0x18, 0x08, 0x08, 0x0C, 0x04, 0x06, 0x02,
0x03, 0x01, 0x01,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00

};

```

```

const unsigned char Smiley_1[1024] = {

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80,
0x80, 0xC0, 0xC0,

0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0,

0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0,

0xC0, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0x78,
0x78, 0x78, 0x3C,
0x3C, 0x1C, 0x1E, 0x0E, 0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03,
0x03, 0x03, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01,
0x03, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x07, 0x0F, 0x0E, 0x0E, 0x1E,
0x1C, 0x3C, 0x3C,
0x38, 0x78, 0x78, 0xF0, 0xF0, 0xE0, 0xE0, 0xC0, 0xC0, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0xC0, 0xE0, 0xF0,
0xF8, 0xFC, 0x7E, 0x3F, 0x1F, 0x0F, 0x07, 0x07, 0x03, 0x01, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80,
0x80, 0x80, 0x80,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x1F, 0x3F, 0x7E,
0xFC, 0xF8, 0xF8,
0xF0, 0xE0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFC, 0xFF, 0xFF,
0xFF, 0xFF, 0x1F,

0x07, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x0F, 0x0F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F,

0x1F, 0x0F, 0x0F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06,
0x0F, 0x0F, 0x1F,

0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x0F, 0x0F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x03, 0x1F,

0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x3F, 0xFF, 0xFF,
0xFF, 0xFF, 0xF8,

0xE0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0xC0, 0xF8,

0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x03, 0x07, 0x0F,

0x1F, 0x3F, 0x7E, 0xFC, 0xF8, 0xF0, 0xE0, 0xE0, 0xC0, 0x80, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x03, 0x07, 0x07, 0x07, 0x0F,
0x0E, 0x1E, 0x1E,

0x1C, 0x1C, 0x3C, 0x38, 0x38, 0x38, 0x38, 0x38, 0x38, 0x70, 0x70, 0x70, 0x70,
0x70, 0x70, 0x70,
0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0x38, 0x38, 0x38, 0x38, 0x38, 0x3C,
0x1C, 0x1C, 0x1E,
0x1E, 0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0x7E,
0x3F, 0x1F, 0x1F,
0x0F, 0x07, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x1F,
0x1E, 0x1E, 0x3C,
0x3C, 0x38, 0x78, 0x70, 0x70, 0xF0, 0xF0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80,
0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xF0, 0x70, 0x70, 0x78,
0x38, 0x3C, 0x3C,
0x1C, 0x1E, 0x1E, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
0x01, 0x03, 0x03,
0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03,

```
0x03, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00
};
```

```
const unsigned char Smiley_2[1024] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x70, 0x70, 0x70, 0x70,
0x70, 0x70, 0x70,
0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xC0,
0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0xF8, 0x78,
0x3C, 0x3C, 0x1C,
0x1E, 0x1E, 0x0E, 0x0F, 0x07, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03, 0x03, 0x01,
0x01, 0x01, 0x01,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0x01,
```

0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x07, 0x0F, 0x0E,
0x1E, 0x1E, 0x1C,
0x3C, 0x3C, 0x78, 0x78, 0xF0, 0xF0, 0xE0, 0xE0, 0xC0, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0xE0, 0xF0, 0xF8,
0xFC, 0xFE, 0x7E, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F,
0xFE, 0xFC, 0xF8,
0xF0, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xFC, 0xFF, 0xFF,
0xFF, 0xFF, 0x1F,
0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x0E, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03,
0x03, 0x07, 0x0F, 0x0F, 0x0F, 0x0F, 0x0E, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0E, 0x0E, 0x0F, 0x0F, 0x0F,
0x07, 0x03, 0x03,
0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07, 0x0F, 0x0F, 0x0F, 0x0F, 0x0E, 0x0C,
0x08, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0x1F,

0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x1F, 0x7F, 0xFF,
0xFF, 0xFF, 0xF8,

0xE0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0xC0, 0xF8,

0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x03, 0x07, 0x0F,

0x1F, 0x3F, 0x7E, 0xFC, 0xF8, 0xF0, 0xE0, 0xE0, 0xC0, 0x80, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x03, 0x07, 0x07, 0x0F, 0x0F,
0x0E, 0x1E, 0x1E,

0x1C, 0x1C, 0x3C, 0x38, 0x38, 0x38, 0x38, 0x38, 0x70, 0x70, 0x70, 0x70, 0x70,
0x70, 0x70, 0x70,

0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0x78, 0x38, 0x38, 0x38, 0x38, 0x3C,
0x1C, 0x1C, 0x1E,

0x1E, 0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0x7E,
0x3F, 0x3F, 0x1F,

0x0F, 0x07, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x1E,
0x1E, 0x3C, 0x3C,
0x38, 0x78, 0x78, 0x70, 0xF0, 0xF0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0,
0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x80, 0x80, 0x80,
0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xE0, 0xE0, 0xF0, 0xF0, 0x70, 0x78,
0x78, 0x38, 0x3C,
0x3C, 0x1E, 0x1E, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01,
0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x03,
0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00
};

```

```

const unsigned char Smiley_3[1024] = {

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0,

0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x70, 0x70, 0x70, 0x70,
0x70, 0x70, 0x70,

0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0,

0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF0, 0xF0, 0x78, 0x78,
0x3C, 0x3C, 0x1E,

0x1E, 0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01,
0x01, 0x01, 0x01,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01,

0x01, 0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x0F, 0x0F,
0x1E, 0x1E, 0x1E,

0x3C, 0x3C, 0x78, 0x78, 0xF8, 0xF0, 0xF0, 0xE0, 0xC0, 0xC0, 0x80, 0x80, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0,
0xE0, 0xF8, 0xFC,

0xFC, 0x7E, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80,

0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x07, 0x0F, 0x1F, 0x3F,
0x7E, 0xFC, 0xFC,

0xF8, 0xF0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF8, 0xFE, 0xFF, 0xFF,
0xFF, 0xFF, 0x07,

0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0,
0xE0, 0xE0, 0xE0,

0xE0, 0xC0, 0xC0, 0xC0, 0x80, 0x00, 0x00, 0x06, 0x0F, 0x0F, 0x0F, 0x1F, 0x1F,
0x1F, 0x1F, 0x1F,

0x0F, 0x0F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06,
0x0F, 0x0F, 0x1F,

0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x0F, 0x0F, 0x06, 0x00, 0x00, 0x80, 0xC0, 0xC0,
0xC0, 0xE0, 0xE0,

0xE0, 0xE0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0x07,

0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x7F, 0xFF, 0xFF,
0xFF, 0xFF, 0xF0,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0,
0xE0, 0xE0, 0xE1,

0xF1, 0xFB, 0xFF, 0x7F, 0x7F, 0x3F, 0x3F, 0x3C, 0x38, 0x78, 0x78, 0x70, 0xF0,
0xE0, 0xE0, 0xE0,

0xE0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80,

0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0,
0xC0, 0xE0, 0xE0,

0xE0, 0xE0, 0xF0, 0x70, 0x78, 0x78, 0x38, 0x3C, 0x3F, 0x3F, 0x7F, 0x7F, 0xFF,
0xFB, 0xF1, 0xE1,

0xE0, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0xE0,

0xFE, 0xFF, 0xFF, 0xFF, 0x7F, 0x1F, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03,
0x07, 0x0F, 0x1F,

0x3F, 0x7F, 0x7E, 0xFC, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x80, 0x00, 0x00, 0x01,
0x01, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03,

0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x01, 0x01, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xE0, 0xF0, 0xF0, 0xF8, 0xFC,
0x7F, 0x3F, 0x1F,

0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x0F, 0x1E,
0x1E, 0x3C, 0x3C,

0x38, 0x78, 0x78, 0x70, 0xF0, 0xE0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0,
0x80, 0x80, 0x80,

0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x80, 0x80, 0x80,

0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xF0, 0x70, 0x70,
0x78, 0x38, 0x3C,


```

0x3C, 0x1E, 0x1E, 0x1F, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x03,
0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
};

```

```

const unsigned char Smiley_4[1024] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x70, 0x70, 0x70,
0x70, 0x70, 0x70,
0x70, 0x70, 0x70, 0x70, 0xF0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xC0,
0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0xF8, 0x78, 0x78,
0x3C, 0x3C, 0x1E,
0x1E, 0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01,
0x01, 0x01, 0x01,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0x01,
0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x0F, 0x0F, 0x0E,
0x1E, 0x1E, 0x3C,
0x3C, 0x3C, 0x78, 0x78, 0xF0, 0xF0, 0xE0, 0xE0, 0xC0, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xE0,
0xF0, 0xF8, 0xFC,
0xFE, 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x0F, 0x0F, 0x1F, 0x7F,
0xFE, 0xFC, 0xF8,
0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0xFE, 0xFF, 0xFF, 0xFF,
0xFF, 0x0F, 0x03,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x0F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F,

0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06,
0x0F, 0x1F, 0x1F,

0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0x07,

0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x7F, 0xFF, 0xFF, 0xFF,
0xFF, 0xF0, 0x80,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x0C, 0x0C, 0x1C, 0x3E, 0x3E, 0x7C, 0x78, 0x78, 0xF0, 0xF0,
0xE0, 0xE0, 0xE0,

0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0,
0xC0, 0xC0, 0xE0,

0xE0, 0xF0, 0xF0, 0x78, 0x78, 0x7C, 0x3C, 0x3E, 0x1C, 0x1C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0xE0,

0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07,
0x0F, 0x1F, 0x3F,

0x7F, 0xFC, 0xF8, 0xF0, 0xE0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
0x01, 0x01, 0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x87, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x87, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x03, 0x01, 0x01, 0x01, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0x7E, 0x7F, 0x3F,
0x1F, 0x0F, 0x07, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x1F, 0x1E, 0x3E, 0x3C, 0x3C, 0x78, 0x78,
0x70, 0xF0, 0xF0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07, 0x0F, 0x0F, 0x0E, 0x0E, 0x0E,
0x0E, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x03, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0x80, 0x80, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xE0, 0xE0, 0xF0, 0x70, 0x78, 0x78,
0x3C, 0x3C, 0x3C, 0x1E, 0x1F, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x0F, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E,
0x0E, 0x0E, 0x0E,

```
0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0E, 0x0F, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07,
0x07, 0x03, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00
};
```

