Final Project

Tuesday, November 3, 2020 9:19 PM

200/200

BILKEYSIGHT
TECHNOLOGIES

Impressive!

WS-1 Oscilloscope Technical Reference

A Bilkey Electric Company

Demo Video: https://msoe.app.box.com/file/742780287317

John Bilkey

EE 2920 – 21

Final Project - Oscilloscope
November 17, 2020

Objectives:

Demonstrate mastery of course material with a multi-week project, the WS-1 (Widder Scope 1).

Description:

I first made my graphics and converted them to C header files with a program I made using Scratch. I then dug around Dr. Johnson's LCD functions and modified them slightly so they would have more precision for selecting positions on the LCD. Next I set up A2D, then made my array reading/recording and printing functions. Finally, I came up with equations to find positions the in the LCD for voltage values. The large amount of parentheses and crazy values included in my calculations made that quite the challenge. I considered making a ton of if statements before and while creating my equations, but I am glad I stuck with figuring out the math.

I initially tried setting up a full bridge rectifier and inverting op-amp to read AC voltages with no DC offset, but the op-amp caused issues with my LCD so I had to scrap that.

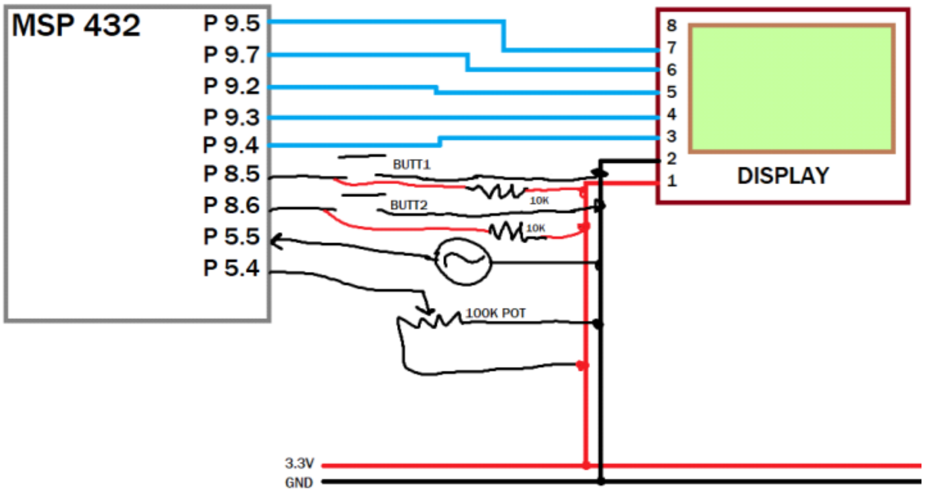
The user interface is based on that of the HP 48-50 series of graphing calculators, some of the finest machines ever made.

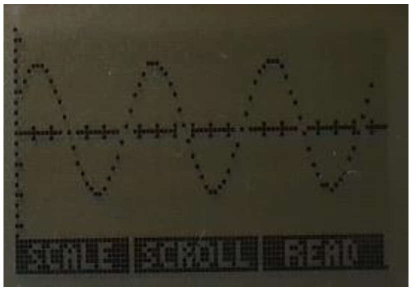
My code allows you to easily adjust the size of the array to suit the needs of users while managing memory constraints. I ran into memory problems during the development process, originally the array was 10 LCD lengths long, it is now 3 by default.

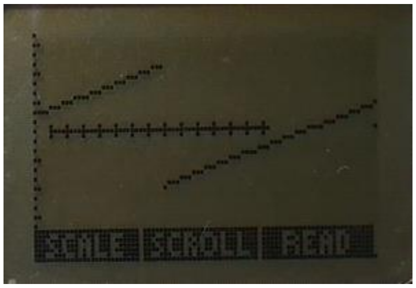
Conclusions:

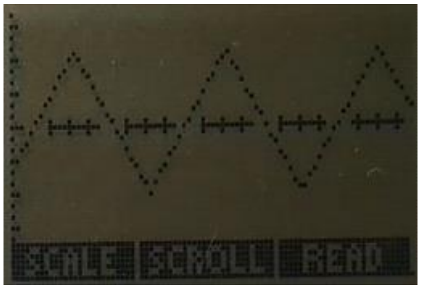
This lab took a long time but I am extremely satisfied with my final product. The scope works great and was overall a fun challenge. Thank you for a fantastic quarter, and see you next quarter Dr. Widder!

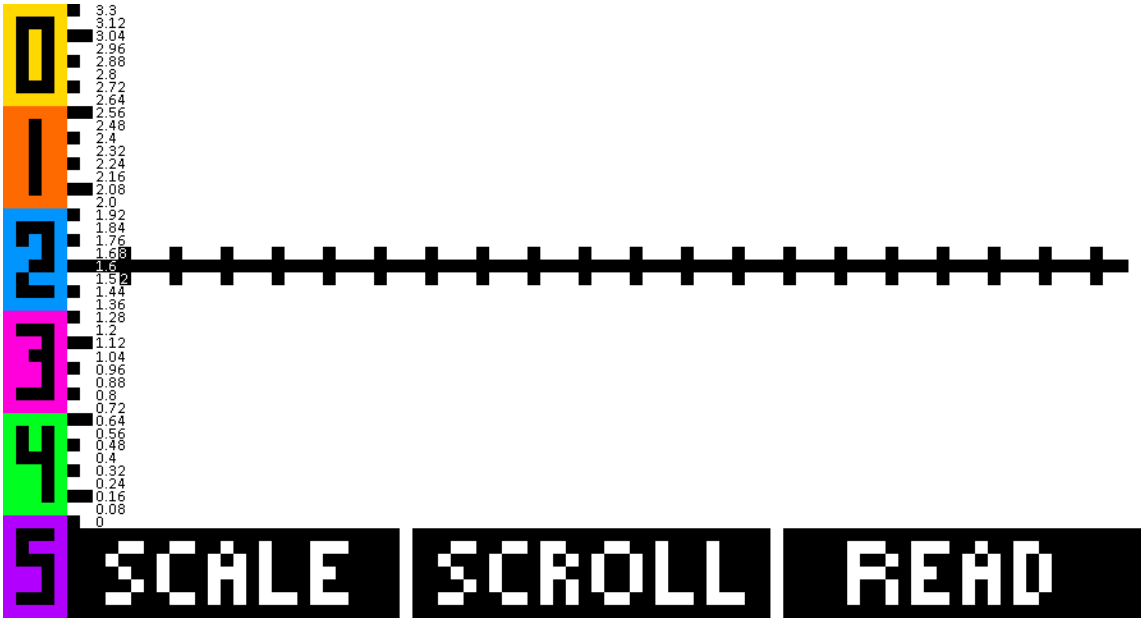
EE 2920
Final Proj...



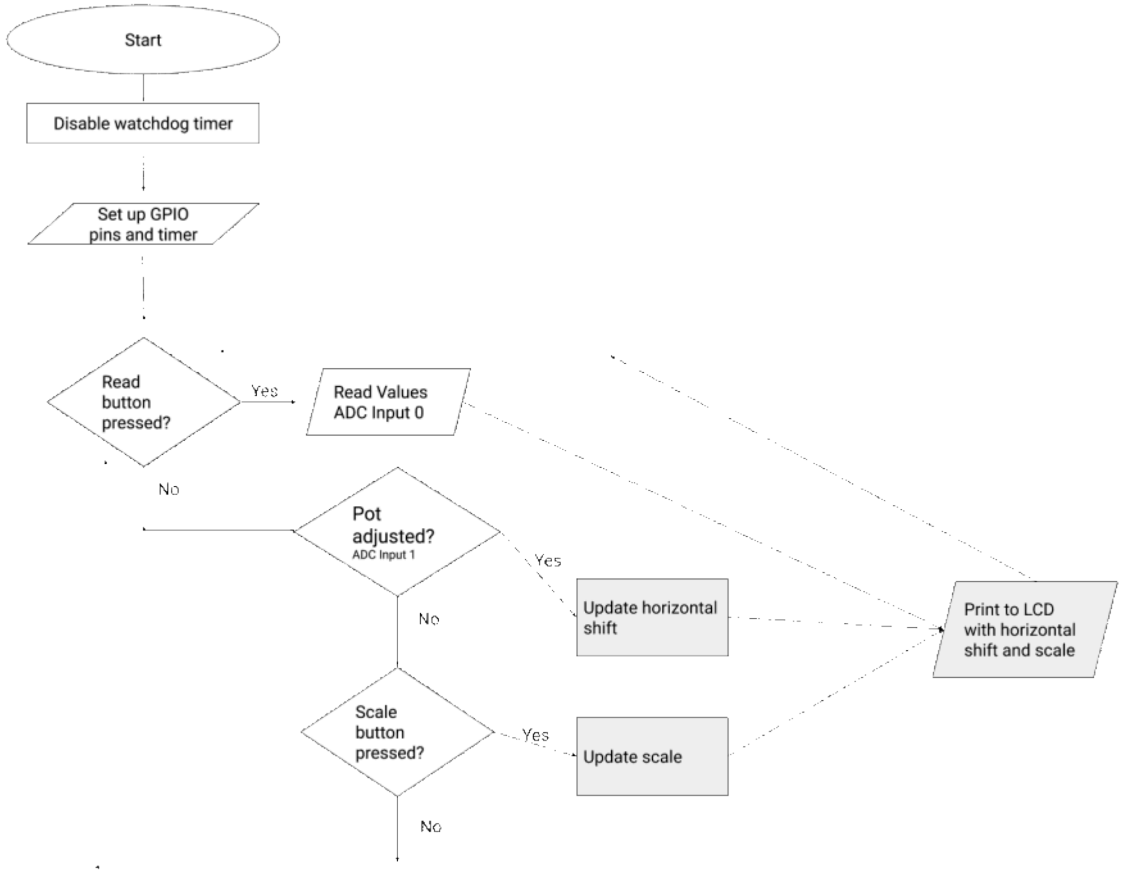








Display format and voltage scale diagram I used when developing the code.



MAIN.C

```
#include "msp.h"

#include "msoe_lib_all.h"

#include <stdio.h>

#include <stdint.h>

#include <math.h>

#include "bitmap.h"

#include "ee1910delay.h"

#include "analog.h"

/******************************************************************************
 * PROJECT: EE 2920 Final Project - Oscilloscope
 * ENTITY: MAIN.C
 * AUTHOR: John Bilkey
 * DATE: November 3, 2020
 * PROVIDES: Captures analog inputs, displays a plot on the LCD
 * REQUIRED HARDWARE:
 *  DISPLAY
 *   PIN 8 (backlight) - NOT CONNECTED
 *   PIN 7 (SCLK)      - P9.5
 *   PIN 6 (MISO)      - P9.7
 *   PIN 5 (D/C)       - P9.2
 *   PIN 4 (resetBAR)  - P9.3
 *   PIN 3 (STE)       - P9.4
 *   PIN 2 (GND)       - GND
 *   PIN 1 (VCC)       - 3.3V
 *
 *  BUTTONS
 *   Button 1: P 8.5 with a 10k pull-up resistor (READ)
 *   Button 2: P 8.6 with a 10k Pull-up resistor (SCALE)
 *
 *  ANALOG
 *   PIN 5.5  - Analog Input 0 V+
 *   PIN 5.4  - Analog Input 1 Potentiometer
 ******************************************************************************/

char printinterval = 1; // increment this many times when printing values in the array

int lastRead; // last read from Analog 1

#define timeBetweenReads 500 // microseconds
#define reads 253 // size of array for storing voltage measurements

// global variables for ADC output storage
volatile float results[reads];

void read(void);
void print(int hoffset, int interval);
void useTimer(void);
void timeScale(void);
int encodedButton(void);

void main(void) {

  // set unused pins to resistor enabled to avoid floating inputs
  P1 -> REN |= 0xFF;
  P2 -> REN |= 0xFF;
  P3 -> REN |= 0xFF;
  P4 -> REN |= 0xFF;
  P5 -> REN |= 0xFF;
  P6 -> REN |= 0xFF;
  P7 -> REN |= 0xFF;
  P8 -> REN |= 0xFF;
  P9 -> REN |= 0xFF;
  P10 -> REN |= 0xFF;

  P8 -> OUT |= (BIT5 | BIT6); // Set out register to 1 (input) for buttons on P 8.5

  Stop_watchdog();

  LCD_Config(); // set up ports for LCD and other stuff.

  A2Dsetup(); // prepare A2D (for reading input)

  //bilkeyelectric, grid, reading

  LCD_print_bmpArray(bilkeyelectric);

  LCD_contrast(1); // make LCD easier to read

  delay(1500); // keep that splash screen on there for a bit

  LCD_clear();

  while (1) {
    LCD_print_bmpArray(reading); // tells the user values are being recorded
    A2Dsetup();

    read();
    delay(500);

    A2Dsetup(); // sets up a2d for pot
    A1select();
    delay(500);
    print((A1read() / (4096 / (reads - 84))), printinterval); // initial display print
    while (encodedButton() != 1) { // wait until READ button is pressed
      if (encodedButton() == 2) { // if SCALE button is pressed
        timeScale();
        print((A1read() / (4096 / (reads - 84))), printinterval); // redraw LCD with input from pot for position
      }
      int lastRead = A1read();
      if ((A1read() < lastRead - 5) || (A1read() > lastRead + 5)) { // redraw display if potentiometer changes
        lastRead = A1read();
        print((A1read() / (4096 / (reads - 84))), printinterval);
      }
      delay(100);

    }
  }
}

void read(void) {

  A0select(); // switch to voltage reading pin
  delay(100);
  char index;
  for (index = 0; index < reads; index++) {
    results[index] = (A0read() * (3.3 / 4096));
    delayMicroseconds(timeBetweenReads);
  }
}

void print(int hoffset, int interval) { // displays values in array

  LCD_print_bmpArray(grid); // add grid for background

  char counter = 1; //position in array
  for (counter = 1; counter < 85; counter++) {
    if (((counter * interval) + hoffset) > reads) { // prevent reading out of array
      break;
    }
    LCD_col_exact(counter);
    LCD_row((int)(5-(results[(counter*interval)+hoffset]/0.64))); // row 0-5

    int intpart = (int)(results[(counter*interval)+hoffset]/0.64);
    float decpart = (results[(counter*interval)+hoffset]/0.64) - intpart;
    int subcolumn =  ((int)(decpart *8));
    LCD_print_column(1 << 7-subcolumn);
  }
}

void timeScale(void) { // changes time scale variable and displays on LCD
  LCD_goto_xy(5, 2);
  switch (printinterval) {

  case 1:
    printinterval = 2;
    LCD_print_str("2");
    break;
  case 2:
    printinterval = 3;
    LCD_print_str("3");
    break;
  default:
    printinterval = 1;
    LCD_print_str("1");
    break;
  }
  delay(1000);
}

int encodedButton(void) {
  int temp = 0; // no button pressed returns 0

  if (((P8 -> IN & BIT5) == 0) || ((P8 -> IN & BIT6) == 0)) {
    if (((P8 -> IN & BIT5) == 0) && ((P8 -> IN & BIT6) == 0) || ((P8 -> IN & BIT6) == 0) && ((P8 -> IN & BIT5) == 0)) {
      //printf("/nBOTH");
      temp = 3; // both buttons pressed returns 3
    } else
    if ((P8 -> IN & BIT5) == 0) {
      //printf("/n1");
      temp = 1; // only button 1 pressed returns 1
    } else
    if ((P8 -> IN & BIT6) == 0) {
      //printf("/m2");
      temp = 2; // only button 2 pressed returns 2
    }
  }
  return temp;
}
```

ANALOG.H

```
#ifndef ANALOG_H_
#define ANALOG_H_

#include "msp.h"

#include <stdint.h>
```

```
/******************************************************************************
 * PROJECT: EE 2920 Final Project - Oscilloscope
 * ENTITY: ANALOG.H
 * AUTHOR: John Bilkey
 * DATE: November 15, 2020
 * PROVIDES: Returns analog to digital converter value, 0 to 4096
 *
 * REQUIRED HARDWARE:
 *   PIN 5.5  - Analog Input 0 Potentiometer
 *   PIN 5.4  - Analog Input 1 V+
 *
 *  Includes modified code from Dr. Widder for EE 2920,
 *  Dr. Ross for EE 1910, Wei Zhao at Texas Instruments Inc
 ******************************************************************************/

void A2Dsetup(void) {
  // ADC Setup
  // You must enable the Analog 0 pin...

  ADC14 -> CTL0 = 0x04000210; // S/H timer, 16clk S/H, ADC ON
  ADC14 -> CTL1 = 0x00000020; // 12-bit conversion
}

void A0select(void) {
  // Function to setup Analog input A0
  // for use in A/D conversion
  // Setup ADC Input 0
  // Pin 30 --> P5.5
  P5 -> SEL0 |= BIT5; // Select alternate mode 11
  P5 -> SEL1 |= BIT5;

  P5 -> DIR &= ~BIT5; // input
  P5 -> REN &= ~BIT5; // No pull u/d

  P5 -> SEL0 &= ~BIT4; // Disable other pin
  P5 -> SEL1 &= ~BIT4;
  P5 -> REN |= BIT4;
  P5 -> DIR |= BIT4;

  ADC14 -> MCTL[0] = 0x00000000;
}

void A1select(void) {
  // Function to setup Analog input A1
  // for use in A/D conversion
  // Setup ADC Input 1
  // P5.4

  P5 -> SEL0 |= BIT4; // Select alternate mode 11
  P5 -> SEL1 |= BIT4;

  P5 -> DIR &= ~BIT4; // input
  P5 -> REN &= ~BIT4; // No pull u/d

  P5 -> SEL0 &= ~BIT5; // Disable other pin
  P5 -> SEL1 &= ~BIT5;
  P5 -> DIR |= BIT5;
  P5 -> REN |= BIT5;

  ADC14 -> MCTL[0] |= ADC14_MCTLN_INCH_1;

}

int A0read(void) {

  // Function to perform a single A/D conversion on Analog input 0, P 5.5

  // enable ADC and start conv
  ADC14 -> CTL0 |= ADC14_CTL0_ENC | ADC14_CTL0_SC;

  // Wait for conversion to complete
  // Conversion is complete when ADC0 flag is set
  while (!ADC14 -> IFGR0) {}

  // returning value
  return ADC14 -> MEM[0];
}

int A1read(void) {
  // Function to perform a single
  // A/D conversion on Analog input 1
  // Start sampling/conversion
  ADC14 -> CTL0 |= ADC14_CTL0_ENC | ADC14_CTL0_SC;

  // Wait for conversion to complete
  // Conversion is complete when ADC0 flag is set
  while (!ADC14 -> IFGR0) {
    ;
  }
  // returning a full int instead of a uint16_t for simplicity
  return ADC14 -> MEM[0];
}

#endif

/*
 * Experimental code for 3 simultaneous reads and single setup
 * Would be used with full bridge rectifier and op-amp to measure AC voltage with no DC offset
 *
//   PIN 5.5  - Analog Input 0 V+
//   PIN 5.4  - Analog Input 1 V-
//   PIN 5.3  - Analog Input 2 Potentiometer

void analogSetup(void) {

    //Configure GPIO

    P5->DIR &= ~(BIT3|BIT4|BIT5); // make input
    P5->REN &= ~(BIT3|BIT4|BIT5); // No pull up/down resistor

    P5 -> SEL1 |= (BIT3|BIT4|BIT5);  //Enable A/D channel A0-A2
    P5 -> SEL0 |= (BIT3|BIT4|BIT5);

    P5 -> REN |= BIT2; // connect analog 3 to GND with pulldown

    __enable_interrupt();
    NVIC_ISER0 = 1 << ((INT_ADC14 - 16) & 31);
    //Enable ADC interrupt in NVIC module, Turn on ADC14, extend sampling time

    // SIMULTTANEOUS MULTI-SAMPLE MODE
    ADC14 -> CTL0 |= (ADC14ON | ADC14MSC | ADC14SHT0__192 | ADC14SHP | ADC14CONSEQ_3);

    //to avoid overflow of results
    ADC14MCTL0 = ADC14INCH_0; //ref+=AVcc, channel = A0
    ADC14MCTL1 = ADC14INCH_1; //ref+=AVcc, channel = A1
    ADC14MCTL2 = ADC14INCH_2+ADC14EOS; //ref+=AVcc, channel = A2, end seq.

    ADC14IER0 = ADC14IE3; //Enable ADC14IFG.3

    SCB_SCR &= ~SCB_SCR_SLEEPONEXIT; //Wake up on exit from ISR

    while(1) {
     ADC14CTL0 |= ADC14ENC | ADC14SC; //Start conv-software trigger
     }
   }

}

void analogRead(int index){
    // Function to perform a single
    // A/D conversion on Analog input 0

    // Start sampling/conversion
    ADC14->CTL0 |= 0x00000003; // enable ADC, start conversion
    // Wait for conversion to complete
    // Conversion is complete when ADC0 flag is set
    while (!ADC14->IFGR0){}

    A0results[index] = ADC14->MEM[0]; //Move A0 results, IFG is cleared
    A1results[index] = ADC14->MEM[1]; //Move A1 results, IFG is cleared
    A2results = ADC14->MEM[2]; //Store A2, IFG is cleared

    }
}
*/
```

MSOE_LIB_LCD.C

```
 // MODIFIED BY BILKEY, NOV 13 2020
void LCD_goto_exact(uint8_t x, uint8_t y){
    LCD_Command_WR(0x80 | x);     // 1 x6 x5 x4 x3 x2 x1 x0 : sets X location
    LCD_Command_WR(0x40 | y);           // 0100 0 y2 y1 y0 : sets Y location
}


// MODIFIED BILKEY, NOV 13 2020
void LCD_print_column(char poggers){
        LCD_Data_WR(poggers);
}

// otherwise it is the same
```

MSOE_LIB_LCD.H

```
// CUSTOM COMMANDS BY JOHN BILKEY
void LCD_goto_exact(uint8_t x, uint8_t y);
void LCD_col_exact(uint8_t col);
void LCD_print_column(char poggers);


// otherwise it is the same
```

EE1910DELAY.H

```
/*
 * ee1910delay.h
 *
 *  Created on: Dec 5, 2018
 *      Author: ross
 */

#ifndef EE1910DELAY_H_
#define EE1910DELAY_H_

#include "msp.h"

volatile uint32_t ticks=0;

void delayMicroseconds(uint32_t us) {

      // Set up Timer32 for down count with default 3MHz clock
       TIMER32_1->CONTROL = TIMER32_CONTROL_SIZE |
               TIMER32_CONTROL_MODE;

       // Load Timer32 counter with period determined by us
       TIMER32_1->LOAD= us*3;

       // Enable the Timer32 interrupt in NVIC
       __enable_irq();
       NVIC->ISER[0] = 1 << ((T32_INT1_IRQn) & 31);

       // Start Timer32 with interrupt enabled
       TIMER32_1->CONTROL |= TIMER32_CONTROL_ENABLE |
                   TIMER32_CONTROL_IE;

       // Wait for desired time
       while(!ticks){
       }
       ticks = 0;

       // Turn off timer
       TIMER32_1->CONTROL &= ~(TIMER32_CONTROL_ENABLE |
                          TIMER32_CONTROL_IE);
}


void delay(uint32_t ms) {

      // Set up Timer32 for down count with default 3MHz clock
       TIMER32_1->CONTROL = TIMER32_CONTROL_SIZE |
               TIMER32_CONTROL_MODE;

       // Load Timer32 counter with period 3000
       TIMER32_1->LOAD= 3000;

       // Enable the Timer32 interrupt in NVIC
       __enable_irq();
       NVIC->ISER[0] = 1 << ((T32_INT1_IRQn) & 31);

       // Start Timer32 with interrupt enabled
       TIMER32_1->CONTROL |= TIMER32_CONTROL_ENABLE |
                   TIMER32_CONTROL_IE;

       // Wait for desired time
       while(ticks!=ms){
       }
       ticks = 0;

       // Turn off timer
       TIMER32_1->CONTROL &= ~(TIMER32_CONTROL_ENABLE |
                          TIMER32_CONTROL_IE);
}
```

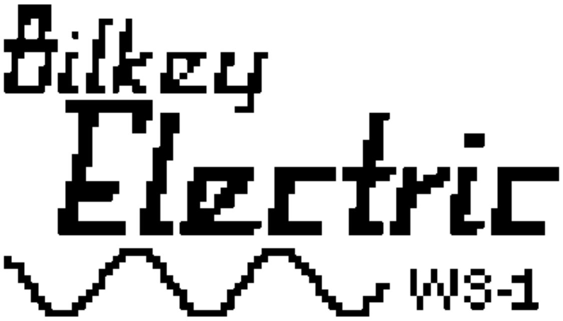
```
void T32_INT1_IRQHandler(void)
{
    TIMER32_1->INTCLR |= BIT0;                // Clear Timer32 interrupt flag
    ticks++;                                  // Record tick
}
#endif
```

BITMAP.H

```
#ifndef BITMAP_H__
#define BITMAP_H__

static
const char bilkeyelectric[] = {

  0b11000000,
  0b11000000,
  0b11111110,
  0b11111110,
  0b11001110,
  0b11111110,
  0b11111110,
  0b11011110,
  0b11000000,
  0b00000000,
  0b00100000,
  0b00100000,
  0b00000000,
  0b11000000,
  0b11111000,
  0b11111000,
  0b00001000,
  0b11100000,
  0b11111000,
  0b11111000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
```

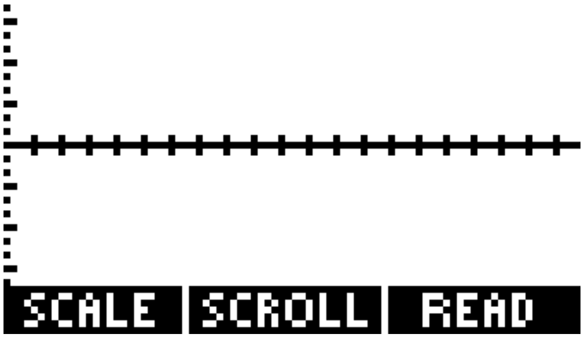


```
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00111000,
0b00111111,
0b00111111,
0b00100001,
0b00100000,
0b00111100,
0b00111111,
0b00011111,
0b00111000,
0b10111111,
0b10111111,
0b10100001,
0b10111000,
0b10111111,
0b10111111,
0b10100001,
0b10111100,
0b10111111,
0b10011111,
0b10111110,
0b10110011,
0b10100011,
0b10000001,
0b00111000,
0b00111111,
0b00111111,
0b00111001,
0b00101111,
0b00100111,
0b00100011,
0b00100001,
0b00111000,
0b00111111,
0b00111111,
0b00100011,
0b00100000,
0b11111000,
0b11111111,
0b00111111,
0b00000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b10000001,
0b11111111,
0b11111111,
0b11111111,
0b00000111,
0b00000001,
0b00000001,
0b00000001,
0b00000001,
0b00001111,
0b00001111,
0b00001111,
0b11000011,
0b11111110,
0b11111110,
0b11111110,
0b00001110,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b00000001,
0b00000001,
0b00000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b11100000,
0b11111110,
0b11111110,
0b11111110,
0b00000010,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00110000,
0b00110000,
0b00110000,
0b00110000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b10000000,
0b11111000,
0b11111111,
0b11111111,
0b00111111,
0b00111001,
0b00111000,
0b00111000,
0b00111000,
0b00011000,
0b00000000,
0b00000000,
0b10000000,
0b11111000,
0b11111111,
0b11111111,
0b00011111,
0b00000000,
0b00000000,
0b11111000,
0b11111110,
0b11111110,
0b10111110,
0b11100110,
0b11110110,
0b01111110,
0b01111110,
0b00011110,
0b00011110,
0b00000110,
0b00000000,
0b10000000,
0b11111110,
0b11111110,
0b11111110,
0b00011110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b10000110,
0b11111110,
0b11111111,
0b11111111,
0b00001111,
0b00000110,
0b00000110,
0b11110010,
0b11111110,
0b11111110,
0b01111110,
0b00011100,
0b00011100,
0b00001110,
0b00001110,
0b11100110,
0b11111110,
0b11111110,
0b11111110,
0b00000110,
0b00000000,
0b10000000,
0b11111110,
0b11111110,
0b11111110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000000,
0b11000000,
0b11000000,
0b10000000,
0b10000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000111,
0b00000111,
0b00000111,
0b00000111,
0b00000110,
0b00000110,
0b00000110,
0b10000110,
0b11000110,
0b11100110,
0b01100110,
0b00100000,
0b00100111,
0b01100111,
0b11100111,
0b11000111,
0b10000110,
0b00000110,
0b00000110,
0b00000111,
0b00000111,
0b00000111,
0b00000111,
0b00000111,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b10000110,
0b11000110,
0b11100010,
0b01100111,
0b00100111,
0b00100111,
0b01100111,
0b11100110,
0b11000110,
0b10000110,
0b00000110,
0b00000110,
0b00000000,
0b00000000,
0b00000000,
0b00000111,
0b00000111,
0b00000111,
0b00000111,
0b00000110,
0b00000110,
0b00000110,
0b00000111,
0b00000111,
0b00000111,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000111,
0b00000111,
0b00000111,
0b00000110,
0b00000110,
0b00000010,
0b00000111,
0b00000111,
0b00000111,
0b00000111,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
  0b00000011,
  0b00000111,
  0b00011110,
  0b00111100,
  0b01110000,
  0b01100000,
  0b01000000,
  0b01000000,
  0b01100000,
  0b01110000,
  0b00111000,
  0b00011100,
  0b00001111,
  0b00000111,
  0b00000001,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000011,
  0b00000111,
  0b00001110,
  0b00111100,
  0b01111000,
  0b01100000,
  0b01000000,
  0b01000000,
  0b01000000,
  0b01100000,
  0b01111000,
  0b00111100,
  0b00001110,
  0b00000111,
  0b00000011,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000001,
  0b00000011,
  0b00001111,
  0b00011110,
  0b00111000,
  0b01110000,
  0b01100000,
  0b01000000,
  0b01000000,
  0b01110000,
  0b01110000,
  0b00011100,
  0b00011100,
  0b00000100,
  0b00000000,
  0b00000000,
  0b00001111,
  0b00111111,
  0b00111100,
  0b00001111,
  0b00001111,
  0b00111100,
  0b00111111,
  0b00001111,
  0b00010010,
  0b00110111,
  0b00100101,
  0b00111111,
  0b00011010,
  0b00001000,
  0b00001000,
  0b00101010,
  0b00100010,
  0b00111111,
  0b00111111,
  0b00100000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000

};

static
const char grid[] = {
```

```
0b01010101,
0b00000100,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b01010101,
0b01000001,
0b00000000,
0b00000000,
0b00000000,
```



```
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b01010101,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b00111000,
0b00010000,
0b00010000,
0b00010000,
0b01010101,
0b00000100,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b01010101,
0b01000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b11111111,
0b11111110,
0b11111110,
0b10110110,
0b10101010,
0b11011010,
0b11111110,
0b11000110,
0b10111010,
0b10111010,
0b11111110,
0b10000110,
0b11101010,
0b10000110,
0b11111110,
0b10000010,
0b10111110,
0b10111110,
0b11111110,
0b10000010,
0b10101010,
0b10111010,
0b11111110,
0b11111110,
0b11111110,
    0b11111110,
    0b00000000,
    0b11111110,
    0b11111110,
    0b10110110,
    0b10101010,
    0b11011010,
    0b11111110,
    0b11000110,
    0b10111010,
    0b10111010,
    0b11111110,
    0b10000010,
    0b11101010,
    0b10010110,
    0b11111110,
    0b11000110,
    0b10111010,
    0b10111010,
    0b11000110,
    0b11111110,
    0b10000010,
    0b10111110,
    0b10111110,
    0b11111110,
    0b10000010,
    0b10111110,
    0b10111110,
    0b11111110,
    0b11111110,
    0b00000000,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b10000010,
    0b11101010,
    0b11101010,
    0b10010110,
    0b11111110,
    0b10000010,
    0b10101010,
    0b10111010,
    0b11111110,
    0b10000110,
    0b11101010,
    0b10000110,
    0b11111110,
    0b10000010,
    0b10111010,
    0b11000110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110,
    0b11111110

};

static
const char reading[] = {

  0b00000000,
  0b00000000,
  0b00000000,
  0b11110000,
  0b11110000,
  0b00110000,
  0b00110000,
  0b00110000,
  0b00110000,
  0b11110000,
  0b11100000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b10000000,
  0b10000000,
  0b10000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
```

```
0b00000000,
0b10000000,
0b10000000,
0b10000000,
0b10000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b10000000,
0b10000000,
0b10000000,
0b00000000,
0b11110000,
0b11110000,
0b00000000,
0b00000000,
0b10110000,
0b10110000,
0b00000000,
0b00000000,
0b10000000,
0b10000000,
0b00000000,
0b10000000,
0b10000000,
0b10000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b10000000,
0b10000000,
0b10000000,
0b11000000,
0b01000000,
0b01000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b01111111,
0b01111111,
0b00000110,
0b00000110,
0b00000110,
0b00011110,
0b01111011,
0b01100001,
0b01000000,
0b00000000,
0b00000000,
0b00011110,
0b00111111,
0b01100101,
0b01100101,
0b01100101,
0b00110111,
0b00010110,
0b00000000,
0b00000000,
0b00111011,
0b01111011,
0b01101101,
0b01100101,
0b00100101,
0b01111111,
0b01111111,
0b00000000,
0b00000000,
0b00111111,
0b01111111,
0b01100001,
0b01100001,
0b00100001,
0b01111111,
0b01111111,
0b00000000,
0b00000000,
0b01111111,
0b01111111,
0b00000000,
0b00000000,
0b01111111,
0b01111111,
0b00000001,
0b00000001,
0b00000001,
0b01111111,
0b01111111,
0b00000000,
0b10000000,
0b10110111,
0b01111111,
0b01101000,
0b01101000,
0b01101111,
0b11100111,
0b11000000,
0b00000000,
0b00000000,
0b01100000,
0b01100000,
0b00000000,
0b00000000,
0b01100000,
0b01100000,
0b00000000,
0b00000000,
0b01100000,
0b01100000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b00000011,
0b00000010,
0b00000010,
0b00000010,
0b00000010,
0b00000011,
0b00000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00111000,
0b01111000,
0b01111000,
0b11111100,
0b11111100,
0b11111100,
0b00111100,
0b00011110,
0b00011110,
0b00001110,
0b00001110,
0b00001110,
0b00001110,
0b00001110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00001110,
0b00111110,
0b01111100,
0b11111100,
0b11111100,
0b11111000,
0b11111000,
0b11111000,
0b01110000,
0b00110000,
0b00110000,
0b00110000,
0b00110000,
0b11110000,
0b11110000,
0b11110000,
0b11111000,
0b01111000,
0b00011100,
0b00011100,
0b00001100,
0b00001100,
0b00001100,
0b00001100,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00000110,
0b00001110,
0b00001100,
0b00001100,
0b00001100,
0b00001100,
0b00011100,
0b00011100,
0b00111100,
0b00111100,
0b11111000,
0b11111000,
0b11111000,
0b11111000,
0b11110000,
0b11110000,
0b11110000,
0b01110000,
0b00110000,
0b00110000,
0b00110000,
0b00110000,
0b00110000,
0b00110000,
0b11110000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b00000001,
0b00111111,
0b11000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b10000000,
0b11000011,
0b01111111,
0b00111111,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b01111111,
0b11000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b11000000,
0b01110011,
0b00111111,
0b00000011,
0b00000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00011111,
0b00011000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b00000010,
0b00000100,
0b00000100,
0b00000100,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001100,
0b00000110,
0b00000011,
0b00000001,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000001,
0b00000011,
0b00000110,
0b00000100,
  0b00001100,
  0b00011000,
  0b00010000,
  0b00010000,
  0b00110000,
  0b00100000,
  0b00100000,
  0b00100000,
  0b00100000,
  0b00100000,
  0b00100000,
  0b00110000,
  0b00110000,
  0b00011000,
  0b00001000,
  0b00001100,
  0b00000100,
  0b00000111,
  0b00000001,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000

};

#endif BITMAP_H_
```