Design of a Graphical LCD Driver and Educational LCD Primer

A Design Project Report
Presented to the Engineering Division of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering (Electrical)

by
Lucas L. Delaney
Project Advisor: Dr. Bruce R. Land
Degree Date: May 2004

Abstract

Master of Electrical Engineering Program

Cornell University

Design Project Report

Project Title: Design of a Graphical LCD Driver and Educational Primer

Author: Lucas L. Delaney

Abstract: Graphical Liquid Crystal Displays add versatility to any project. I designed a graphical LCD driver for use with the Crystalfontz CFAG12864B 128 x 64 pixel graphical LCD for student use. In addition, I wrote a basic primer on the operation of graphical LCDs so that future student developers will be better able to program the CFAG12864B or any other Samsung KS0108 based graphical LCD.

Report Approved by
Project Advisor: _____Date: _____

Executive Summary

Whenever a student can learn from other students, the education of everyone involved is enriched. The Cornell Electrical and Computer Engineering course ECE 476: Microcontroller Design is the kind of course where the more knowledge is exchanged between students, the more everyone learns. This project is designed to help students in that class or any other students working on a project that requires a graphical user interface have a better understanding of how to program their interface. It is comprised of a driver file that makes a popular graphical LCD, Crystalfontz CFAG12864B product line, easier to use and gives an explanation of how Samsung KS0108 based Graphical LCDs work. Many graphical LCDs on the market place use the KS0108 controller or a controller that uses similar principles and there is not that much step-by-step documentation on how to get started programming with one of these devices. This project is designed as a tool to help me share my experience with this project with others who may be working on similar projects either now or in the future.

Table of contents

1. Introduction

The course ECE 476: Microcontroller Design requires many tools that allow its students to fully experience the possibilities of designing projects using Microcontrollers. In order for instructors to design laboratory experiments and demonstrations it is essential that he have the tools necessary to make them as easy to put together as possible. The goal of this project is select a low-cost graphical LCD and design a driver that would allow such experiments and demonstrations to be designed around it.

In most of the experiments used for ECE 476, a 16x2 Crystalfontz Alphanumeric LCD is used as the major user output and represents the user interface. Alphanumeric LCDs display characters in pre-designated blocks and the LCD screen and this limits their use to simple number and character displays and crude images drawn from numbers or characters (a bouncing ball using the character 'o' or other such graphical techniques using text). While this is suitable for many applications, there are some which would benefit greatly from an easy to use graphical LCD. Most graphical LCDs are not supported by standard C libraries as are simple alphanumeric displays so it becomes much more time consuming to use them in projects. This can be especially prohibitive during regular laboratory experiments because they are often designed to prove a specific instructive idea, and generating a driver for a graphical LCD cannot be done during the allotted time. This paper and project outline the design of a graphical LCD driver for the Crystalfontz CFAG12864B series (128 x 64 pixel) graphical display which can be easily modified to drive any Samsung KS0108 based graphical LCD.

2. Design Problem and Requirements

2.1 Design Problem

My Design Problem is to find the available documentation on KS0108 based graphical displays and write a driver for the Atmel ATmega32 MCU based on this information. In addition, I plan on thoroughly explaining how the LCD works so that if someone wants to modify, update, or improve on the driver, it will be a simple task. I have found that there is very little information on writing graphical LCD drivers and think that this paper will serve as a useful tool and possible primer for those setting out on completing a similar task.

2.2 Requirements

The requirements of this project are as follows:

- Select and purchase a readily available, low-cost 128 x 64 graphical LCD which could be obtained by course instructors.

- Thoroughly research and understand the operation of the graphical LCD and write a primer so that others can benefit from my research.

- Write a driver that has the ability to place images on the screen.

- Write an image library that replicates the functions of an alphanumeric LCD by storing images of each character which can be placed on the display.

3. Range of Solutions

The range of solutions in a project of this nature is relatively limited. The project is relatively straightforward in what it needs to accomplish. There are two main parts of the project: Writing clear documentation on how the LCD works, and writing a driver that can be used as a tool for ECE 476. There are many KS0108 based graphical LCDs of varying sizes. The KS0108 LCD driver is capable of driving 64 x 64 pixels and many can be put together and multiplexed to create displays of varying sizes. I chose the Crystalfontz CFAG12864B because it is low-cost and many of the other LCD displays used in ECE 476 are Crystalfontz displays implying that there is a prior relationship between the company and the class and perhaps the LCDs may be able to be purchased in volume or donated to the class in the future.

4. Design and Implementation

4.1 The CFAG12864B

The CFAG12864B is a 128 x 64 pixel graphical LCD with backlight. It is driven by 2 64 x 64 pixel Samsung KS0108 drivers. Figure 1 shows an image of what the LCD looks like with a sample output of a tractor. The actual view area of the LCD is 60 mm x 32.6 mm.

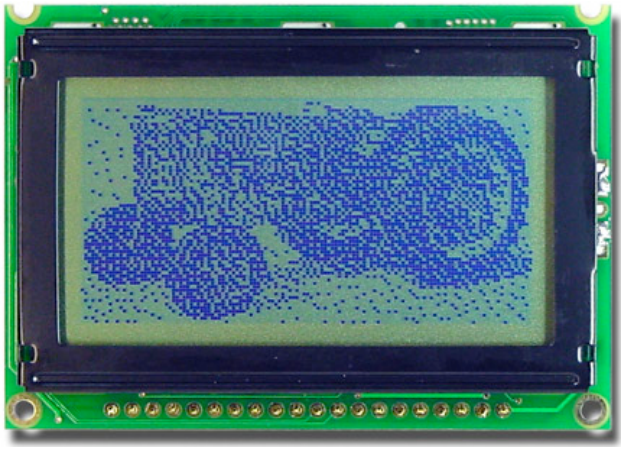


Figure 1: The CFAG12864B graphical Liquid Crystal Display (Pin 1 is leftmost)

4.2 Pin Assignments

4.2.1 Setting up the LCD and turning it on

The Appendix contains a detail pin-out of the LCD but the important pins for this section are as follows:

Pin #	Symbol	Function/Typical Value
1	Vdd	Supply voltage – connect to 5V
2	Vss(GND)	Ground – connect to ground
3	Vo	Operating voltage – Connect through pot as is described below
18	Vee	Negative Voltage output – outputs -5V – connect to pot
19	A	Power supply for backlight (+)
20	K	Power supply for backlight (-)

In order for the LCD to power up, Pin 1 must be connected to + 5V, Pin 2 must be connected to GND and Pin 3 & Pin 18 must be connected as illustrated in Fig. 2. Pin 18

generates -5V as an output and it must be run through a trimpot (or voltage divider) and fed into Pin 3. This provides the voltage differential of Vdd-Vo which must be at least 7.5V. Adjusting this value adjusts the contrast but the Pins must be connected in this way in order for the image to be seen on the screen (if it isn't there is essentially no contrast and nothing will be displayed.

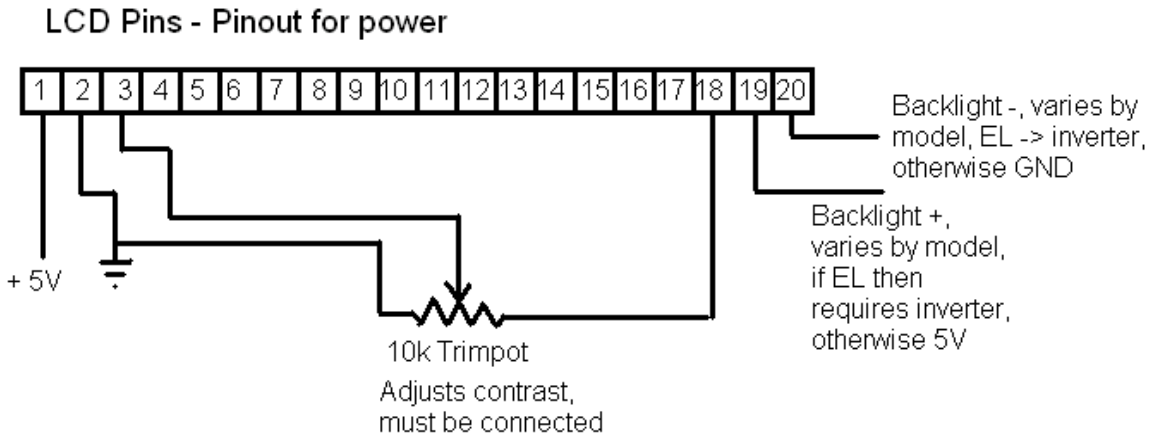


Figure 2: This figure clarifies how the LCD pins must be connected in order for the LCD to work

4.2.2 Inputs and Outputs

This display has 14 I/O ports and 6 power related inputs/outputs. This paper will only refer to the input options of the LCD because this driver writes to the LCD but LCD is capable of outputting the contents of its RAM to the MCU for debugging purposes. Fig. 2 shows a block diagram of the operation of the LCD. The Appendix also contains a more detailed version of how this display is set up using the ATmega32.

These 14 outputs each have a specific purpose which is outlined below. Almost all of the data sheets that I came across only gave a vague idea of what they need to be set to in order to perform the appropriate LCD functions. I have gathered many data sheets for KS0108 driven LCDs and this section attempts to explain what each pin does and how they each affect what is displayed on the screen. There are 8 data bits that provide a number of functions in the operation of the LCD. They are the main information carriers to the LCD. They are located on Pins 4 - 11 with Pin 4 assigned to Data Bit 0 and Pin 11 assigned to Data Bit 7. Table 2 gives a brief overview of these assignments.

Pin #	Symbol	Function/Typical Value	Port Assignment
4	DB0	Data bit 0	D.0
5	DB1	Data bit 1	D.1
6	DB2	Data bit 2	D.2
7	DB3	Data bit 3	D.3
8	DB4	Data bit 4	D.4
9	DB5	Data bit 5	D.5
10	DB6	Data bit 6	D.6
11	DB7	Data bit 7	D.7

Table 2: Data bits Pin assignments

The only real reason that I included the pin assignments is that throughout this section every pin assignment will be displayed and their port assignments in the driver will be clearly understandable. The driver assigns these 8 data bits to Port D in the ATmega32 MCU. This also noted in the Hardware schematic located in the Appendix. The functions of the data bits will become clearer in later sections of the paper.

4.2.3 Control Bits

The CFAG12864B has 6 control bits which are used to control the operation of the KS0108 hardware drivers and display data on the LCD. They are listed in table 3, along with their MCU port assignments.

Pin #	Symbol	Function/Typical Value	Port Assignment
12	CS1	Column Select 1 (or Chip Select 1) – Selects the left KS0108 driver which is also left half of the screen	C.0
13	CS2	Column Select 2 (or Chip Select 2) – Selects the right KS0108 driver which is also right half of the screen	C.1
14	RST	Reset – set high to reset the display, low otherwise	C.2
15	R/W	Read/Write – Selects which type of instruction is being performed on the LCD – set high to read from the LCD to the MCU, set low to write from the MCU to the LCD	C.3
16	D/I	Data/Instruction – tells the LCD whether or not data is being written to the screen or the MCU is using the data bits to perform an instruction – set high for data transfer and set low to designate and instruction is being performed	C.4

17	E	Enable – The enable is used to clock operations to the LCD – it ends up being the clock of the LCD and when it clocks, sequential instructions are performed	C.5

Table 4: Control bits Pin assignments

The most important (or most commonly used and confused) control bits are R/W, D/I, E. Enable must be set low and then high in order for an operation to be passed to the LCD. In addition there must be a delay of at least 1 ms in between operations to compensate for the necessary cycle time of the enable bit. This timing can be one of the most frustrating errors because if it is not delayed properly, a seemingly good instruction will not provide the desired result. R/W and D/I are used to determine the mode of operation that the LCD is in. Table 4 illustrates how these two control bits are used to control operations.

D/I	R./W	Mode
0	0	An Instruction is being written (such as define Y address)
0	1	MCU reads the status of the LCD – whether it is busy or ready for another command
1	0	Data write – data is written to the display ram at whatever X-Y coordinates have been set
1	1	Data read – the data in the display RAM is read to the MCU

Table 4: D/I and R/W modes of operation

4.3 Writing Data to the Screen

One of the most frustrating parts of graphical LCD documentation is that rarely do they make it easy to understand exactly how to write data to the screen. What bits control what, in what order do I have to execute different instructions, what bits do I need to set in order to do what I want? I found that these questions are rarely answered in a straightforward manner and I plan on using this section to explain how to control the LCD and how to write data to the screen. After all, this is the purpose of the project and the LCD and if you can't write anything to the screen, then what is the point?

4.3.1 Turning the Display (RAM) On

First things first, the display needs to be turned on this is done by sending the following instruction to the screen:

Instruction	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Display ON	0	0	0	0	1	1	1	1	1	1

Table 5: Display ON instruction

I saved a lot of programming effort by defining each bit globally so that the actually port values need not be assigned every time a bit needs to be toggled. For example, Data bit 7 (D7 in my driver) is given the assignment: const unsigned char D7 = 0b10000000; at the beginning of the program. This makes toggling the bits much easier and it also makes the program much easier to read and follow because it is easy to see what is going on in each instruction

```
PORTD = ~D7 & ~D6;                          //sets PORTD to 0b00111111;
PORTC = ~E & ~DI & ~RW & ~RS & ~CS2;         //sets PORTC enable low and also
                                                  //sets D/I and R/W low
delay_ms(2);                                //delay 2ms
PORTC = ~DI & ~RW & ~RS & ~CS2;         //toggle enable high
delay_ms(2);                                //delay 2 ms
PORTC = ~E & ~DI & ~RW & ~RS & ~CS2;         //toggle enable low
```

This set of code turns the display on. The reason that I drew this example out and included the code (instead of just referring to the Appendix) is so that it is clear how to go from the instruction outlined in Table 5 to performing it CodeVision C. In this driver, this instruction would actually be done using a function called writeinstruction(instruction) where the instruction would be 0b00111111. I hope this gives a good idea of how to implement an instruction. There are a few instructions that need to be performed in order to operate the LCD and they are all done in this manner: set instruction, toggle E, and make sure that it is delayed more than 1 ms between toggles. The reason that I found it alright to use software delays in the initialization is that the delay time can be relatively inaccurate and still perform properly. As long as enough time has passed so that the enable can rise or fall, the toggle instruction should work properly.

4.3.2 How the Input Bits Translate into What is on the Screen

In order to place any information on the screen, it is important to understand how the bits control what is on the screen. There are 8192 pixels on a 128 X 64 pixel screen and each is pixels is control by a series of instructions. Figure 3 and Figure 4 show how the screen is broken down into its X and Y axes. Fig 3. shows the left half of the screen (if CS1 is 1 and CS2 is 0). The Y address refers to which line the pixels should be written to and the X page sets the column to which they will be written to. Please see the Appendix for how to set the Yaddress and Xpage functions, but there are three basic steps that must be done in order to determine where the pixels will go. First, the Y address must be set. The Y address actually has a counter so it need only be set once and then every time there is a data write it will be incremented to the next line. This allows the driver to scan through the lines and display the proper data. The next step that must be completed is setting the X page. This determines which column of the screen will now be written to. The third step is issuing a data write command. Whatever data bits are high will be darkened on the Y address line in the X page as is illustrated in Fig. 4. The driver scans through the pages, using the internal Y address counter to its advantage and resetting the X page (and eventually the CS) as it scans through the lines. I am adding Figures 3 and 4 to this report because I think that they do a much better job of explaining what is going on pixel by pixel on the screen than do any of the data sheets that I have seen. One very important thing to note is that the X and Y axes are reversed from what they usually would be. I am not sure why this is, but I suspect it has something to do with how the two KS0108 controllers are connected. Every 128 x 64 graphical LCD that I found was arranged in this way, even if it was not controlled by the KS0108 so I suspect it is the convention.

To reiterate, to select which bit on the screen will be effected:

- Set Y address which chooses one of the 64 vertical rows in the half of the screen that is currently selected by CS1 or CS2.
- Set the X page which selects which of the 8 horizontal stripes the 8 data bits will be placed in.

- Write data to the 8 bits that were just selected using a function such as writedata in this driver.

The process of selecting where on the screen the bit is going to go is just a narrowing process. CS1 and CS2 narrow it down to half the screen. Y address narrows it down to a 1 bit wide stripe of height 64 in that half of the screen. X page picks an 8 bit chunk of this 64 bit stripe and writedata writes 8 bits to that chunk. This is the essence of putting bits on the screen.

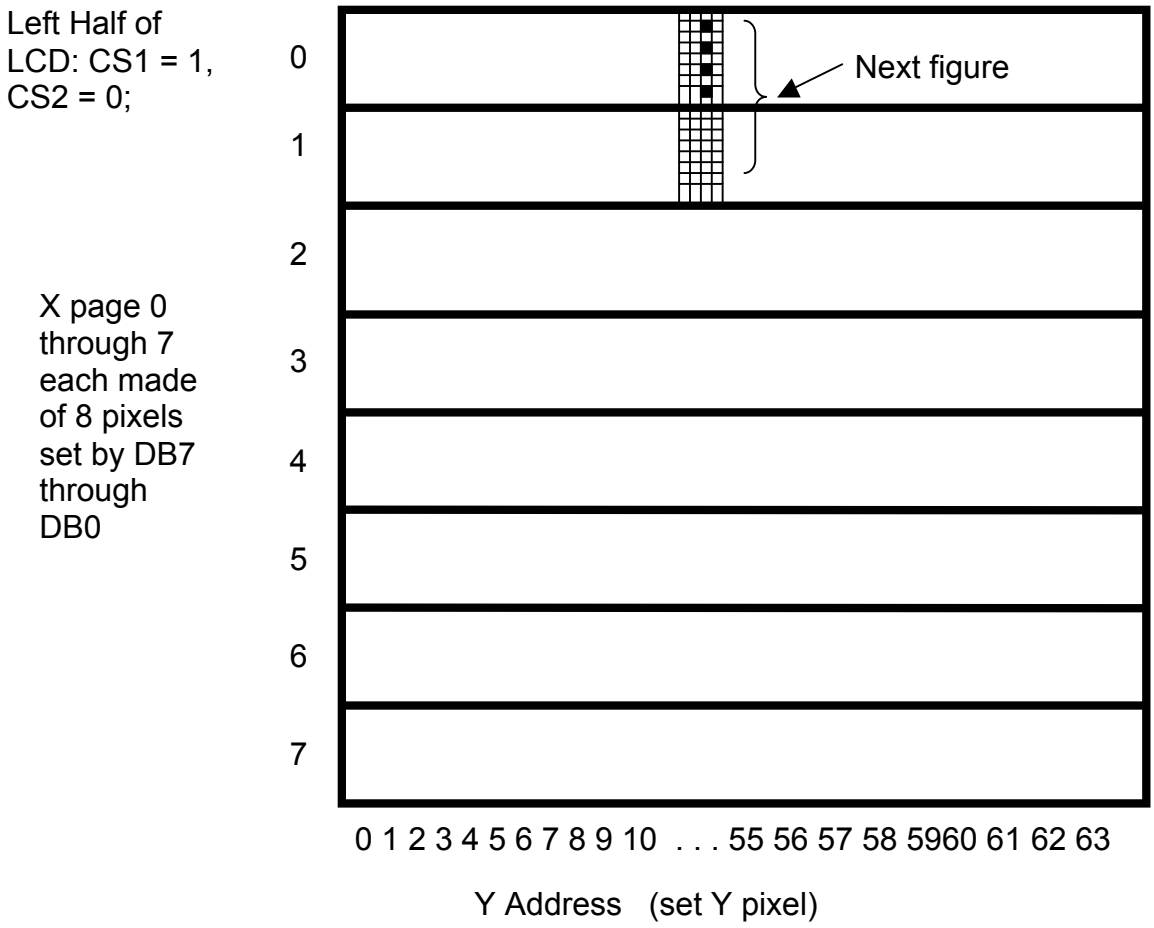


Figure 3: This figure illustrates how the left half (or right half if CS2 = 1 and CS1 = 0) works. The Y – address determines what Y line the pixels will be placed on, the X page determines which of the eight vertical 8 pixel strips the pixels will be placed on and the Data bits of the write data instruction will the be places across the X page on the specified Y line.

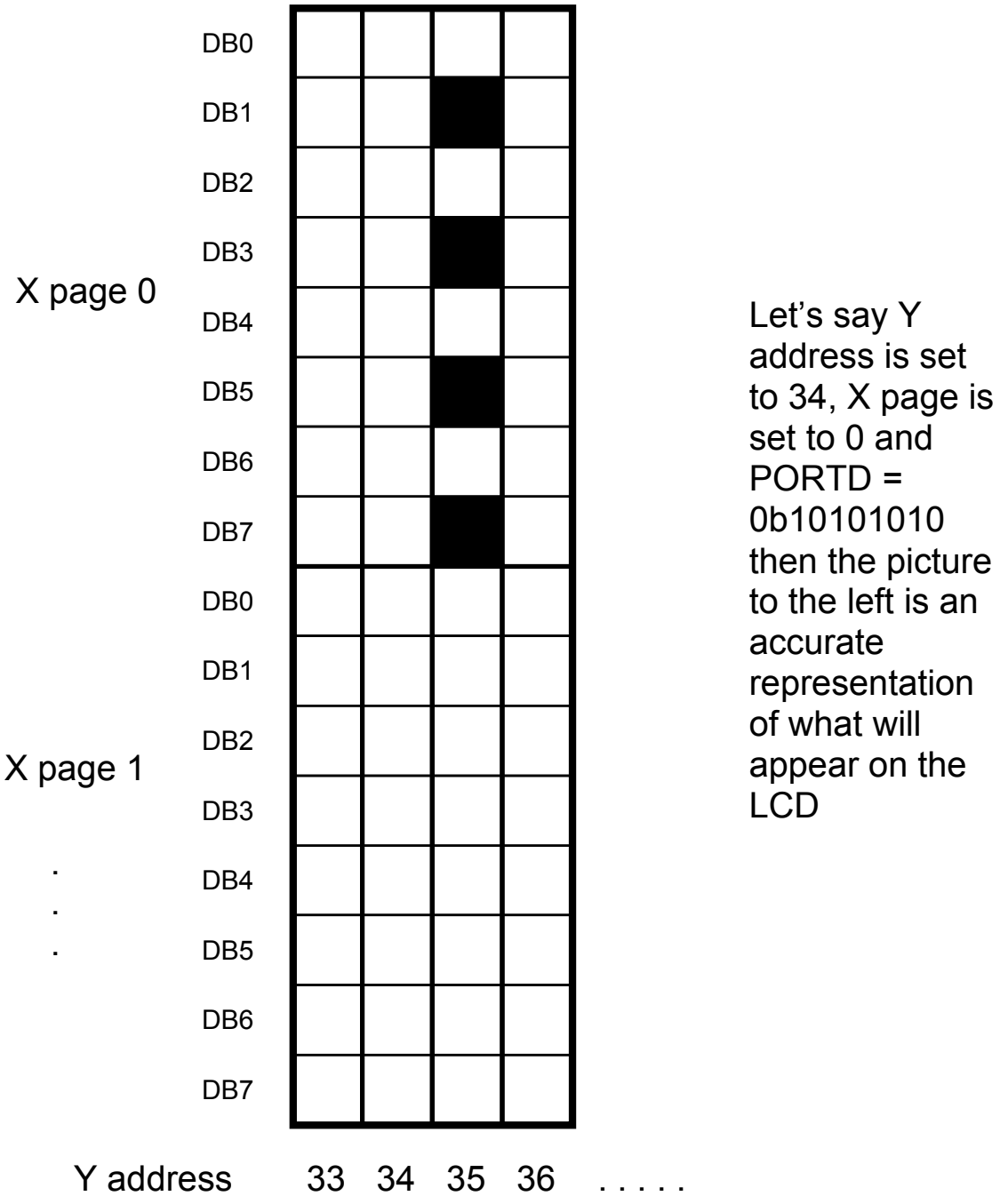


Figure 4: This image shows a zoomed in version of Figure 3. It also gives an example of how a pixel is written. If the Y address is set to 34, X page is set to 0, and a data write instruction is given with PORTD = 0b10101010, then the pixels in those locations will be dark.

5. Results

This project had four main goals. Select and Acquire a low-cost graphical LCD, write an educational primer on how it works so that other students who want to use it will have an easy time implementing it, write a driver that makes the graphical LCD easier to use, and write a character library that helps make graphical LCDs more usable in applications that could be done by alphanumeric displays. All of basic functions were successful. The main functions that the driver operates are writedata, writeinstruction, clrscreen, and displaypic, along with the facilitating functions setbit and resetbit. Although this is not in header form, all of the functions are stand alone and only require Mega32.h and delay.h to work. There is some trouble with the column select bits inverting the halves of the screen. I think that this has something to do with a subfunction resetting a column select bit and the system not selecting the right half of the screen. I plan on resolving this before releasing the final driver code. I also encoded an alphanumeric uppercase and lowercase library that is located in Appendix 5 to complete the fourth goal of this project. All four libraries are 128 x 64 pixel matrixes broken down into their proper bytes for display on the LCD. They could be broken down further into their individual characters and placed using a modification of the displaypic function. See Appendices 4 and 5 for details and code.

The headers of this driver are as follows:

void initialize(void);

This function initializes the LCD screen as well as the MCU.

unsigned char setbit(unsigned char whichbit, unsigned char I);
unsigned char resetbit(unsigned char whichbit, unsigned char I);

These functions set and reset either a control or data bit in the control or data variables denoted by I. An example of this is when enable needs to be toggled. To toggle E:

```
control = 0b00000000;       //initialize control to zero
data = 0b00000000;          //initialized data to zero
PORTD = data;               //output data to data bus
control = resetbit(E, control);         //reset enable to zero
PORTC = control;                        //output control to control bus
delay_ms(1);                            //delay to ensure toggle occurs
control = setbit(E,control);            //toggle enable to one
```

```
PORTC = control;                //output control to control bus
delay_m(1);                     //delay to ensure toggle occurs
control = resetbit(E, control); //toggle enable to zero
PORTC = control;                //output control to control bus
delay_ms(1);                    ////delay to ensure toggle occurs
```

void writedata(unsigned char data1); //write data to LCD

This function writes data to the display RAM. If the y address and xpage are set, this will send the data to the screen;

Ex: writedata(0b01010101);
//This sends these bits to the screen and they are placed wherever the current position is.

void writeinstruction(unsigned char instruct); //write instruction to LCD

This function sends an instruction to the LCD. The instructions are signified by the data bits but this is not necessary. Look at the chart in Appendix 2 and whatever D7 ~ D0 says. SO

writeinstruction(0x00111111) turns the display on and
writeinstruction(0x00111110) turns the display off.

void clrscreen(void); //Clear LCD screen

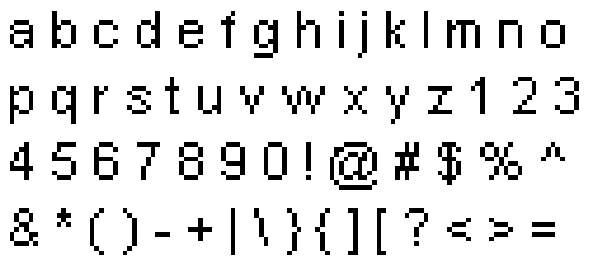
This function sets all the pixels on the screen to 0 (off);

void loadpic(unsigned char *picdata); //Load pic from picdata mem location to screen

This function loads a picture from memory onto the screen. In Appendix 5 there are 4 bitmaps converted to hexadecimal numbers. For example,

loadpic(lowercase);

would display



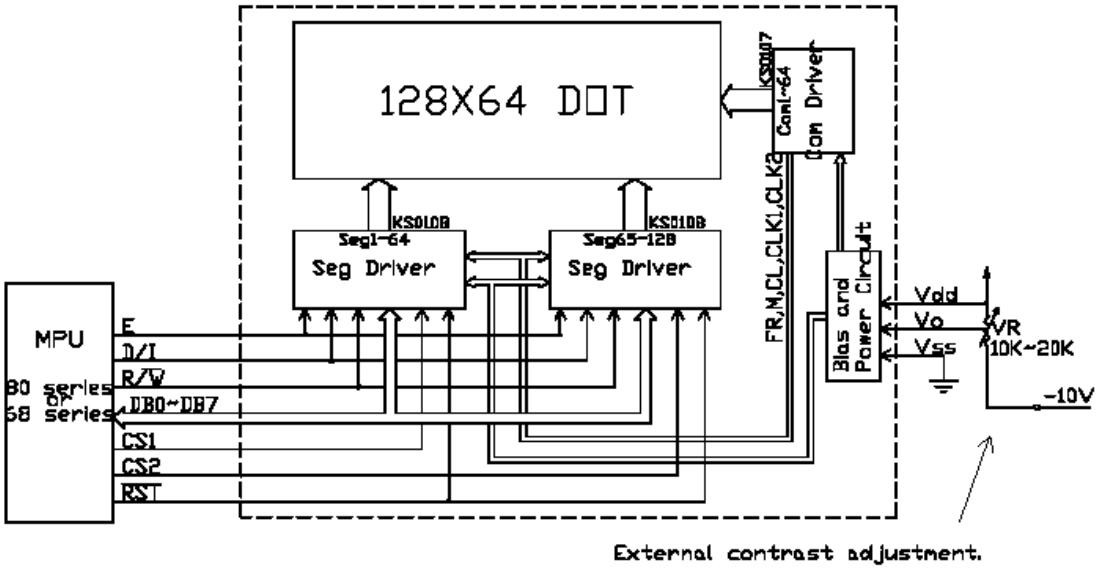
on the LCD screen.

6. Conclusion

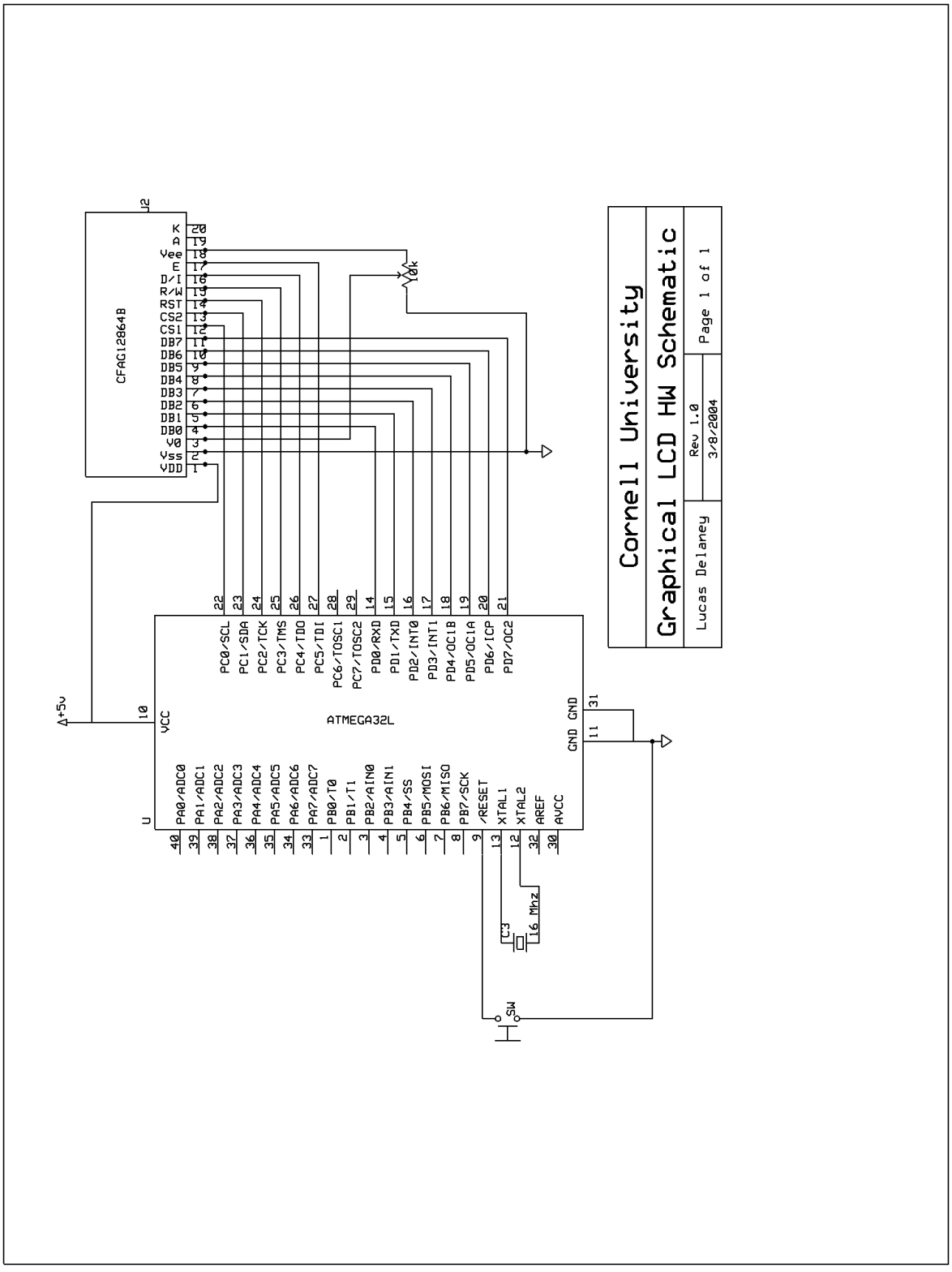
Graphical Liquid Crystal Displays can enhance the usability of almost any project. They often require more initial time and effort to get them running than do alphanumeric LCDs but they do not have the character limits of such displays. Documentation is often not readily available for graphic LCDs and fewer microcontrollers have built-in drivers for their operation. This discourages their use even further. Hopefully, the discussion in this paper made their use clearer and will help anyone who is thinking of programming a graphical LCD get through the beginning of their project as painlessly as possible.

This project was successful but I would like to implement more functions in the header. Some of the functions that were available to ECE 476 students when using the television as an output are not fully worked out in this header, but it is a start in the basic graphical display functions of an LCD. I can think of some advanced features (such as animation) that would be more difficult to implement but would be very cool.

Appendix (1) – Hardware Schematic and Block Diagram



A1.1 Internal LCD Hardware Diagram



A1.2 Schematic of Hardware Setup

Appendix (2) – Instruction Set

Instruction	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display ON/OFF	0	0	0	0	1	MPU 1	Read 1	Timing 1	1	0/1	Controls the display on or off. Internal status and display RAM data are not affected. 0:OFF, 1:ON
Set Address	0	0	0	1	Y address (0~63)						Sets the Y address in the Y address counter.
Set Page (X address)	0	0	1	0	1	1	1	Page (0 ~7)			Sets the X address at the X address register.
Display Start Line	0	0	1	1	Display start line(0~63)						Indicates the display data RAM displayed at the top of the screen.
Status Read	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0	Read status. BUSY 0:Ready 1:In operation ON/OFF 0:Display ON 1:Display OFF RESET 0:Normal 1:Reset
Write Display Data	1	0	Display Data								Writes data (DB0:7)into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read Display Data	1	1	Display Data								Reads data (DB0:7) from display data RAM to the data bus.

Appendix (3) – C ++ Testing and Initialization Code

```
//Lucas Delaney
//MEng LCD Proj
//Adv. Dr. Bruce Land
//lcdbittest.c

//this program tests each of the lines of a Crystalfonts
//CFAG128664B model Graphic LCD when connected in the
//following manner:

/*

C.7     DONT CARE
C.6     DONT CARE
C.5     ENABLE
C.4     DATA/INSTRUCTION
C.3     READ/WRITE
C.2     RESET
C.1     COLUMN SELECT 2
C.2     COLUMN SELECT 2

D.7     DATA BIT 7
D.6     DATA BIT 6
D.5     DATA BIT 5
D.4     DATA BIT 4
D.3     DATA BIT 3
D.2     DATA BIT 2
D.1     DATA BIT 1
D.0     DATA BIT 0

*/

//used as an example in the program organization doc.
#include <Mega32.h>
#include <delay.h>
//timeout values for each task
#define t1 1000
#define t2 125
#define t3 60
#define begin {
#define end }

#define DISPLAY_ON        0x3f  //0011 1111
#define DISPLAY_OFF       0x3e  //0011 1110
#define DISPLAY_STARTLINE  0xc0  //1100 0000
#define DISPLAY_PAGE_SET   0xb8  //1011 1000
#define DISPLAY_COLUMN_SET 0x40  //0100 0000

const unsigned char E  = 0b00100000;
const unsigned char DI = 0b00010000;
const unsigned char RW = 0b00001000;
const unsigned char RS = 0b00000100;
const unsigned char C2 = 0b00000010;
```

```
const unsigned char C1 = 0b00000001;
const unsigned char D7 = 0b10000000;
const unsigned char D6 = 0b01000000;
const unsigned char D5 = 0b00100000;
const unsigned char D4 = 0b00010000;
const unsigned char D3 = 0b00001000;
const unsigned char D2 = 0b00000100;
const unsigned char D1 = 0b00000010;
const unsigned char D0 = 0b00000001;



//the three task subroutines
void task1(void);             //blink at 2 or 8 Hz
//void task2(void);           //blink at 1 Hz
//void task3(void);           //detect button and modify task 1 rate

void initialize(void); //all the usual mcu stuff
void setbit(unsigned char whichbit, unsigned char I);
void resetbit(unsigned char whichbit, unsigned char I);


//void writeinstruct(unsigned char instruction, unsigned char colselect);
unsigned char time1, time2, time3;  //timeout counters
unsigned char tsk2c;                                    //task 2 counter to get to 1/2 second
unsigned char tsk3m;                                    //task 3 message to task 1
unsigned char led;                                              //light states
//unsigned bit CH2;
//*********************************************************
//timer 0 compare ISR
interrupt [TIM0_COMP] void timer0_compare(void)
begin
 //Decrement the three times if they are not already zero
 if (time1>0)      --time1;
 //if (time2>0)    --time2;
 //if (time3>0)    --time3;
end

//*********************************************************
//Entry point and task scheduler loop
void main(void)
begin
 initialize();

 //main task scheduler loop
 while(1)
 begin
  if (time1==0)   task1();
  //if (time2==0)             task2();
  //if (time3==0) task3();
 end
end

//*********************************************************
//Task subroutines
//Task 1
```

```
void task1(void)
begin
  time1=t1;  //reset the task timer
  if (tsk3m != 0) time1 >>= 2;  //check for task 3 message

  //task 1 - every 1000 ms toggle ISR

  //this code is used to test the hardware set up
  //to make sure that each line on the lcd is getting
  //the correct values from the port assignments.




  //toggle enable (E)

  if (PORTC && E)
  begin
        PORTC = PORTC & ~E;
  end
  else
  begin
        PORTC = PORTC | E;
  end



  //toggle Data/Instruction (DI)

  if (PORTC && DI)
  begin
        PORTC = PORTC & ~DI;
  end
  else
  begin
        PORTC = PORTC | DI;
  end

  //toggle Read/Write (RW)

  if (PORTC && RW)
  begin
        PORTC = PORTC & ~RW;
  end
  else
  begin
        PORTC = PORTC | RW;
  end

    //toggle Reset (RS)

  if (PORTC && RS)
  begin
        PORTC = PORTC & ~RS;
  end
```

```
else
begin
        PORTC = PORTC | RS;
end

   //toggle Column Select 2 (C2)

if (PORTC && C2)
begin
        PORTC = PORTC & ~C2;
end
else
begin
        PORTC = PORTC | C2;
end


     //toggle Column Select 1 (C1)

if (PORTC && C1)
begin
        PORTC = PORTC & ~C1;
end
else
begin
        PORTC = PORTC | C1;
end

//toggle Data Bit 7 (D7)


if (PORTD && D7)
begin
        PORTD = PORTD & ~D7;
end
else
begin
        PORTD = PORTD | D7;
end

  //toggle Data Bit 6 (D6)


if (PORTD && D6)
begin
        PORTD = PORTD & ~D6;
end
else
begin
        PORTD = PORTD | D6;
end


   //toggle Data Bit 5 (D5)
```

```
if (PORTD && D5)
begin
        PORTD = PORTD & ~D5;
end
else
begin
        PORTD = PORTD | D5;
end

 //toggle Data Bit 4 (D4)


if (PORTD && D4)
begin
        PORTD = PORTD & ~D4;
end
else
begin
        PORTD = PORTD | D4;
end


  //toggle Data Bit 3 (D3)


if (PORTD && D3)
begin
        PORTD = PORTD & ~D3;
end
else
begin
        PORTD = PORTD | D3;
end

   //toggle Data Bit 2 (D2)


if (PORTD && D2)
begin
        PORTD = PORTD & ~D2;
end
else
begin
        PORTD = PORTD | D2;
end

     //toggle Data Bit 1 (D1)


if (PORTD && D1)
begin
        PORTD = PORTD & ~D1;
end
else
begin
        PORTD = PORTD | D1;
```

```
  end

      //toggle Data Bit 0 (D0)


  if (PORTD && D0)
  begin
          PORTD = PORTD & ~D0;
  end
  else
  begin
          PORTD = PORTD | D0;
  end


end


/*

//*******************************
//Task 2
void task2(void)
begin
  time2=t2;  //reset the task timer
  if (--tsk2c == 0)  //have we waited 1/2 second?
  begin
          tsk2c = 4;                      //reload the 1/2 sec counter
    //toggle the ones bit
          led = led ^ 0x02;
          PORTB = led;
  end
end

//*******************************
//Task 3
void task3(void)
begin
  time3=t3;    //reset the task timer
  tsk3m = ~PIND & 0x01;  //generate the message for task 1
end


  */
//************************************************************
//Set it all up
void initialize(void)
begin
//set up the ports
  DDRD=0xff;      // PORT D is an output
  DDRB=0xff;    // PORT B is an output
  DDRC = 0xff; //PORT C is an output
  //PORTB=0;
  //PORTC = 0;

PORTB = 0b01010101;
```

```
  //PORTD = 0;
  //set up timer 0
  TIMSK=2;                    //turn on timer 0 cmp match ISR
  OCR0 = 250;     //set the compare re to 250 time ticks
            //prescalar to 64 and turn on clear-on-match
  TCCR0=0b00001011;

  //init the LED status (all off)
  led=0x00;

  //init the task timers
  time1=t1;
  //time2=t2;
  //time3=t3;

  //init the task 2 state variable
  //for four ticks
  tsk2c=4;

  //init the task 3 message
  //for no message
  tsk3m=0;

  //crank up the ISRs
  #asm
            sei
  #endasm
end

/*
void writeinstruct (unsigned char instruction, unsigned char colselect)
{
  delay_ms(15);

  PORTC = 00010001 ; //ENABLE_LO | CS | RS_LO
  PORTD = instruction;  //SetPortVal(DATA, Command, 1);
  PORTC = 00110001;   //;SetPortVal(CONTROL, ENABLE_HI | CS | RS_LO, 1);
  delay_ms(15);
  PORTC = 00010001 ;  //SetPortVal(CONTROL, ENABLE_LO | CS | RS_LO, 1);

  //SetPortVal(DATA, 0, 1);
}

*/
```

Appendix (4) – LCD Display Source Code

```
//Lucas Delaney
//MEng LCD Proj
//Adv. Dr. Bruce Land
//lcdbittest.c

//This program is a driver for the CFAG12864B Graphical LCD
/*Main functions:
writedata
writeinstruction
clrscreen
displaypic
*/

/*

C.7       DONT CARE
C.6       DONT CARE
C.5       ENABLE
C.4       DATA/INSTRUCTION
C.3       READ/WRITE
C.2       RESET
C.1    COLUMN SELECT 2
C.2       COLUMN SELECT 2

D.7       DATA BIT 7
D.6       DATA BIT 6
D.5       DATA BIT 5
D.4       DATA BIT 4
D.3       DATA BIT 3
D.2       DATA BIT 2
D.1       DATA BIT 1
D.0       DATA BIT 0

*/

//used as an example in the program organization doc.
#include <Mega32.h>
#include <delay.h>
//timeout values for each task
#define t1 1000
#define t2 125
#define t3 60
#define begin {
#define end }

#define DISPLAY_ON         0x3f  //0011 1111
#define DISPLAY_OFF        0x3e  //0011 1110
#define DISPLAY_STARTLINE  0xc0  //1100 0000
#define DISPLAY_PAGE_SET   0xb8  //1011 1000
#define DISPLAY_COLUMN_SET 0x40  //0100 0000

const unsigned char E  = 0b00100000;
const unsigned char DI = 0b00010000;
```

```
const unsigned char RW = 0b00001000;
const unsigned char RS = 0b00000100;
const unsigned char C2 = 0b00000010;
const unsigned char C1 = 0b00000001;
const unsigned char D7 = 0b10000000;
const unsigned char D6 = 0b01000000;
const unsigned char D5 = 0b00100000;
const unsigned char D4 = 0b00010000;
const unsigned char D3 = 0b00001000;
const unsigned char D2 = 0b00000100;
const unsigned char D1 = 0b00000010;
const unsigned char D0 = 0b00000001;

unsigned char time = 1;

//the three task subroutines
void task1(void);             //blink at 2 or 8 Hz
//void task2(void);           //blink at 1 Hz
//void task3(void);           //detect button and modify task 1 rate

void initialize(void); //all the usual mcu stuff
unsigned char setbit(unsigned char whichbit, unsigned char I);   //set bit of control or data variable
unsigned char resetbit(unsigned char whichbit, unsigned char I);//reset bit of control or data variable
void writedata(unsigned char data1);            //write data to LCD
void writeinstruction(unsigned char instruct);            //write instruction to LCD
void clrscreen(void);         //Clear LCD screen
void loadpic(unsigned char *picdata);   //Load pic from picdata mem location to screen
void setcursorpos(unsigned char x, unsigned char y);   //set cursor position

//void writeinstruct(unsigned char instruction, unsigned char colselect);
unsigned char time1, time2, time3;   //timeout counters
unsigned char tsk2c;                                    //task 2 counter to get to 1/2 second
unsigned char tsk3m;                                    //task 3 message to task 1
unsigned char led;                                              //light states
unsigned char data;
unsigned char control;


//unsigned bit CH2;
//*********************************************************
//timer 0 compare ISR
interrupt [TIM0_COMP] void timer0_compare(void)
begin
  //Decrement the three times if they are not already zero
  if (time1>0)       --time1;
  //if (time2>0)     --time2;
  //if (time3>0)     --time3;
end

//*********************************************************
//Entry point and task scheduler loop
void main(void)
begin
  initialize();

  //main task scheduler loop
```

```
 while(1)
 begin
  if (time1==0)   task1();
  //if (time2==0)              task2();
  //if (time3==0) task3();
 end
end

//*************************************************************
//Task subroutines
//Task 1
void task1(void)
begin
 time1=t1;  //reset the task timer
 if (tsk3m != 0) time1 >>= 2;  //check for task 3 message

 //task 1 - every 1000 ms toggle ISR

//toggle();
 control = 0b00000000;

 control = setbit(E, control);
 PORTC = control;
 delay_ms(500);
 control = resetbit(E,control);
 PORTC = control;

 end


/*

//*******************************
//Task 2
void task2(void)
begin
 time2=t2;  //reset the task timer
 if (--tsk2c == 0)  //have we waited 1/2 second?
 begin
          tsk2c = 4;                      //reload the 1/2 sec counter
   //toggle the ones bit
          led = led ^ 0x02;
          PORTB = led;
 end
end

//*******************************
//Task 3
void task3(void)
begin
 time3=t3;    //reset the task timer
 tsk3m = ~PIND & 0x01;  //generate the message for task 1
end


 */
```

```
//*********************************************************
//Set it all up
void initialize(void)
begin
//set up the ports
  DDRD=0xff;      // PORT D is an output
  DDRB=0xff;    // PORT B is an output
  DDRC = 0xff; //PORT C is an output
  //PORTB=0;
  //PORTC = 0;

PORTB = 0b01010101;


//initialize LCD
data = 0b00000000;
control = 0b00000101;

//toggle reset
PORTD = data;
PORTC = control;
delay_ms(time);
resetbit(RS, control);
PORTC = control;
delay_ms(time);
setbit(RS, control);
PORTC = control;
delay_ms(time);

//setup column select and initialize display
setbit(C1, control);
resetbit(C2,control);
PORTC = control;
writeinstruction(0x3E); // Display OFF
writeinstruction(0xC0);
writeinstruction(0xB8);
writeinstruction(0x40);
writeinstruction(0x3F); // Display ON
resetbit(C1, control);
setbit(C2,control);
PORTC = control;
writeinstruction(0x3E); // Display OFF
writeinstruction(0xC0);
writeinstruction(0xB8);
writeinstruction(0x40);
writeinstruction(0x3F); // Display ON

writeinstruction(0b01010101);
writeinstruction(0b10111001);



/*
PORTD = 0b00111111;
```

```
PORTC = control;
delay_ms(2);
setbit(E,control);
PORTC = control;
delay_ms(2);
resetbit(E,control);
PORTC = control;
delay_ms(2);


//set y
data = 0b01000011;
PORTD = data;

delay_ms(2);
PORTC = control;
delay_ms(2);
setbit(E,control);
PORTC = control;
delay_ms(2);
resetbit(E,control);
PORTC = control;
delay_ms(2);

//set x
data = 0b10111011;
PORTD = data;

delay_ms(2);
PORTC = control;
delay_ms(2);
setbit(E,control);
PORTC = control;
delay_ms(2);
resetbit(E,control);
PORTC = control;
delay_ms(2);

 */


 //PORTD = 0;
 //set up timer 0
 TIMSK=2;                    //turn on timer 0 cmp match ISR
 OCR0 = 250;     //set the compare re to 250 time ticks
          //prescalar to 64 and turn on clear-on-match
 TCCR0=0b00001011;

 //init the LED status (all off)
 led=0x00;

 //init the task timers
 time1=t1;
 //time2=t2;
 //time3=t3;
```

```
  //init the task 2 state variable
  //for four ticks
  tsk2c=4;

  //init the task 3 message
  //for no message
  tsk3m=0;

  //crank up the ISRs
  #asm
            sei
  #endasm
end

/*
void writeinstruct (unsigned char instruction, unsigned char colselect)
{
  delay_ms(15);

  PORTC = 00010001 ; //ENABLE_LO | CS | RS_LO
  PORTD = instruction;  //SetPortVal(DATA, Command, 1);
  PORTC = 00110001;   //;SetPortVal(CONTROL, ENABLE_HI | CS | RS_LO, 1);
  delay_ms(15);
  PORTC = 00010001 ;  //SetPortVal(CONTROL, ENABLE_LO | CS | RS_LO, 1);

  //SetPortVal(DATA, 0, 1);
}

*/


unsigned char setbit(unsigned char whichbit, unsigned char I)
{
            unsigned char A;
            switch (whichbit)
            {

                        case 'D7':
                                    A = I | D7;
                        break;
                        case 'D6':
                                    A = I | D6;
                        break;
            case 'E':
            case 'D5':
                                    A = I | D5;
                        break;
                        case 'DI':
                        case 'D4':
                                    A = I | D4;
                        break;
            case 'RW':
            case 'D3':
                                    A = I | D3;
                        break;
```

```
		case 'RS':
		case 'D2':
			A = I | D2;
		break;
	case 'C2':
	case 'D1':
			A = I | D1;
		break;
		case 'C1':
		case 'D0':
			A = I | D0;
		break;
	}
	return A;
}


unsigned char resetbit(unsigned char whichbit, unsigned char I)
{
	unsigned char A;
	switch (whichbit)
	{

		case 'D7':
			A = I & ~D7;
		break;
		case 'D6':
			A = I & ~D6;
		break;
	case 'E':
	case 'D5':
			A = I & ~D5;
		break;
		case 'DI':
		case 'D4':
			A = I & ~D4;
		break;
	case 'RW':
	case 'D3':
			A = I & ~D3;
		break;
		case 'RS':
		case 'D2':
			A = I & ~D2;
		break;
	case 'C2':
	case 'D1':
			A = I & ~D1;
		break;
		case 'C1':
		case 'D0':
			A = I & ~D0;
		break;
	}
	return A;
}
```

```
void writedata(unsigned char data1)
{
	control = setbit(E, control);//Toggle enable
	PORTC = control;
	delay_ms(time);
	control = setbit(DI, control);//set data
	control = resetbit(RW, control); //set write
	PORTC = control;
	delay_ms(time);
	PORTD = data1;	//output data
	control = resetbit(E, control); //toggle enable
	PORTC = control;
	delay_ms(time);
}


void writeinstruction(unsigned char instruct)
{
	control = setbit(E, control);//toggle enable
	PORTC = control;
	delay_ms(time);
	control = resetbit(DI, control);//set instruction
	control  = resetbit(RW, control); //set write
	PORTC = control;
	delay_ms(time);
	PORTD = instruct;	//output instruction
	control = resetbit(E, control);	//toggle enable
	PORTC = control;
	delay_ms(time);
}


//This function clears the LCD scream
void clrscreen(void)
{
	unsigned char page;
	unsigned char column;
	control = 0;

	writeinstruction(0b01000000);	//set page
	for (page = 0; page <8; page++)	//clear left side
	{
		control = setbit(C1, control);	//select left side
		control = resetbit(C2, control);
		PORTC = control;
		writeinstruction(0b10111000 + page); //sweep pages
		writeinstruction(0b01000000);

		for (column = 0; column <128; column++)
		{				//clear right side
			if(column ==64)
			{
				control = resetbit(C1, control);	//select right side
				control = setbit(C2, control);
```

```
                                        PORTC = control;
                                        writeinstruction(0b10111000 + page);//sweep pages
                                        writeinstruction(0b01000000);
                                }

                                writedata(0);       //for each set of 8 pixels, write no data
                        }
                }
}


void loadpic(unsigned char *picdata)
{
        unsigned char page;
        unsigned char column;
        page = 0;
        column = 0;
        writeinstruction(0b01000000);       //set page
        for (page = 0; page <8; page++)
        {
                control = setbit(C1, control);    //select left side
                control = resetbit(C2, control);
                PORTC = control;
                writeinstruction(0b10111000 + page);//sweep pages
                writeinstruction(0b01000000);

                        for (column = 0; column <128; column++)
                {                               //move to right side
                        if(column ==64)
                        {
                                control = resetbit(C1, control);  //select right side
                                control = setbit(C2, control);
                                PORTC = control;
                                writeinstruction(0b10111000 + page);//sweep pages
                                writeinstruction(0b01000000);
                        }

                        writedata(picdata[(128*page)+column]);      //for each set of 8 pixels, write data
in pic array
                }
         }
}

void setcursorpos(unsigned char x, unsigned char y)
{

        writeinstruction(0b01000000); //set y address to zero

        //if x falls on the left side of the screen
        if (x < 64)
        {
                setbit(C1,control);             //choose side
                resetbit(C2,control);
                writeinstruction(0b10101000 + (y / 8));  //page
                writeinstruction(0b01000000 + x);   //column
                delay_ms(1);
```

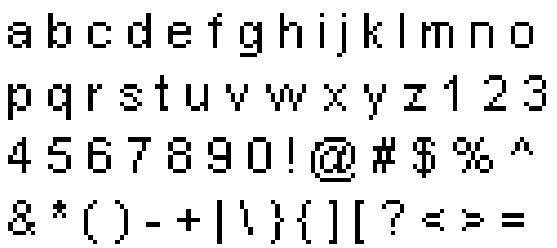
```
        }
        else
        //if x is greater than 64 and falls on the right side of the screen
        {
                setbit(C2,control);             //choose side
                resetbit(C1,control);
                writeinstruction(0b10101000 + (y / 8)); //page
                writeinstruction(0b01000000+ x - 64);  //column
                delay_ms(1);

        }
}
```

Appendix (5) – Character libraries

```
const char lowercase[] = {
0x07, 0x87, 0x47, 0x47, 0x47, 0x87, 0x07, 0x07, 0x07, 0x07, 0xF7, 0x87, 0x47, 0x47, 0x87, 0x07,
0x0F, 0x0F, 0x0F, 0x8F, 0x4F, 0x4F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0x8F, 0x4F, 0x4F, 0x8F,
0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0x8F, 0x4F, 0x4F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0x4F, 0xEF,
0x5F, 0x0F, 0x0F, 0x0F, 0x0F, 0x8F, 0x4F, 0x4F, 0x8F, 0xCF, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x8F,
0x4F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0xDF, 0x0F, 0x0F, 0x0F, 0x0F, 0xDF, 0x0F, 0x0F, 0x0F,
0x0F, 0xFF, 0x0F, 0x8F, 0x4F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0xCF, 0x8F,
0x4F, 0xCF, 0x4F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0xCF, 0x4F, 0x4F, 0x4F, 0x8F, 0x0F, 0x0F,
0x0F, 0x0F, 0x8F, 0x4F, 0x4F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x00, 0x06, 0x09, 0x09, 0x05, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x04, 0x08, 0x08, 0x07, 0x00,
0x00, 0x00, 0x00, 0x07, 0x08, 0x08, 0x08, 0x04, 0x00, 0x00, 0x00, 0x00, 0x07, 0x08, 0x08, 0x04,
0x0F, 0x00, 0x00, 0x00, 0x00, 0x07, 0x09, 0x09, 0x09, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F,
0x00, 0x00, 0x00, 0x00, 0x00, 0x27, 0x28, 0x28, 0x24, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x20, 0x1F, 0x00, 0x00, 0x00,
0x00, 0x0F, 0x01, 0x06, 0x08, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x0F, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00,
0x00, 0x00, 0x07, 0x08, 0x08, 0x08, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xF0, 0x20, 0x10, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x10, 0x10, 0x20, 0xF0, 0x00,
0x00, 0x00, 0x00, 0xF0, 0x20, 0x10, 0x00, 0x00, 0x00, 0x00, 0x20, 0x50, 0x50, 0x90, 0x20, 0x00,
0x00, 0x00, 0x10, 0xFC, 0x10, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x00, 0x00, 0x30, 0xC0, 0x00, 0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x30, 0xC0, 0x00, 0xE0, 0x10,
0xE0, 0x00, 0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x10, 0x20, 0xC0, 0x20, 0x10, 0x00, 0x00, 0x00,
0x00, 0x30, 0xC0, 0x00, 0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0xD0, 0x30, 0x10, 0x00,
0x00, 0x00, 0x00, 0x10, 0x08, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x04, 0x84, 0x44,
0x38, 0x00, 0x00, 0x00, 0x00, 0x08, 0x04, 0x24, 0x24, 0xD8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0F, 0x01, 0x02, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x01, 0x0F, 0x00,
0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x02, 0x01, 0x00,
0x00, 0x00, 0x00, 0x03, 0x02, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x01, 0x03, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00,
0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x01, 0x00, 0x01, 0x02, 0x00, 0x00, 0x00,
0x00, 0x00, 0x08, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x03, 0x02, 0x02, 0x02, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x03, 0x02, 0x02,
0x02, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x2C, 0x22, 0xFF, 0x20, 0x00, 0x00, 0x00, 0x00, 0x4C, 0x8B, 0x89, 0x89, 0x71, 0x00,
0x00, 0x00, 0x00, 0x7E, 0x89, 0x89, 0x89, 0x72, 0x00, 0x00, 0x00, 0x00, 0x01, 0xE1, 0x19, 0x07,
0x01, 0x00, 0x00, 0x00, 0x00, 0x76, 0x89, 0x89, 0x89, 0x76, 0x00, 0x00, 0x00, 0x00, 0x4E, 0x91,
0x91, 0x91, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x81, 0x81, 0x81, 0x7E, 0x00, 0x00, 0x00, 0x00,
0xBF, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x06, 0x72, 0x89, 0x85, 0x45, 0xF9, 0x8D, 0x42, 0x3C, 0x00,
0x00, 0x00, 0x00, 0xE4, 0x3C, 0xE7, 0x3C, 0x27, 0x00, 0x00, 0x00, 0x00, 0x46, 0x89, 0xFF, 0x89,
0x72, 0x00, 0x00, 0x00, 0x00, 0x06, 0x09, 0x89, 0x66, 0x18, 0x66, 0x91, 0x90, 0x60, 0x00, 0x00,
0x00, 0x00, 0x08, 0x06, 0x01, 0x06, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0x40, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80, 0x40, 0x00, 0x00, 0x00, 0x00, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80, 0x40, 0x00, 0x01, 0x02, 0x02, 0x42, 0xC2, 0x02, 0x02, 0x02, 0x01, 0xC0,
0x40, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x40, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 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, 0x18, 0x25, 0x22, 0x26, 0x19, 0x28, 0x00, 0x00, 0x00, 0x00, 0x02, 0x01, 0x02, 0x00, 0x00,
0x00, 0x00, 0x3F, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x3F, 0x00, 0x00, 0x00, 0x00,
0x08, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0x04, 0x04, 0x1F, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x30, 0x00, 0x00, 0x00, 0x00, 0x80, 0x7B, 0x04, 0x00,
0x00, 0x00, 0x04, 0x7B, 0x80, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF,
```

```
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2C, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0A,
0x0A, 0x0A, 0x11, 0x00, 0x00, 0x00, 0x00, 0x11, 0x0A, 0x0A, 0x0A, 0x04, 0x00, 0x00, 0x00, 0x00,
0x0A, 0x0A, 0x0A, 0x0A, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

lowercase corresponds to:

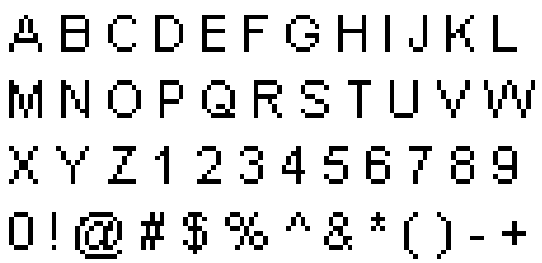


a b c d e f g h i j k l m n o
p q r s t u v w x y z 1 2 3
4 5 6 7 8 9 0 ! @ # $ % ^
& * ( ) - + | \ } {] [? < > =

```
const char capitalletters[] = {
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0xC7, 0x27, 0xC7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0xEF, 0x2F, 0x2F, 0x2F, 0x2F, 0xCF, 0x0F, 0x0F, 0x0F, 0x0F, 0x8F, 0x4F, 0x2F, 0x2F, 0x2F, 0x4F,
0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x2F, 0x2F, 0x2F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x2F,
0x2F, 0x2F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x2F, 0x2F, 0x2F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F,
0x8F, 0x4F, 0x2F, 0x2F, 0x2F, 0x4F, 0x8F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F,
0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F,
0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x8F, 0x4F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x06, 0x05, 0x04, 0x05, 0x06, 0x18, 0x00, 0x00, 0x00, 0x00,
0x1F, 0x11, 0x11, 0x11, 0x11, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x07, 0x08, 0x10, 0x10, 0x10, 0x08,
0x00, 0x00, 0x00, 0x00, 0x1F, 0x10, 0x10, 0x10, 0x08, 0x07, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x11,
0x11, 0x11, 0x11, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
0x07, 0x08, 0x10, 0x10, 0x12, 0x0A, 0x06, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x01, 0x01, 0x01, 0x01,
0x1F, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x10, 0x10, 0x0F, 0x00, 0x00,
0x00, 0x00, 0x1F, 0x02, 0x01, 0x03, 0x0C, 0x10, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x10, 0x10, 0x10,
0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x30, 0xC0, 0x00, 0xC0, 0x30, 0xF8, 0x00, 0x00, 0x00, 0x00,
0xF8, 0x10, 0x60, 0x80, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x10, 0x08, 0x08, 0x08, 0x10,
0xE0, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x88, 0x88, 0x88, 0x70, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x10,
0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x48, 0x48, 0xC8, 0x48, 0x30, 0x00,
0x00, 0x00, 0x00, 0x30, 0x48, 0x48, 0x88, 0x88, 0x10, 0x00, 0x00, 0x00, 0x00, 0x08, 0x08, 0xF8,
0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x00,
0x18, 0x60, 0x80, 0x00, 0x80, 0x60, 0x18, 0x00, 0x00, 0x00, 0x18, 0xE0, 0x00, 0x80, 0x70, 0x08,
0x70, 0x80, 0x00, 0xE0, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x01, 0x06, 0x01, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00,
0x07, 0x00, 0x00, 0x01, 0x02, 0x07, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x04, 0x04, 0x04, 0x02,
```

```
0x01, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02,
0x04, 0x05, 0x05, 0x06, 0x05, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x03, 0x04, 0x00,
0x00, 0x00, 0x00, 0x02, 0x04, 0x04, 0x04, 0x04, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x04, 0x04, 0x04, 0x04, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x06, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x06, 0x01, 0x00, 0x00,
0x00, 0x01, 0x06, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xCC, 0x30, 0x30, 0xCC, 0x02, 0x00, 0x00, 0x00, 0x00, 0x02,
0x0C, 0x10, 0xE0, 0x10, 0x0C, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x62, 0x1A, 0x06, 0x02,
0x00, 0x00, 0x00, 0x00, 0x08, 0x04, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x82, 0x42,
0x22, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x84, 0x02, 0x12, 0x12, 0xEC, 0x00, 0x00, 0x00, 0x00, 0x60,
0x58, 0x44, 0xFE, 0x40, 0x00, 0x00, 0x00, 0x00, 0x98, 0x16, 0x12, 0x12, 0xE2, 0x00, 0x00, 0x00,
0x00, 0xFC, 0x12, 0x12, 0x12, 0xE4, 0x00, 0x00, 0x00, 0x00, 0x02, 0xC2, 0x32, 0x0E, 0x02, 0x00,
0x00, 0x00, 0x00, 0xEC, 0x12, 0x12, 0x12, 0xEC, 0x00, 0x00, 0x00, 0x00, 0x9C, 0x22, 0x22, 0x22,
0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x80, 0x80, 0x00, 0x01, 0x00, 0x00, 0x00, 0x80, 0x00,
0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x81, 0x01, 0x01, 0x01, 0x01, 0x01,
0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x01, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x81, 0x01, 0x01,
0x01, 0x01, 0x00, 0x00, 0x80, 0x80, 0x00, 0x01, 0x01, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x81, 0x81, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x81, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81, 0x00, 0x00, 0x00, 0x00,
0x80, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x40, 0x40, 0x40, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x5F, 0x00,
0x00, 0x00, 0x00, 0x7C, 0x83, 0x39, 0x44, 0x42, 0x22, 0x7C, 0x46, 0x21, 0x9E, 0x00, 0x00, 0x00,
0x00, 0x72, 0x1E, 0x73, 0x1E, 0x13, 0x00, 0x00, 0x00, 0x00, 0x23, 0x44, 0xFF, 0x44, 0x39, 0x00,
0x00, 0x00, 0x00, 0x03, 0x04, 0x44, 0x33, 0x0C, 0x33, 0x48, 0x48, 0x30, 0x00, 0x00, 0x00, 0x00,
0x04, 0x03, 0x00, 0x03, 0x04, 0x00, 0x00, 0x00, 0x30, 0x4B, 0x44, 0x4C, 0x33, 0x50, 0x00, 0x00,
0x00, 0x00, 0x05, 0x03, 0x05, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x81, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00, 0x08, 0x08,
0x3E, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 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, 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, 0x01, 0x00, 0x00, 0x00, 0x00,
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
};
```

capitalletters corresponds to:




```
const char biglower[] = {
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
```

```
0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0xCF, 0x2F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x2F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x62, 0x91, 0x89, 0x89, 0x49, 0xFE, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0x42, 0x81, 0x81, 0x81, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E,
0x81, 0x81, 0x81, 0x81, 0x42, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x81, 0x81, 0x81, 0x42,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x89, 0x89, 0x89, 0x89, 0x4E, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0xFF, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x81, 0x81, 0x81, 0x42,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x02, 0x01, 0x01, 0x01, 0xFE, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xFF, 0x10, 0x08, 0x14, 0x62, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x04, 0x04, 0x04, 0x04,
0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x04, 0x04, 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, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x04,
0x02, 0x02, 0x02, 0xFC, 0x04, 0x02, 0x02, 0x02, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE,
0x04, 0x02, 0x02, 0x02, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x02, 0x02, 0x02, 0x02,
0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x84, 0x02, 0x02, 0x02, 0xFC, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFC, 0x02, 0x02, 0x02, 0x84, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE,
0x04, 0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x8C, 0x12, 0x12, 0x22, 0x22, 0xC4, 0x00, 0x00,
0x00, 0x00, 0x00, 0x02, 0xFF, 0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x00,
0x80, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00,
0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01,
0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0xC0, 0x00, 0xC0, 0x30, 0x0C, 0x00, 0x00, 0x00,
0x00, 0x0C, 0xF0, 0x00, 0xC0, 0x38, 0x04, 0x38, 0xC0, 0x00, 0xF0, 0x0C, 0x00, 0x00, 0x00, 0x00,
0x04, 0x08, 0x90, 0x60, 0x90, 0x08, 0x04, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0xC0, 0x00, 0xC0,
0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x04, 0xC4, 0x34, 0x0C, 0x04, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x01, 0x00, 0x00, 0x00, 0x01, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x11, 0x0E, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x03, 0x02, 0x02, 0x02, 0x02, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

biglower corresponds to:

```
const char bigcaps[] = {
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0xE7, 0x17, 0xE7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x0F, 0xFF, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0xEF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xCF, 0x2F,
0x1F, 0x1F, 0x1F, 0x2F, 0x4F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x1F, 0x1F, 0x1F, 0x1F,
0x2F, 0xCF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x0F,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0xCF, 0x2F, 0x1F, 0x1F, 0x1F, 0x1F, 0x2F, 0x4F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x00, 0x00, 0x00, 0x30, 0x0C, 0x07, 0x04, 0x04, 0x04, 0x07, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3F, 0x21, 0x21, 0x21, 0x21, 0x21, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x10,
0x20, 0x20, 0x20, 0x10, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x20, 0x20, 0x20, 0x20,
0x10, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x21, 0x21, 0x21, 0x21, 0x21, 0x21, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0F, 0x10, 0x20, 0x20, 0x22, 0x22, 0x12, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F,
0x01, 0x01, 0x01, 0x01, 0x01, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00,
0x00, 0x00, 0x18, 0x20, 0x20, 0x20, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x10, 0x00, 0x00, 0x00, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x60, 0x80, 0x00,
0x00, 0x00, 0x80, 0x60, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x20, 0xC0, 0x00, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0x10, 0x10, 0x10, 0x20, 0xC0,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x10, 0x10, 0x10, 0x10, 0x10, 0xE0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0x10, 0x10, 0x10, 0x20, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xF0, 0x10, 0x10, 0x10, 0x10, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x10,
0x10, 0x10, 0x10, 0x10, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x3F, 0x04, 0x02, 0x03, 0x04, 0x08, 0x10, 0x20, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3F, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x01, 0x0E,
0x30, 0x0E, 0x01, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x03, 0x0C,
0x10, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x20, 0x20, 0x10, 0x0F,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x02, 0x02, 0x02, 0x02, 0x02, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x28, 0x28, 0x10, 0x2F, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3F, 0x02, 0x02, 0x02, 0x06, 0x1A, 0x21, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x21,
0x21, 0x21, 0x22, 0x22, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x10, 0x10, 0x10, 0xF0, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0,
0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0xC0, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x30, 0xC0, 0x00, 0x00, 0x00, 0xE0, 0x10, 0xE0,
0x00, 0x00, 0x00, 0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x10, 0x60, 0x80, 0x00, 0x80, 0x60, 0x10,
0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x60, 0x80, 0x00, 0x80, 0x60, 0x10, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x10, 0x10, 0x10, 0xD0, 0x30, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F,
0x10, 0x20, 0x20, 0x20, 0x10, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x0C, 0x30,
0x0C, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0E, 0x30, 0x0E, 0x01, 0x00, 0x01,
0x0E, 0x30, 0x0E, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x18, 0x04, 0x03, 0x04, 0x18, 0x20,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x3E, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x20, 0x30, 0x2C, 0x23, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

bigcaps corresponds to:

References

Datasheets:

Crystalfontz, America Inc, *CFAG12864B-WGH-V:128 x 64 Graphic LCD*, Valleyford, WA, 2004.

Displaytech, Ltd., 64128A*:128 x 64 Graphic LCD, Version 1.1,* Carlsbad, CA.

Samsung Digital, Inc, *KS0108: 64 x 64 LCD Driver*, 2004.

Links:

www.crystalfontz.com
www.samsung.com
www.displaytech-us.com