DGEsc: Dungeon Escape Game (Reflection)

CSEE 4840 Embedded System Design

Spring 2016

Yi Wu (yw2707)

Yuxin Yang (yy2586)

Shengjia Zhang (sz2547)

Xiaoqing Yong (xy2246)

Content

1. Overview

This game is nothing like anything on the market. It is based on a real story happened to Yutao Gu, a senior in Columbia University. As Columbia EE students, we have taken classes on communication circuits, and RF design. The legendary yet tragic story of one of our outstanding alumni Edwin Howard Armstrong, the inventor of FM radio, inspired us to develop this game. You will further explore the details and plots of this Game Plot section.

In this project we implemented a third person 2D RPG themed game on the SoCkit board using VGA monitor to display and keyboard to control. Player will control the character in the game to explore the maps, trigger plots, and finally beat the enemies. Following is an example of the game scene and our main character.

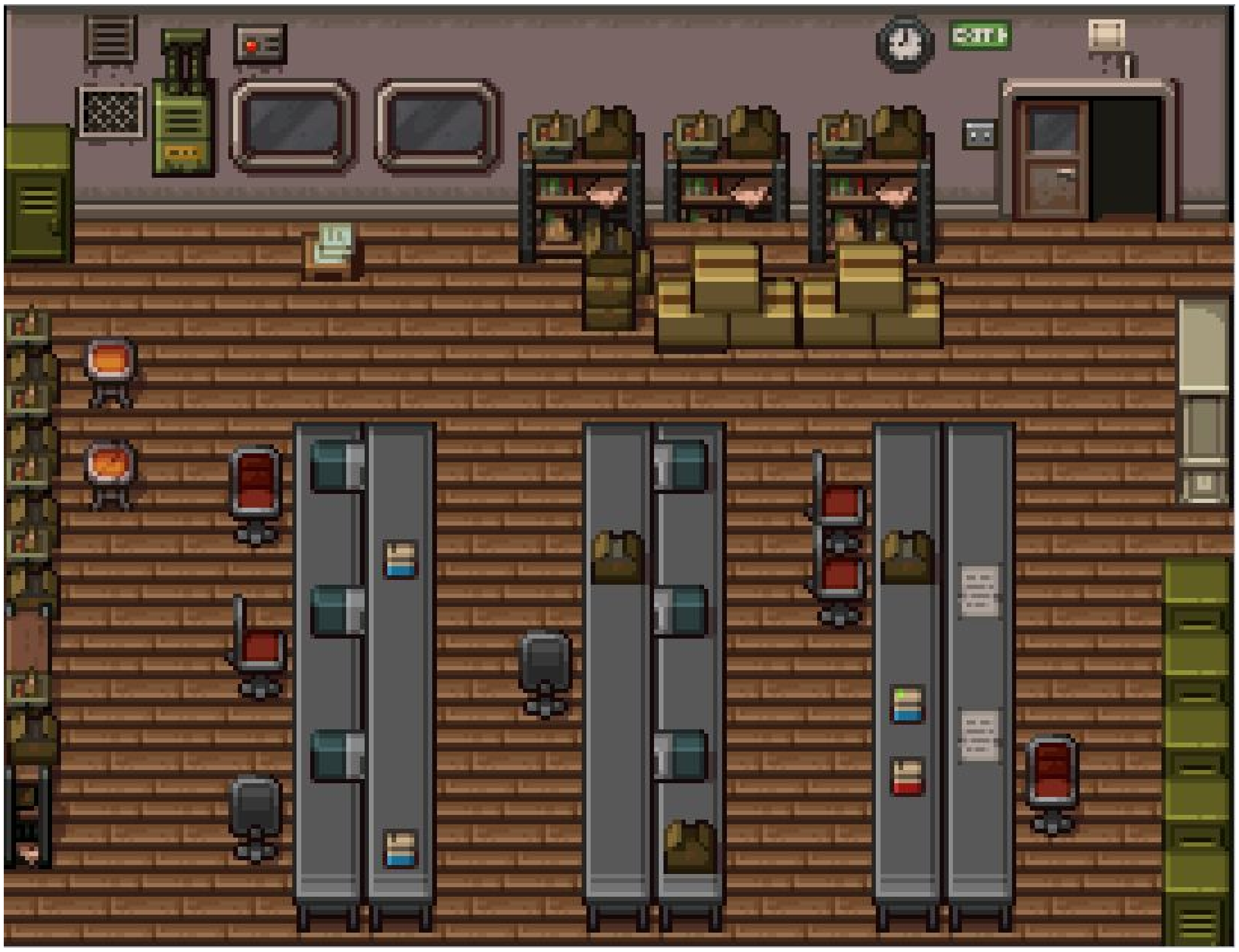


Figure 1: Example of a game scene

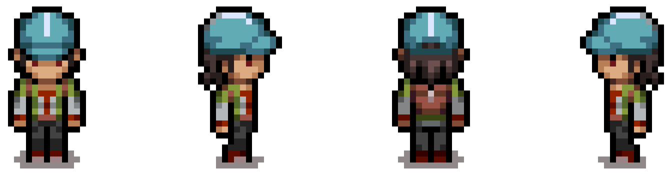


Figure 2: Eric, main character of the game

1.1 Game Plot

Our game story happens on the 12th floor Mudd building, Columbia University, in embedded system lab. Where our main character, Eric, is doing his homework on this local online chatting room in this lab room. He finishes late at night, and there is nobody else in this room. Suddenly, some stranger greeted him back using his console, and the power went off. As Eric explores the cause of this power down, he revealed a giant secret buried in Mudd Building... To prevent spoiler, we strongly encourage you to play this game yourself, to share the confusion, the horror and oddity with Eric.

1.2 Overall Design

We use a third-party software RPG maker to generate maps, as well as characters, objects and monsters. We convert the generated picture files first into mif files, and push the memory line by line to the memory module in FPGA.

SystemVerilog are used to program, store and display sprites with VGA interface on FPGA board, receive maps and control commands sent by software as well as processing sounds. On the software perspective, we write our program in C language for map and object generation, receiving keyboard input, handling game logic and other application of game rules.

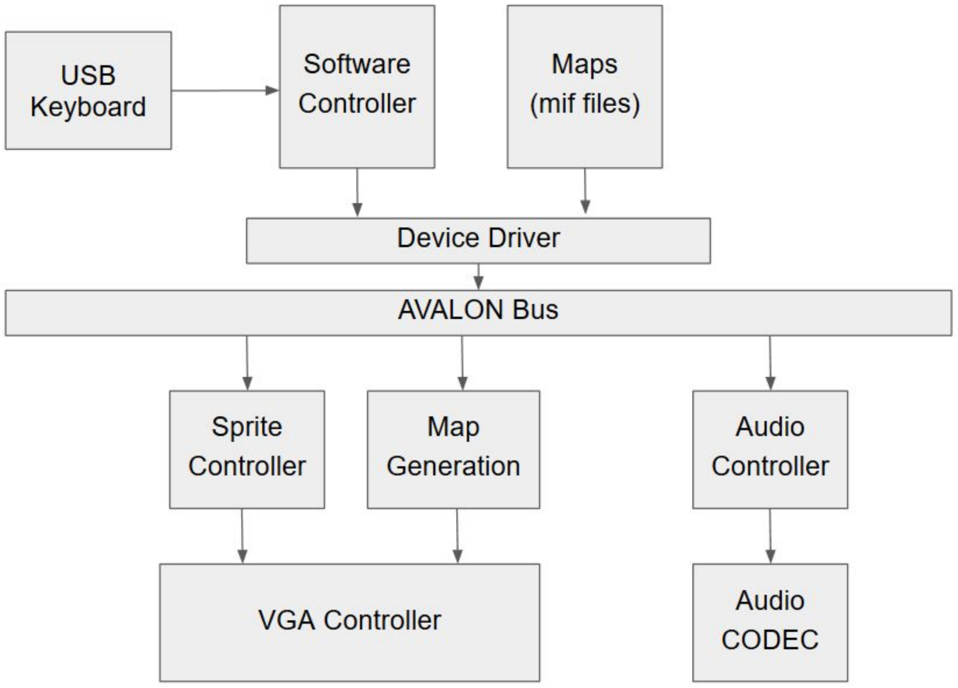


Figure 3. Overall Design

2. VGA Module

We use a VGA monitor to display our game. The SOCkit board we use includes a 15-pin D-SUB connector for VGA output. The VGA synchronization signals are provided directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are used) is used to produce the analog data signals (red, green, and blue). The following figure gives the associated schematic.

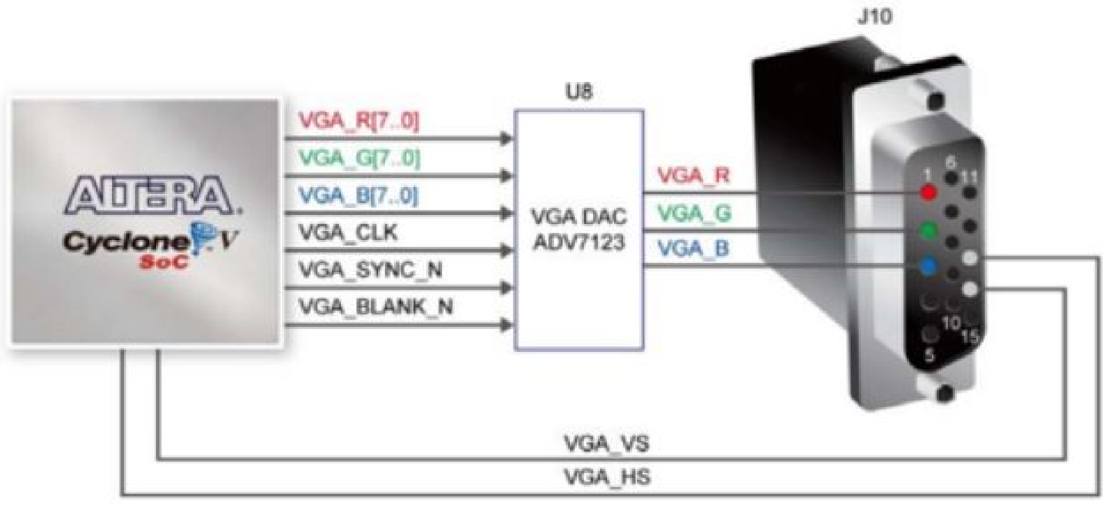


Figure 4. Connection between Cyclone V Sockit board and VGA

2.1 Background

The dimension of all the maps are 640 × 480. We first construct these maps in RPG Maker, then converted all the maps into MIF files and stored in software. Every time the character enters a scene, we transmit the corresponding MIF file to the hardware memory module and display on VGA using frame buffer reading from our memory.

2.2 Sprite

We use sprites to display moving object, which can produce repeated pattern of pictures using a small size of different elements. In this way we can minimizing the requirement for large screen pixels RGB information. The Sprite use multiple layers to overlap the pictures and generate a movement effect by control the movement of small elements in different layers. Our main character Eric has four facing directions. When the he moves, we designed two extra transition statuses in between to simulate continuous movement. So there are 12 images for Eric in total. Those 12 images are stored directly in FPGA memories.

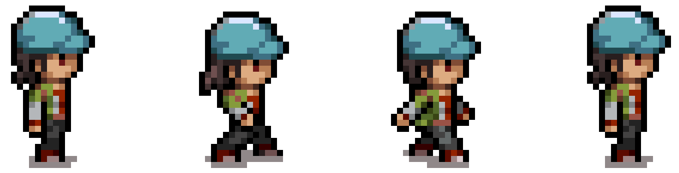
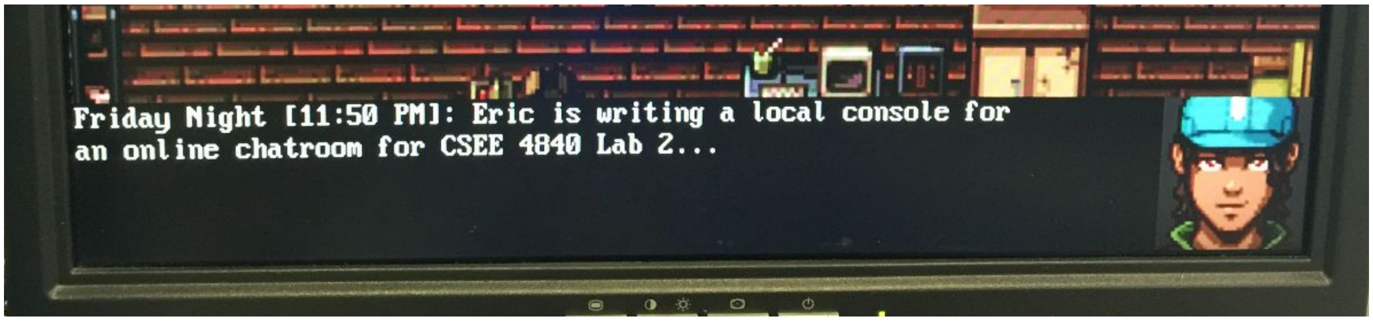


Figure 5. Simulate continuous movements

2.3 Dialog

We set the lower part of the screen to display the dialog and a portrait of the character using framebuffer.



Friday Night [11:50 PM]: Eric is writing a local console for an online chatroom for CSEE 4840 Lab 2...

Figure 6. Displaying a dialog

To print the characters on screen, we chose to use software to rewrite the memory module with the lower part of map covered by our dialog. We used the FONT matrix provided in LAB2 “putchar.h” to prepare the characters in software. And with the same method as printing map, we write into memory module through Kernel code.

2.4 Visual Effects

We can achieve many visual effects by manipulating R, G, B values of each background pixel. We designed effects of blackout and light flashing and entering alternate world.

For example:

For blackout on a certain map, we set the R, G and B value of each pixel on background to be identical (e.g. all set to the R’s value) and shift them to the right by one bit to reduce the brightness.

For the nuclear (“alternative world”) effect, we achieve this by turning on and off the blackout effect. To enter alternate world, we simply shift our R data by 2 bits.

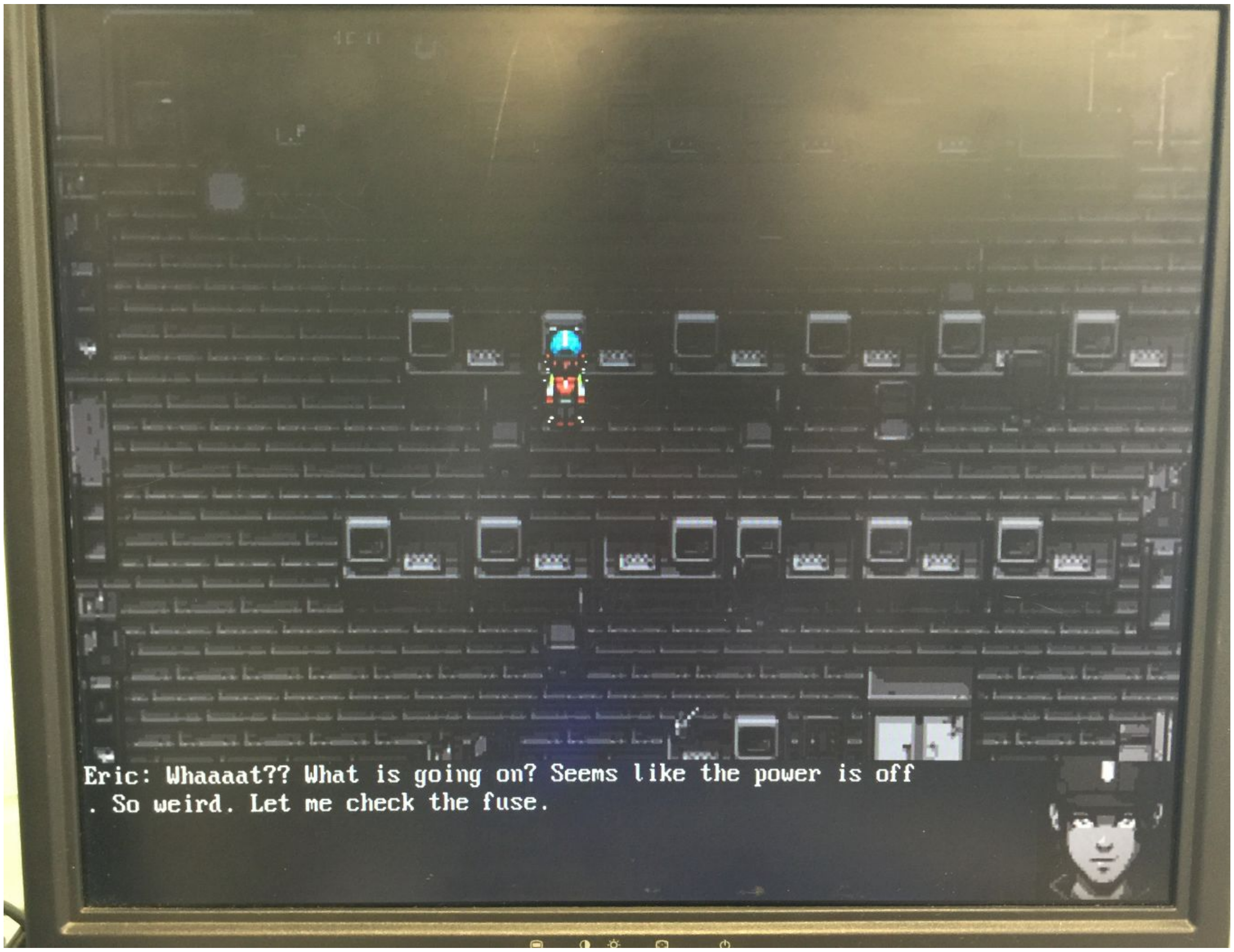

Eric: Whaaaat?? What is going on? Seems like the power is off
. So weird. Let me check the fuse.

Figure 7. Blackout effect

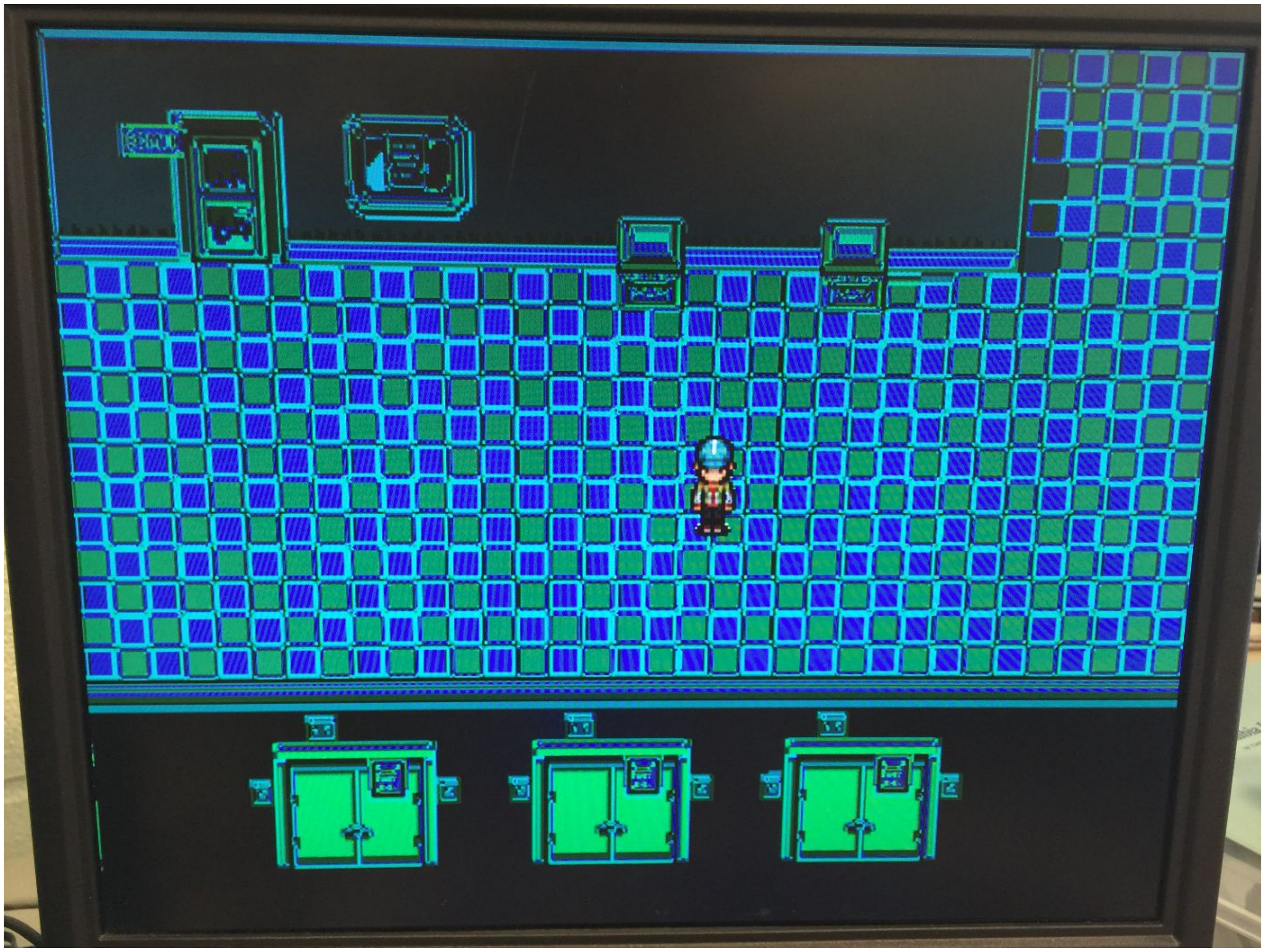


Figure 8. ‘Alternative world’ effect

3. Software - Hardware Interface

3.1 Software - Hardware Communication

Software have to tell hardware about RGB information, map data, character coordinates, etc. The following figure is our communication scheme.

Address	Data	Address	Data
0	color[11:8]: R	9	y_in[7:0]
1	color[7:4]: G	10	char_crl[7:0]
2	color[3:0] B	11	map_crl[7:0]
3	map_line_num[23:16]	12	filt_crl[7:0]
4	map_line_num[15:8]	13	music_crl[7:0]
5	map_line_num[7:0]	14	xv_in[15:8]
6	x_in[15:8]	15	xv_in[7:0]
7	x_in[7:0]	16	yv_in[15:8]
8	y_in[15:8]	17	yv_in[7:0]

Table 1. Communication scheme

The first 3 bits are for map data transfer, and the next 3 bits are for map address transfer. Bit 6-9 transfers the coordinates of Eric sprite, and bit 10 is for the transfer of the facing direction of Eric. Bit 11 is to set map write_enable so that it allows us to write into memory module if set high. Bit 12 is the filter control for us to control what filter to used upon the map (e.g. power_down or alternative world theme). Bit 13 is for us to control the music clips played in this game. And finally bit 14 to 17 are the coordinates for vilain Eric.

3.2 Sprite Control

The sprite stored in SRAM with mif format can be displayed on screen on given X and Y coordinates. This is done by the sprite controller. It takes X and Y coordinates from game logic controller and access the sprite data on the SRAM with given address, then the sprite controller can generate RGB values for the VGA controller.

Example calculation:

```
adr_eric <= ((hc-x_in) + (vc-(y_in-54))*30);
```

This is the code we used in our vga_led_emulator.v file to calculate the address to read in eric.mif. Knowling x_in and y_in, we need to do some row and column transform operations to make it appear right on the screen. In this case, 54 is the height of the sprite, and 30 is the width, so we calculate x and y positions by calculate where VGA is printing, in the small 54*30 box.

4. Game Logic and Control

4.1 Keyboard Controllers

We only use four keyboard keys throughout the whole game, which are ↑ , ↓ ,← , →, corresponding to moving Up, Down, Left, Right. Player can also check items and advance the plot using these keys. We also used those four keys to skip the conversations in storyline.

4.2 Map System

Instead of using tilesets, we are using a large framebuffer (which is the entire screen) to print out our background/map. Due to Sockit Board memory block size limitation, we are only able to save one 640*480 background mif file in the memory of the Sockit Board. Therefore, we decided to save all the map mif files in the software (which has 1 GB memory) and when we want to switch the map, we read all mif files from the software and send it to the Sockit board. The Sockit board frame buffer will print the map pixel by pixel on the screen and at the same time, save them into the memory block so that the map will stay on the screen till we switch to next map.

However, since we are not using tilesets to create the map, the map itself is actually a large background. Therefore, in order to tell the player where he or she is allow to go or not, we used 2-D arrays for each map to store the map information in small grids. The 2-D array stores integer numbers which indicates obstacle, path or lead to an exit for switching to other map. -1 stands for obstacle, 0 stands for path and other number stands for map switch. For example, 2 stands for a path to second map.

Example map matrix:

```
static const int map4[26][34] = {
 /* 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 */
  { 3,  3,  3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 3,  3,  3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},/* 5 */
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  { 0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},/* 10 */
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},/* 15 */
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},/* 20 */
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}
};
```

4.3 Character Movement and Control

Each of our map are represented by a matrix with 26 rows and 34 columns. Reachable areas are marked by 0 and unreachables by -1.

For areas where scene change happens, we marked them by the map number to be displayed. Everytime we the character is moved, we will query the destination coordinate in the matrix and act accordingly. The character, Eric's sprites is stored in the Sockit board ram. There are for all 12 sprite included Eric's four direction movements and each direction has three frames of behavior.

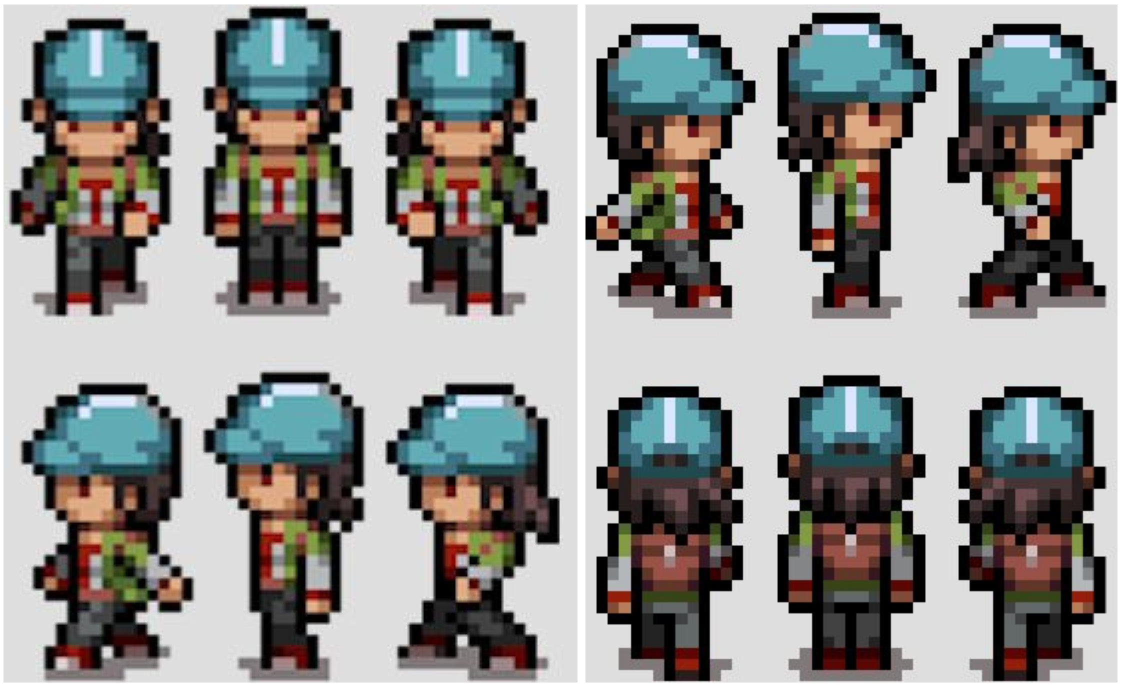


Figure 9. Eric's 4-direction movement sprites

The general movement function will check the current map data 2-D array if the location is a path or not. If there is a path, the software will decided which direction Eric is going to move and what facing state Eric will be to behavior this movement. Then the general movement function will use row and column number to calculate what exactly pixel location and what sprite to print on the screen.

4.4 Campaign Mode and General Movement Mode

Campaign and General Movement functions provide a possible solution for the game to switch between main story line and "open world" style gaming. The users can either choice to follow the game plot or exploring the game by themselves.

a) Campaign Mode

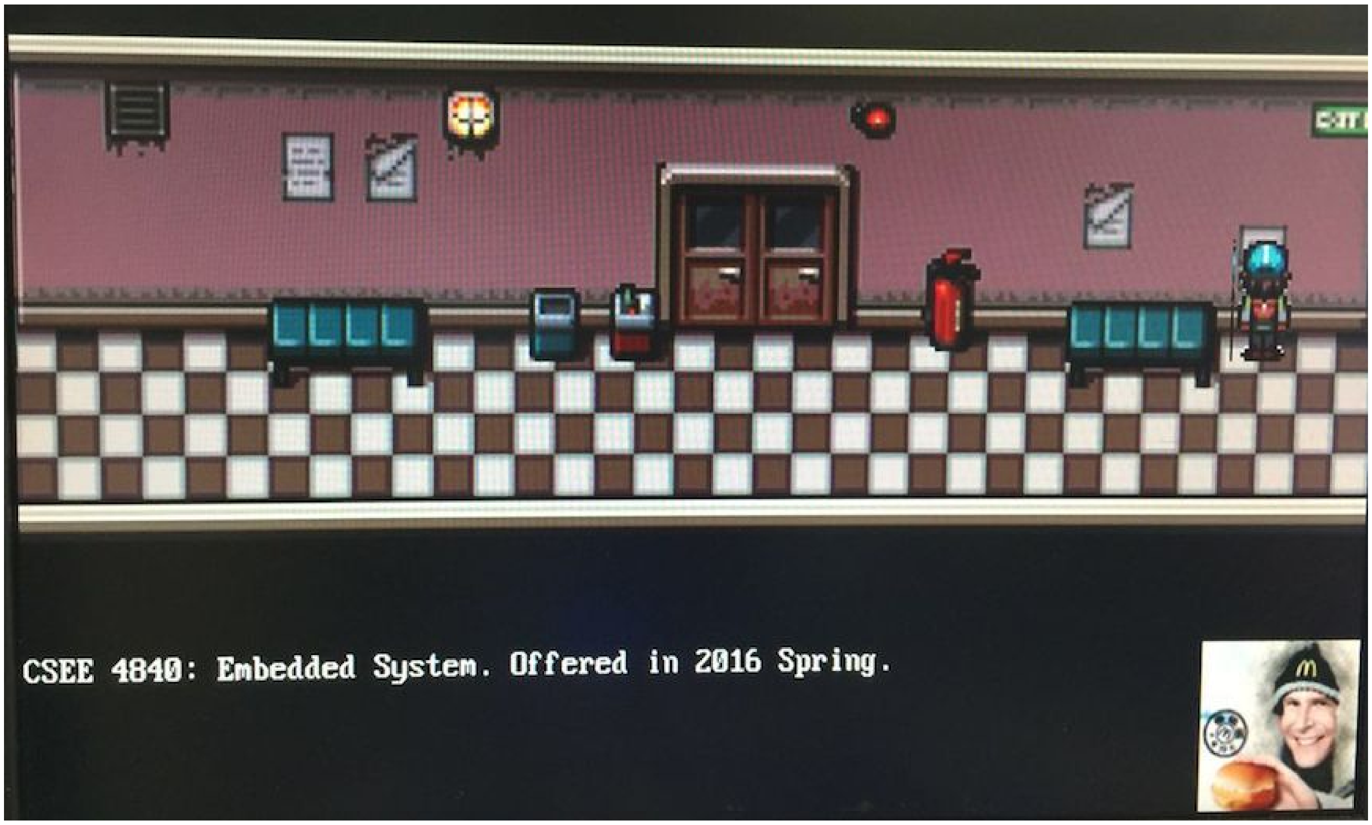
Campaign mode is designed to go through the main story line. It begins with a stranger responding to Eric's newly written console, and a sudden power-off with unknown reason. Driven by curiosity and terror, Eric continued with his journey and uncovered a history that has never come to light. The algorithm is simple. All the dialogs data and and event trigger will stored in the campaign function. For every stage, the user will be promote to trigger certain event in order to push the story line.

b) General Movement Mode

General Movement keep checking if the keyboard arrow keys have been pressed or not. If a arrow key is pressed, it will check the the position the user is intend to going is a path, obstacle or map switching indicator. If there is an obstacle on the player's way. The function will prevent user to move on to the obstacle. If there is an map switch indicator, Eric's sprite will disappear seconds then the new map will be print. After successful print out the map, Eric will appear on the screen with preset location depend on the current map number and last map number.

4.5 Environment Map Mask

In order to make this game reflects more reality, we refined some details of the map. So when Eric walks on certain tile, he triggers some warning or notes. For example here is a poster in the hallway:



CSEE 4840: Embedded System. Offered in 2016 Spring.

Figure 10. Game scene, Hallway decoration.

With such realistic design, it is easier for user to feel the experience of Eric, a lost hard working student encountering the oddest occurrence in the world.

5. Audio Module

The Sockit Board we are using here Cyclone V provides high quality 24 bit audio via SSM2603 Audio CODEC chip. SSM2603 uses I2C protocol for configuration. .It supports three ports: Mic-in, Line-in(audio in), Line-out(audio out) and with various sampling rate from 8Khz to 96kHz.

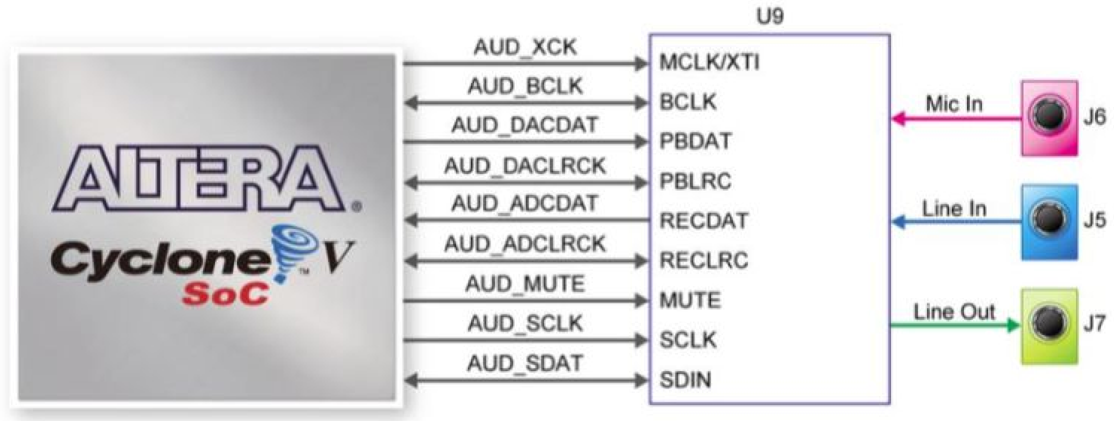
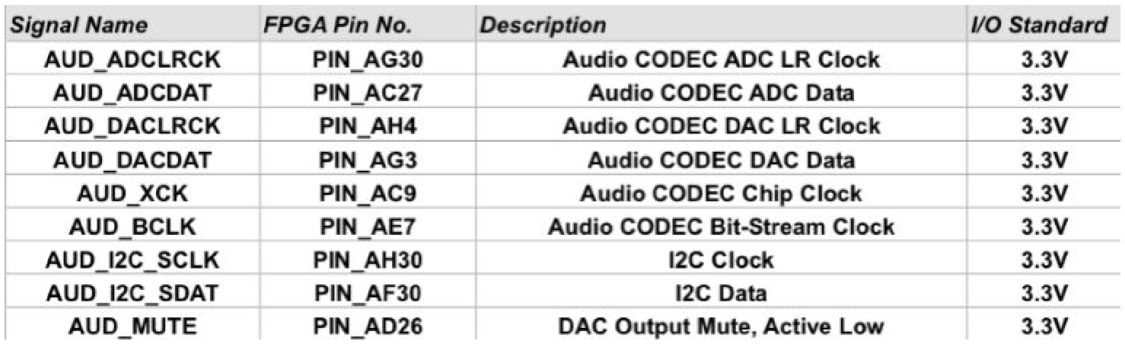


Figure 11. Connection between Cyclone V Sockit board and SSM2603 Audio CODEC



Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
AUD_ADCLRCK	PIN_AG30	Audio CODEC ADC LR Clock	3.3V
AUD_ADCDAT	PIN_AC27	Audio CODEC ADC Data	3.3V
AUD_DACLRCK	PIN_AH4	Audio CODEC DAC LR Clock	3.3V
AUD_DACDAT	PIN_AG3	Audio CODEC DAC Data	3.3V
AUD_XCK	PIN_AC9	Audio CODEC Chip Clock	3.3V
AUD_BCLK	PIN_AE7	Audio CODEC Bit-Stream Clock	3.3V
AUD_I2C_SCLK	PIN_AH30	I2C Clock	3.3V
AUD_I2C_SDAT	PIN_AF30	I2C Data	3.3V
AUD_MUTE	PIN_AD26	DAC Output Mute, Active Low	3.3V

Figure 12. Pin Assignment for SSM2603 Audio CODEC

5.1 Configuring codec

Inter-Integrated Circuit (I2C) protocol, used for audio codec configuration (sampling rate, sampling width, etc.). I2C uses serial data and serial clock. Data is sent 1-bit at a time. Different bits are synchronised and distinguished by the serial clock.

A 24-bit i2c_data is a complete piece of information. Audio codec slave address is 0x34/0x35. Codec takes 16-bit data words: 7-bit register address, 9-bit contents. The formulation of the i2c_data is shown below:

24-bit i2c_data	7 bit slave addr + 1 bit r/w, 2*8=16 bit data, with ack after each 8 bit

24-bit i2c_data	7 bit slave addr + 1 bit r/w + 7 bit register addr + 9 bit contents, with ack after each 8-bit

There are 19 registers in total, and each takes 9-bit contents. The function of the registers are:

R0/1	input volume of L/R channel ADC
R2/3	input volume of L/R channel DAC
R4	analog audio path
R5	digital audio path
R6	necessary power management bits
R7	digital audio interface. Left-justified slave mode: Format→ 01. 16-bit audio samples:WL→ 00.
R8	sampling rate. 11.025KHz for DAC. 44.1kHZ for ADC.

5.2 Sending samples to DAC

Digital audio interface sends samples to DAC and receive samples from ADC. DAC operates in left-justified slave mode: FPGA drives all of the clocks. DAC receives 16 bit samples.

MCLK - 11.28MHz: master clock generated by Phase-Locked Loops.
LRclk - 11kHz (1/0 → L/R channel).
BCLK - ¼*MCLK.

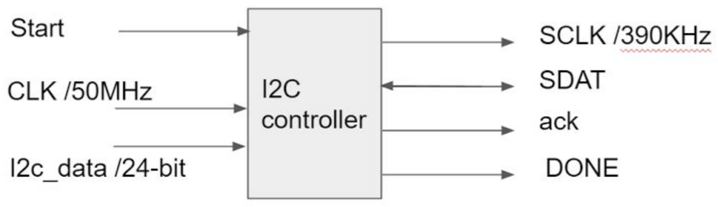
5.3 Music control and play

In total we have 2 pieces of music stored in the ROM. The music starts to play at the negative edge of beginning signal. Former music stops immediately whenever a new negedge occurs. Each piece of music plays for only once as per each negedge.

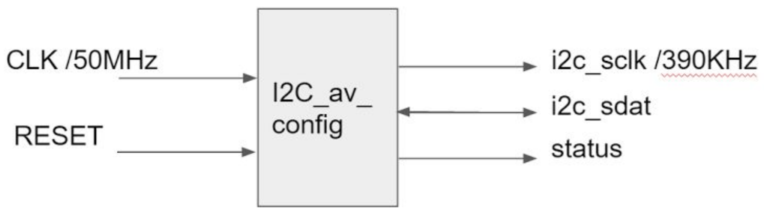
5.4 Module

(*Original code comes from Howard Mao's blog http://zhehaomao.com/blog/fpga/2014/01/15/sockit-8.html)

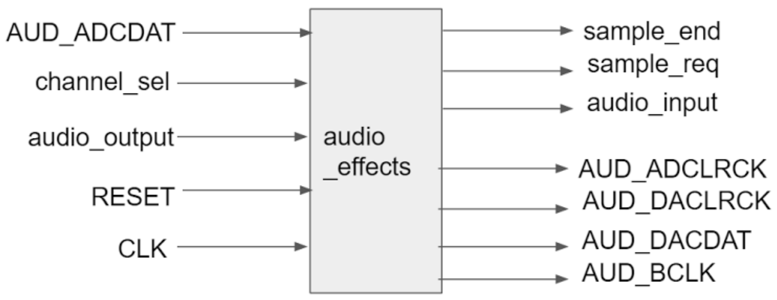
I2C controller: Sending 24-bit i2c_data according to I2C protocol.



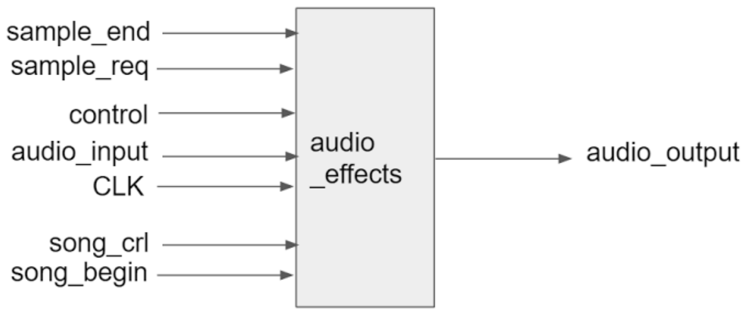
I2C_av_config: Determining what the 24-bit i2c_data should be, according to the desired feature and functionality. And sending data out based on I2C protocol calling the module I2C controller.



Digital audio interface: sending samples to DAC and receiving samples from ADC. (left-justified slave mode: FPGA drives all of the clocks.)



Audio_effects: Reading samples from ROM. Which piece of music is played is based on the control signals song_crl and song_begin. The music starts to play at the negative edge of beginning signal. Former music stops immediately whenever a new negedge occurs. Each piece of music plays for only once as per each negedge.



5.5 Preparing for .mif file

Several .m4a audio files are converted to .mif files in matlab. We downsampled the samples to 11kHz.

6. Conclusion

It has been such a pleasant journey to develop this game of Eric's bizarre adventure from scratch to such an amazing fully developed product that is ready to launch into market. We have met all the milestones that we expected, and implemented all the functions we wanted to explore about FPGA Sockit board. More importantly we learned lots of practical techniques in the process of designing embedded systems and physically turning our pure imagination into real prototypes. We also learned that the actual developing process is full of ups and downs, full of good and bad surprises. It has been really different from theories, sometimes we need to spend hours debugging a single bug, we need to maintain flexibility to make changes in our design, and more importantly, we have learned to constantly brainstorm for possible improvements. This has been a real amazing learning experience for us and we are all honored to have taken this class.

7. Acknowledgement

We give great thanks to Prof. Stephen A. Edwards for helping us with memory issues, printing characters, and constantly giving suggestions for improvements of our design.

We also give great thanks to Adam Incera, our TA, who helped us correctly set up Linux environment on Sockit board, and encouraging us to catch up with our milestones and expectations.

Appendix

A. Hardware

1. VGA_LED.sv

```
/*
 * Avalon memory-mapped peripheral for the VGA LED Emulator
 *
 * Stephen A. Edwards
 * Columbia University
 */
```

```
module VGA_LED(input logic            clk,
          input logic       reset,
          input logic [7:0]  writedata,
          input logic       write,
          input                  chipselect,
          input logic [4:0]  address,

          output logic [7:0] VGA_R, VGA_G, VGA_B,

          output logic       VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n,
           output logic [7:0]       VGA_MUSIC_CRL,
          output logic       VGA_SYNC_n);

           logic [23:0] write_line_num;
           logic [11:0] color;
           logic [15:0] x_in, y_in, xv_in, yv_in;
           logic [7:0] map_crl, char_crl,filt_crl;

   VGA_LED_Emulator led_emulator(.clk50(clk), .reset(reset), .VGA_R(VGA_R),
.color(color), .write_line_num(write_line_num), .map_crl(map_crl), .x_in(x_in),
.y_in(y_in), .xv_in(xv_in), .yv_in(yv_in), .char_crl(char_crl), .filt_crl(filt_crl),

.VGA_G(VGA_G), .VGA_B(VGA_B), .VGA_CLK(VGA_CLK), .VGA_HS(VGA_HS),
.VGA_VS(VGA_VS),

.VGA_BLANK_n(VGA_BLANK_n), .VGA_SYNC_n(VGA_SYNC_n));
  always_ff @(posedge clk)
          if (reset) begin
   color[11:0] <= 12'hFFF;
   write_line_num[7:0] <= 8'h50;
   write_line_num[15:8] <= 8'hFF;
   write_line_num[23:16] <= 8'h02;
   map_crl[7:0] <= 8'h01;
   char_crl[7:0] <= 8'b10000011;
   x_in <= 16'd1;
   y_in <= 16'd55;
   xv_in <= 16'd100;
   yv_in <= 16'd100;

   VGA_MUSIC_CRL <= 8'd1;


          end else if (chipselect && write)
          case (address)
   5'h0 : color[11:8] <= writedata[3:0];
   5'h1 : color[7:4] <= writedata[3:0];
```

```
5'h2 : color[3:0] <= writedata[3:0];
5'h3 : write_line_num[23:16] <= writedata;
5'h4 : write_line_num[15:8] <= writedata;
5'h5 : write_line_num[7:0] <= writedata;
5'h6 : x_in[15:8] <= writedata;
5'h7 : x_in[7:0] <= writedata;
5'h8 : y_in[15:8] <= writedata;
5'h9 : y_in[7:0] <= writedata;
5'd10 : char_crl[7:0] <= writedata;
5'd11 : map_crl[7:0] <= writedata;
5'd12 : filt_crl[7:0] <= writedata;
5'd13 : VGA_MUSIC_CRL[7:0] <= writedata;
5'd14 : xv_in[15:8] <= writedata;
5'd15 : xv_in[7:0] <= writedata;
5'd16 : yv_in[15:8] <= writedata;
5'd17 : yv_in[7:0] <= writedata;

        endcase

endmodule
```

2. VGA_LED_Emulator.sv

```
/*
 * Seven-segment LED emulator
 *
 * Stephen A. Edwards, Columbia University
 */

module VGA_LED_Emulator(
 input logic       clk50, reset,
 input logic [7:0]   map_crl, char_crl, filt_crl,
 input logic [23:0]  write_line_num,
 input logic [11:0]  color,
 input logic [15:0]  x_in, y_in,xv_in, yv_in,
 output logic [7:0]  VGA_R, VGA_G, VGA_B,
 output logic               VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n, VGA_SYNC_n);

  parameter HACTIVE          = 11'd 1280,
            HFRONT_PORCH = 11'd 32,
            HSYNC            = 11'd 192,
            HBACK_PORCH  = 11'd 96,
            HTOTAL         = HACTIVE + HFRONT_PORCH + HSYNC + HBACK_PORCH; //
1600
```

```
 // Parameters for vcount
 parameter VACTIVE          = 10'd 480,
         VFRONT_PORCH = 10'd 10,
         VSYNC          = 10'd 2,
         VBACK_PORCH  = 10'd 33,
         VTOTAL         = VACTIVE + VFRONT_PORCH + VSYNC + VBACK_PORCH; //
525

 logic [10:0]                    hcount; // Horizontal counter
                                 // Hcount[10:1] indicates pixel column (0-639)
 logic                           endOfLine;
 logic [3:0]              R,B,G;

 always_ff @(posedge clk50 or posedge reset)
         if (reset)          hcount <= 0;
         else if (endOfLine) hcount <= 0;
         else                hcount <= hcount + 32'd 1;

 assign endOfLine = hcount == HTOTAL - 1;

 logic [9:0]                     vcount;
 logic                           endOfField;

 always_ff @(posedge clk50 or posedge reset)
         if (reset)          vcount <= 0;
         else if (endOfLine)
         if (endOfField)   vcount <= 0;
         else                vcount <= vcount + 32'd 1;

 assign endOfField = vcount == VTOTAL - 1;

 assign VGA_HS = !( (hcount[10:8] == 3'b101) & !(hcount[7:5] == 3'b111));
 assign VGA_VS = !( vcount[9:1] == (VACTIVE + VFRONT_PORCH) / 2);

 assign VGA_SYNC_n = 1; // For adding sync to video signals; not used for VGA

 assign VGA_BLANK_n = !( hcount[10] & (hcount[9] | hcount[8]) ) &
                 !( vcount[9] | (vcount[8:5] == 4'b1111) );

 assign VGA_CLK = hcount[0]; // 25 MHz clock: pixel latched on rising edge


  logic [23:0] read_line_num;
  logic [11:0] color_out;
  logic we;
```

```
   logic flash;
   logic nuk;


   logic [10:0] adr_eric;
   logic [11:0] data_ericf1;
   logic [11:0] data_ericb1;
   logic [11:0] data_ericr1;
   logic [11:0] data_ericl1;

   logic [11:0] data_ericf2;
   logic [11:0] data_ericb2;
   logic [11:0] data_ericr2;
   logic [11:0] data_ericl2;

   logic [11:0] data_ericf3;
   logic [11:0] data_ericb3;
   logic [11:0] data_ericr3;
   logic [11:0] data_ericl3;


   logic ericf1_on;
   logic ericb1_on;
   logic ericl1_on;
   logic ericr1_on;

   logic ericf2_on;
   logic ericb2_on;
   logic ericl2_on;
   logic ericr2_on;

   logic ericf3_on;
   logic ericb3_on;
   logic ericl3_on;
   logic ericr3_on;
   logic ericv_on;

   pre_mem_process pre_mem_process(.hc(hc), .vc(vc),
.read_line_num(read_line_num), .nuk(nuk), .map_crl(map_crl), .filt_crl(filt_crl),
.we(we), .flash(flash));
   memory memory(.clk50(clk50), .write_line_num(write_line_num), .we(we),
.read_line_num(read_line_num),.color(color), .color_out(color_out));
   ericf1   ericf1 (.clock(clk50), .address(adr_eric), .q(data_ericf1));
   ericf2   ericf2 (.clock(clk50), .address(adr_eric), .q(data_ericf2));
   ericf3   ericf3 (.clock(clk50), .address(adr_eric), .q(data_ericf3));
```

```
    ericb1   ericb1 (.clock(clk50), .address(adr_eric), .q(data_ericb1));
    ericb2   ericb2 (.clock(clk50), .address(adr_eric), .q(data_ericb2));
    ericb3   ericb3 (.clock(clk50), .address(adr_eric), .q(data_ericb3));


    ericl1   ericl1 (.clock(clk50), .address(adr_eric), .q(data_ericl1));
    ericl2   ericl2 (.clock(clk50), .address(adr_eric), .q(data_ericl2));
    ericl3   ericl3 (.clock(clk50), .address(adr_eric), .q(data_ericl3));


    ericr1   ericr1 (.clock(clk50), .address(adr_eric), .q(data_ericr1));
    ericr2   ericr2 (.clock(clk50), .address(adr_eric), .q(data_ericr2));
    ericr3   ericr3 (.clock(clk50), .address(adr_eric), .q(data_ericr3));

    integer hc, vc;

  always_latch
    begin
    hc = hcount[10:1];
    vc = vcount;
    if( (hc >= (x_in)) && (hc < (16'd30 + x_in)) && (vc >= (y_in-16'd54) )&& (vc <= y_in))
                    begin
                    if (char_crl[3:0] == 4'b0001)
                            begin
                            ericf1_on = 1;
                            end
                    else if (char_crl[3:0] == 4'b0010)
                            begin
                            ericf2_on = 1;
                            end
                    else if (char_crl[3:0] == 4'b0011)
                            begin
                            ericf3_on = 1;
                            end
                    else if (char_crl[3:0] == 4'b0100)
                            begin
                            ericb1_on = 1;
                            end
                    else if (char_crl[3:0] == 4'b0101)
                            begin
                            ericb2_on = 1;
                            end
                    else if (char_crl[3:0] == 4'b0110)
                            begin
                            ericb3_on = 1;
				end
			else if (char_crl[3:0] == 4'b0111)
				begin
				ericl1_on = 1;
				end
			else if (char_crl[3:0] == 4'b1000)
				begin
				ericl2_on = 1;
				end
			else if (char_crl[3:0] == 4'b1001)
				begin
				ericl3_on = 1;
				end
			else if (char_crl[3:0] == 4'b1010)
				begin
				ericr1_on = 1;
				end
			else if (char_crl[3:0] == 4'b1011)
				begin
				ericr2_on = 1;
				end
			else if (char_crl[3:0] == 4'b1100)
				begin
				ericr3_on = 1;
				end
			else
				begin
				ericf2_on = 1;
				end

			adr_eric <= ((hc-x_in) + (vc-(y_in-54))*30);
			end
		else
			begin
			ericf1_on = 0;
			ericb1_on = 0;
			ericl1_on = 0;
			ericr1_on = 0;

			ericf2_on = 0;
			ericb2_on = 0;
			ericl2_on = 0;
			ericr2_on = 0;

			ericf3_on = 0;
			ericb3_on = 0;
```

```
                ericl3_on = 0;
                ericr3_on = 0;
                end


  if( (hc >= (xv_in)) && (hc < (16'd30 + xv_in)) && (vc >= (yv_in-16'd54) )&& (vc <=
yv_in) && char_crl[7])
                begin
                        ericv_on = 1;
                        adr_eric <= ((hc-xv_in) + (vc-(yv_in-54))*30);
                end
  else
                begin
                        ericv_on = 0;
                end


  if( data_ericf1[11:0] == 12'd4095 || data_ericf1[11:0] == 12'd3822 || data_ericf1[11:0]
== 12'd2730 | data_ericf1[11:0] == 12'd4078|| data_ericf1[11:0] == 12'd3276 ||
data_ericf1[11:0] == 12'd3003)
                begin
                        ericf1_on = 0;
                end
  if( data_ericb1[11:0] == 12'd4095 || data_ericb1[11:0] == 12'd3822 || data_ericb1[11:0]
== 12'd2730 || data_ericb1[11:0] == 12'd4078|| data_ericb1[11:0] == 12'd3276 ||
data_ericb1[11:0] == 12'd3003)
                begin
                        ericb1_on = 0;
                end
  if( data_ericl1[11:0] == 12'd4095 || data_ericl1[11:0] == 12'd3822 || data_ericl1[11:0]
== 12'd2730 || data_ericl1[11:0] == 12'd4078|| data_ericl1[11:0] == 12'd3276 ||
data_ericl1[11:0] == 12'd3003)
                begin
                        ericl1_on = 0;
                end
  if( data_ericr1[11:0] == 12'd4095 || data_ericr1[11:0] == 12'd3822 || data_ericr1[11:0]
== 12'd2730 || data_ericr1[11:0] == 12'd4078|| data_ericr1[11:0] == 12'd3276 ||
data_ericr1[11:0] == 12'd3003)
                begin
                        ericr1_on = 0;
                end
```

```
   if( data_ericf2[11:0] == 12'd4095 || data_ericf2[11:0] == 12'd3822 || data_ericf2[11:0]
== 12'd2730 || data_ericf2[11:0] == 12'd4078 || data_ericf2[11:0] == 12'd3276 ||
data_ericf2[11:0] == 12'd3003)
                    begin
                              ericf2_on = 0;
                    end
   if( data_ericb2[11:0] == 12'd4095 || data_ericb2[11:0] == 12'd3822 || data_ericb2[11:0]
== 12'd2730 || data_ericb2[11:0] == 12'd4078 || data_ericb2[11:0] == 12'd3276 ||
data_ericb2[11:0] == 12'd3003)
                    begin
                              ericb2_on = 0;
                    end
   if( data_ericl2[11:0] == 12'd4095 || data_ericl2[11:0] == 12'd3822 || data_ericl2[11:0]
== 12'd2730 || data_ericl2[11:0] == 12'd4078 || data_ericl2[11:0] == 12'd3276 ||
data_ericl2[11:0] == 12'd3003)
                    begin
                              ericl2_on = 0;
                    end
   if( data_ericr2[11:0] == 12'd4095 || data_ericr2[11:0] == 12'd3822 || data_ericr2[11:0]
== 12'd2730 || data_ericr2[11:0] == 12'd4078 || data_ericr2[11:0] == 12'd3276 ||
data_ericr2[11:0] == 12'd3003)
                    begin
                              ericr2_on = 0;
                    end



   if( data_ericf3[11:0] == 12'd4095 || data_ericf3[11:0] == 12'd3822 || data_ericf3[11:0]
== 12'd2730 || data_ericf3[11:0] == 12'd4078 || data_ericf3[11:0] == 12'd3276 ||
data_ericf3[11:0] == 12'd3003)
                    begin
                              ericf3_on = 0;
                    end
   if( data_ericb3[11:0] == 12'd4095 || data_ericb3[11:0] == 12'd3822 || data_ericb3[11:0]
== 12'd2730 || data_ericb3[11:0] == 12'd4078 || data_ericb3[11:0] == 12'd3276 ||
data_ericb3[11:0] == 12'd3003)
                    begin
                              ericb3_on = 0;
                    end
   if( data_ericl3[11:0] == 12'd4095 || data_ericl3[11:0] == 12'd3822 || data_ericl3[11:0]
== 12'd2730 || data_ericl3[11:0] == 12'd4078 || data_ericl3[11:0] == 12'd3276 ||
data_ericl3[11:0] == 12'd3003)
                    begin
                              ericl3_on = 0;
                    end
   if( data_ericr3[11:0] == 12'd4095 || data_ericr3[11:0] == 12'd3822 || data_ericr3[11:0]
== 12'd2730 || data_ericr3[11:0] == 12'd4078 || data_ericr3[11:0] == 12'd3276 ||
data_ericr3[11:0] == 12'd3003)
                    begin
                              ericr3_on = 0;
                    end



   if(ericf1_on)
   begin
                    VGA_R = {data_ericf1[11:8],data_ericf1[11:8]};
                    VGA_G = {data_ericf1[7:4],data_ericf1[7:4]};
                    VGA_B = {data_ericf1[3:0],data_ericf1[3:0]};
   end
   else if(ericf2_on)
   begin
                    VGA_R = {data_ericf2[11:8],data_ericf2[11:8]};
                    VGA_G = {data_ericf2[7:4],data_ericf2[7:4]};
                    VGA_B = {data_ericf2[3:0],data_ericf2[3:0]};
   end
   else if(ericf3_on)
   begin
                    VGA_R = {data_ericf3[11:8],data_ericf3[11:8]};
                    VGA_G = {data_ericf3[7:4],data_ericf3[7:4]};
                    VGA_B = {data_ericf3[3:0],data_ericf3[3:0]};
   end
   else if(ericb1_on)
   begin
                    VGA_R = {data_ericb1[11:8],data_ericb1[11:8]};
                    VGA_G = {data_ericb1[7:4],data_ericb1[7:4]};
                    VGA_B = {data_ericb1[3:0],data_ericb1[3:0]};
   end
   else if(ericb2_on)
   begin
                    VGA_R = {data_ericb2[11:8],data_ericb2[11:8]};
                    VGA_G = {data_ericb2[7:4],data_ericb2[7:4]};
                    VGA_B = {data_ericb2[3:0],data_ericb2[3:0]};
   end
   else if(ericb3_on)
   begin
                    VGA_R = {data_ericb3[11:8],data_ericb3[11:8]};
                    VGA_G = {data_ericb3[7:4],data_ericb3[7:4]};
                    VGA_B = {data_ericb3[3:0],data_ericb3[3:0]};
   end
   else if(ericl1_on)
   begin
```

```
                VGA_R = {data_ericl1[11:8],data_ericl1[11:8]};
                VGA_G = {data_ericl1[7:4],data_ericl1[7:4]};
                VGA_B = {data_ericl1[3:0],data_ericl1[3:0]};
end
else if(ericl2_on)
begin
                VGA_R = {data_ericl2[11:8],data_ericl2[11:8]};
                VGA_G = {data_ericl2[7:4],data_ericl2[7:4]};
                VGA_B = {data_ericl2[3:0],data_ericl2[3:0]};
end
else if(ericl3_on)
begin
                VGA_R = {data_ericl3[11:8],data_ericl3[11:8]};
                VGA_G = {data_ericl3[7:4],data_ericl3[7:4]};
                VGA_B = {data_ericl3[3:0],data_ericl3[3:0]};
end
else if(ericr1_on)
begin
                VGA_R = {data_ericr1[11:8],data_ericr1[11:8]};
                VGA_G = {data_ericr1[7:4],data_ericr1[7:4]};
                VGA_B = {data_ericr1[3:0],data_ericr1[3:0]};
end

else if(ericr2_on)
begin
                VGA_R = {data_ericr2[11:8],data_ericr2[11:8]};
                VGA_G = {data_ericr2[7:4],data_ericr2[7:4]};
                VGA_B = {data_ericr2[3:0],data_ericr2[3:0]};
end
else if(ericr3_on)
begin
                VGA_R = {data_ericr3[11:8],data_ericr3[11:8]};
                VGA_G = {data_ericr3[7:4],data_ericr3[7:4]};
                VGA_B = {data_ericr3[3:0],data_ericr3[3:0]};
end
else if(flash)
begin
        VGA_R = {1'b0, color_out[11:9],1'b0, color_out[11:9]};
        VGA_G = {1'b0, color_out[11:9],1'b0, color_out[11:9]};
        VGA_B = {1'b0, color_out[11:9],1'b0, color_out[11:9]};
end
else if(nuk)
begin
        VGA_R = {2'b0, color_out[11:10],2'b0, color_out[11:10]};
        VGA_G = {color_out[7:4],2'b0, color_out[7:4]};
        VGA_B = {color_out[3:0],2'b0, color_out[3:0]};
```

```
    end

    else
    begin
            VGA_R = {color_out[11:8],color_out[11:8]};
            VGA_G = {color_out[7:4],color_out[7:4]};
            VGA_B = {color_out[3:0],color_out[3:0]};
    end

    if(ericv_on)
    begin
                    VGA_R = {data_ericf2[11:8],data_ericf2[11:8]};
                    VGA_G = {data_ericf2[11:8],data_ericf2[11:8]};
                    VGA_B = {data_ericf2[11:8],data_ericf2[11:8]};
    end
    end
endmodule


module pre_mem_process (

 input logic [32:0] hc,
 input logic [32:0] vc,
 input logic [7:0] map_crl,
 input logic [7:0] filt_crl,
 output logic [23:0] read_line_num,
 output logic we,
 output logic flash,
 output logic nuk);

always_comb
begin
    we = map_crl[0];
    flash = filt_crl[0];
    nuk = filt_crl[1];
    read_line_num = hc+vc*640;
end
endmodule

module memory (
 input logic       clk50,
 input logic [11:0]  color,
 input logic [23:0] write_line_num,
 input logic [23:0] read_line_num,
 input logic we,
 output logic [11:0] color_out);
```

```
logic [11:0] mem [307200:0];

always_ff @(posedge clk50)
begin
   if (we)
   begin
         mem[write_line_num] <= color;
   end
   color_out[11:0] <= mem[read_line_num];
end
endmodule
```

3. SoCkit_top.v

```
//
======================================================================
====
// Copyright (c) 2013 by Terasic Technologies Inc.
//
======================================================================
====
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
```

```
//
========================================================================
====
//
//  Terasic Technologies Inc
//  9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan
//
//
//                  web: http://www.terasic.com/
//                  email: support@terasic.com
//
//
========================================================================
====
//
========================================================================
====
//
// Major Functions:    SoCKit_Default
//
//
========================================================================
====
// Revision History :
//
========================================================================
====
//   Ver  :| Author                  :| Mod. Date :| Changes Made:
//   V1.0 :| xinxian                 :| 04/02/13  :| Initial Revision
//
========================================================================
====

//`define ENABLE_DDR3
//`define ENABLE_HPS
//`define ENABLE_HSMC_XCVR

module SoCKit_Top(

            ///////////AUD////////////
            AUD_ADCDAT,
            AUD_ADCLRCK,
            AUD_BCLK,
            AUD_DACDAT,
            AUD_DACLRCK,
            AUD_I2C_SCLK,
		AUD_I2C_SDAT,
		AUD_MUTE,
		AUD_XCK,

`ifdef ENABLE_DDR3
		/////////DDR3/////////
		DDR3_A,
		DDR3_BA,
		DDR3_CAS_n,
		DDR3_CKE,
		DDR3_CK_n,
		DDR3_CK_p,
		DDR3_CS_n,
		DDR3_DM,
		DDR3_DQ,
		DDR3_DQS_n,
		DDR3_DQS_p,
		DDR3_ODT,
		DDR3_RAS_n,
		DDR3_RESET_n,
		DDR3_RZQ,
		DDR3_WE_n,
`endif /*ENABLE_DDR3*/

		/////////FAN/////////
		FAN_CTRL,

`ifdef ENABLE_HPS
		/////////HPS/////////
		HPS_CLOCK_25,
		HPS_CLOCK_50,
		HPS_CONV_USB_n,
		HPS_DDR3_A,
		HPS_DDR3_BA,
		HPS_DDR3_CAS_n,
		HPS_DDR3_CKE,
		HPS_DDR3_CK_n,
		HPS_DDR3_CK_p,
		HPS_DDR3_CS_n,
		HPS_DDR3_DM,
		HPS_DDR3_DQ,
		HPS_DDR3_DQS_n,
		HPS_DDR3_DQS_p,
		HPS_DDR3_ODT,
		HPS_DDR3_RAS_n,
		HPS_DDR3_RESET_n,
		HPS_DDR3_RZQ,
		HPS_DDR3_WE_n,
		HPS_ENET_GTX_CLK,
		HPS_ENET_INT_n,
		HPS_ENET_MDC,
		HPS_ENET_MDIO,
		HPS_ENET_RESET_n,
		HPS_ENET_RX_CLK,
		HPS_ENET_RX_DATA,
		HPS_ENET_RX_DV,
		HPS_ENET_TX_DATA,
		HPS_ENET_TX_EN,
		HPS_FLASH_DATA,
		HPS_FLASH_DCLK,
		HPS_FLASH_NCSO,
		HPS_GSENSOR_INT,
		HPS_I2C_CLK,
		HPS_I2C_SDA,
		HPS_KEY,
		HPS_LCM_D_C,
		HPS_LCM_RST_N,
		HPS_LCM_SPIM_CLK,
		HPS_LCM_SPIM_MISO,
		HPS_LCM_SPIM_MOSI,
		HPS_LCM_SPIM_SS,
		HPS_LED,
		HPS_LTC_GPIO,
		HPS_RESET_n,
		HPS_SD_CLK,
		HPS_SD_CMD,
		HPS_SD_DATA,
		HPS_SPIM_CLK,
		HPS_SPIM_MISO,
		HPS_SPIM_MOSI,
		HPS_SPIM_SS,
		HPS_SW,
		HPS_UART_RX,
		HPS_UART_TX,
		HPS_USB_CLKOUT,
		HPS_USB_DATA,
		HPS_USB_DIR,
		HPS_USB_NXT,
		HPS_USB_RESET_PHY,
		HPS_USB_STP,
		HPS_WARM_RST_n,
`endif /*ENABLE_HPS*/
```

```
        /////////HSMC/////////
        HSMC_CLKIN_n,
        HSMC_CLKIN_p,
        HSMC_CLKOUT_n,
        HSMC_CLKOUT_p,
        HSMC_CLK_IN0,
        HSMC_CLK_OUT0,
        HSMC_D,

`ifdef ENABLE_HSMC_XCVR

        HSMC_GXB_RX_p,
        HSMC_GXB_TX_p,
        HSMC_REF_CLK_p,
`endif
        HSMC_RX_n,
        HSMC_RX_p,
        HSMC_SCL,
        HSMC_SDA,
        HSMC_TX_n,
        HSMC_TX_p,

        /////////IRDA/////////
        IRDA_RXD,

        /////////KEY/////////
        KEY,

        /////////LED/////////
        LED,

        /////////OSC/////////
        OSC_50_B3B,
        OSC_50_B4A,
        OSC_50_B5B,
        OSC_50_B8A,

        /////////PCIE/////////
        PCIE_PERST_n,
        PCIE_WAKE_n,

        /////////RESET/////////
        RESET_n,

        /////////SI5338/////////
```

```
SI5338_SCL,
SI5338_SDA,

/////////SW/////////
SW,

/////////TEMP/////////
TEMP_CS_n,
TEMP_DIN,
TEMP_DOUT,
TEMP_SCLK,

/////////USB/////////
USB_B2_CLK,
USB_B2_DATA,
USB_EMPTY,
USB_FULL,
USB_OE_n,
USB_RD_n,
USB_RESET_n,
USB_SCL,
USB_SDA,
USB_WR_n,

/////////VGA/////////
VGA_B,
VGA_BLANK_n,
VGA_CLK,
VGA_G,
VGA_HS,
VGA_R,
VGA_SYNC_n,
VGA_VS,
///////////hps//////////
memory_mem_a,
memory_mem_ba,
memory_mem_ck,
memory_mem_ck_n,
memory_mem_cke,
memory_mem_cs_n,
memory_mem_ras_n,
memory_mem_cas_n,
memory_mem_we_n,
memory_mem_reset_n,
memory_mem_dq,
memory_mem_dqs,
memory_mem_dqs_n,
memory_mem_odt,
memory_mem_dm,
memory_oct_rzqin,
hps_io_hps_io_emac1_inst_TX_CLK,
hps_io_hps_io_emac1_inst_TXD0,
hps_io_hps_io_emac1_inst_TXD1,
hps_io_hps_io_emac1_inst_TXD2,
hps_io_hps_io_emac1_inst_TXD3,
hps_io_hps_io_emac1_inst_RXD0,
hps_io_hps_io_emac1_inst_MDIO,
hps_io_hps_io_emac1_inst_MDC,
hps_io_hps_io_emac1_inst_RX_CTL,
hps_io_hps_io_emac1_inst_TX_CTL,
hps_io_hps_io_emac1_inst_RX_CLK,
hps_io_hps_io_emac1_inst_RXD1,
hps_io_hps_io_emac1_inst_RXD2,
hps_io_hps_io_emac1_inst_RXD3,
hps_io_hps_io_qspi_inst_IO0,
hps_io_hps_io_qspi_inst_IO1,
hps_io_hps_io_qspi_inst_IO2,
hps_io_hps_io_qspi_inst_IO3,
hps_io_hps_io_qspi_inst_SS0,
hps_io_hps_io_qspi_inst_CLK,
hps_io_hps_io_sdio_inst_CMD,
hps_io_hps_io_sdio_inst_D0,
hps_io_hps_io_sdio_inst_D1,
hps_io_hps_io_sdio_inst_CLK,
hps_io_hps_io_sdio_inst_D2,
hps_io_hps_io_sdio_inst_D3,
hps_io_hps_io_usb1_inst_D0,
hps_io_hps_io_usb1_inst_D1,
hps_io_hps_io_usb1_inst_D2,
hps_io_hps_io_usb1_inst_D3,
hps_io_hps_io_usb1_inst_D4,
hps_io_hps_io_usb1_inst_D5,
hps_io_hps_io_usb1_inst_D6,
hps_io_hps_io_usb1_inst_D7,
hps_io_hps_io_usb1_inst_CLK,
hps_io_hps_io_usb1_inst_STP,
hps_io_hps_io_usb1_inst_DIR,
hps_io_hps_io_usb1_inst_NXT,
hps_io_hps_io_spim0_inst_CLK,
hps_io_hps_io_spim0_inst_MOSI,
hps_io_hps_io_spim0_inst_MISO,
hps_io_hps_io_spim0_inst_SS0,
            hps_io_hps_io_spim1_inst_CLK,
            hps_io_hps_io_spim1_inst_MOSI,
            hps_io_hps_io_spim1_inst_MISO,
            hps_io_hps_io_spim1_inst_SS0,
            hps_io_hps_io_uart0_inst_RX,
            hps_io_hps_io_uart0_inst_TX,
            hps_io_hps_io_i2c1_inst_SDA,
            hps_io_hps_io_i2c1_inst_SCL,
            hps_io_hps_io_gpio_inst_GPIO00
            );

 //=======================================================
 //  PORT declarations
 //=======================================================

 ///////// AUD /////////
 input                          AUD_ADCDAT;
 inout                          AUD_ADCLRCK;
 inout                          AUD_BCLK;
 output                         AUD_DACDAT;
 inout                          AUD_DACLRCK;
 output                         AUD_I2C_SCLK;
 inout                          AUD_I2C_SDAT;
 output                         AUD_MUTE;
 output                         AUD_XCK;

`ifdef ENABLE_DDR3
 ///////// DDR3 /////////
 output [14:0]                        DDR3_A;
 output [2:0]                         DDR3_BA;
 output                         DDR3_CAS_n;
 output                         DDR3_CKE;
 output                         DDR3_CK_n;
 output                         DDR3_CK_p;
 output                         DDR3_CS_n;
 output [3:0]                         DDR3_DM;
 inout [31:0]                         DDR3_DQ;
 inout [3:0]                          DDR3_DQS_n;
 inout [3:0]                          DDR3_DQS_p;
 output                         DDR3_ODT;
 output                         DDR3_RAS_n;
 output                         DDR3_RESET_n;
 input                          DDR3_RZQ;
 output                         DDR3_WE_n;
`endif /*ENABLE_DDR3*/
```

```
  ///////// FAN /////////
  output                          FAN_CTRL;

`ifdef ENABLE_HPS
  ///////// HPS /////////
  input                           HPS_CLOCK_25;
  input                           HPS_CLOCK_50;
  input                           HPS_CONV_USB_n;
  output [14:0]                         HPS_DDR3_A;
  output [2:0]                          HPS_DDR3_BA;
  output                          HPS_DDR3_CAS_n;
  output                          HPS_DDR3_CKE;
  output                          HPS_DDR3_CK_n;
  output                          HPS_DDR3_CK_p;
  output                          HPS_DDR3_CS_n;
  output [3:0]                          HPS_DDR3_DM;
  inout [31:0]                          HPS_DDR3_DQ;
  inout [3:0]                           HPS_DDR3_DQS_n;
  inout [3:0]                           HPS_DDR3_DQS_p;
  output                          HPS_DDR3_ODT;
  output                          HPS_DDR3_RAS_n;
  output                          HPS_DDR3_RESET_n;
  input                           HPS_DDR3_RZQ;
  output                          HPS_DDR3_WE_n;
  input                           HPS_ENET_GTX_CLK;
  input                           HPS_ENET_INT_n;
  output                          HPS_ENET_MDC;
  inout                           HPS_ENET_MDIO;
  output                          HPS_ENET_RESET_n;
  input                           HPS_ENET_RX_CLK;
  input [3:0]                           HPS_ENET_RX_DATA;
  input                           HPS_ENET_RX_DV;
  output [3:0]                          HPS_ENET_TX_DATA;
  output                          HPS_ENET_TX_EN;
  inout [3:0]                           HPS_FLASH_DATA;
  output                          HPS_FLASH_DCLK;
  output                          HPS_FLASH_NCSO;
  input                           HPS_GSENSOR_INT;
  inout                           HPS_I2C_CLK;
  inout                           HPS_I2C_SDA;
  inout [3:0]                           HPS_KEY;
  output                          HPS_LCM_D_C;
  output                          HPS_LCM_RST_N;
  input                           HPS_LCM_SPIM_CLK;
  inout                           HPS_LCM_SPIM_MISO;
  output                          HPS_LCM_SPIM_MOSI;
```

```
 output                  HPS_LCM_SPIM_SS;
 output [3:0]                 HPS_LED;
 inout                   HPS_LTC_GPIO;
 input                   HPS_RESET_n;
 output                  HPS_SD_CLK;
 inout                   HPS_SD_CMD;
 inout [3:0]                  HPS_SD_DATA;
 output                  HPS_SPIM_CLK;
 input                   HPS_SPIM_MISO;
 output                  HPS_SPIM_MOSI;
 output                  HPS_SPIM_SS;
 input [3:0]                  HPS_SW;
 input                   HPS_UART_RX;
 output                  HPS_UART_TX;
 input                   HPS_USB_CLKOUT;
 inout [7:0]                  HPS_USB_DATA;
 input                   HPS_USB_DIR;
 input                   HPS_USB_NXT;
 output                  HPS_USB_RESET_PHY;
 output                  HPS_USB_STP;
 input                   HPS_WARM_RST_n;
`endif /*ENABLE_HPS*/

 ///////// HSMC /////////
 input [2:1]                  HSMC_CLKIN_n;
 input [2:1]                  HSMC_CLKIN_p;
 output [2:1]                 HSMC_CLKOUT_n;
 output [2:1]                 HSMC_CLKOUT_p;
 input                   HSMC_CLK_IN0;
 output                  HSMC_CLK_OUT0;
 inout [3:0]                  HSMC_D;
`ifdef ENABLE_HSMC_XCVR
 input [7:0]                  HSMC_GXB_RX_p;
 output [7:0]                 HSMC_GXB_TX_p;
 input                   HSMC_REF_CLK_p;
`endif
 inout [16:0]                 HSMC_RX_n;
 inout [16:0]                 HSMC_RX_p;
 output                  HSMC_SCL;
 inout                   HSMC_SDA;
 inout [16:0]                 HSMC_TX_n;
 inout [16:0]                 HSMC_TX_p;

 ///////// IRDA /////////
 input                   IRDA_RXD;
```

```
///////// KEY /////////
input [3:0]                         KEY;

///////// LED /////////
output [3:0]                        LED;

///////// OSC /////////
input                         OSC_50_B3B;
input                         OSC_50_B4A;
input                         OSC_50_B5B;
input                         OSC_50_B8A;

///////// PCIE /////////
input                         PCIE_PERST_n;
input                         PCIE_WAKE_n;

///////// RESET /////////
input                         RESET_n;

///////// SI5338 /////////
inout                         SI5338_SCL;
inout                         SI5338_SDA;

///////// SW /////////
input [3:0]                         SW;

///////// TEMP /////////
output                        TEMP_CS_n;
output                        TEMP_DIN;
input                         TEMP_DOUT;
output                        TEMP_SCLK;

///////// USB /////////
input                         USB_B2_CLK;
inout [7:0]                         USB_B2_DATA;
output                        USB_EMPTY;
output                        USB_FULL;
input                         USB_OE_n;
input                         USB_RD_n;
input                         USB_RESET_n;
inout                         USB_SCL;
inout                         USB_SDA;
input                         USB_WR_n;

///////// VGA /////////
output [7:0]                        VGA_B;
output                    VGA_BLANK_n;
output                    VGA_CLK;
output [7:0]                   VGA_G;
output                    VGA_HS;
output [7:0]                   VGA_R;
output                    VGA_SYNC_n;
output                    VGA_VS;

/////////hps pin///////
output wire [14:0]             memory_mem_a;
output wire [2:0]              memory_mem_ba;
output wire                         memory_mem_ck;
output wire                         memory_mem_ck_n;
output wire                         memory_mem_cke;
output wire                         memory_mem_cs_n;
output wire                         memory_mem_ras_n;
output wire                         memory_mem_cas_n;
output wire                         memory_mem_we_n;
output wire                         memory_mem_reset_n;
inout  wire [31:0]             memory_mem_dq;
inout  wire [3:0]              memory_mem_dqs;
inout  wire [3:0]              memory_mem_dqs_n;
output wire                         memory_mem_odt;
output wire [3:0]              memory_mem_dm;
input  wire                    memory_oct_rzqin;
output wire                         hps_io_hps_io_emac1_inst_TX_CLK;
output wire                         hps_io_hps_io_emac1_inst_TXD0;
output wire                         hps_io_hps_io_emac1_inst_TXD1;
output wire                         hps_io_hps_io_emac1_inst_TXD2;
output wire                         hps_io_hps_io_emac1_inst_TXD3;
input  wire                    hps_io_hps_io_emac1_inst_RXD0;
inout  wire                    hps_io_hps_io_emac1_inst_MDIO;
output wire                         hps_io_hps_io_emac1_inst_MDC;
input  wire                    hps_io_hps_io_emac1_inst_RX_CTL;
output wire                         hps_io_hps_io_emac1_inst_TX_CTL;
input  wire                    hps_io_hps_io_emac1_inst_RX_CLK;
input  wire                    hps_io_hps_io_emac1_inst_RXD1;
input  wire                    hps_io_hps_io_emac1_inst_RXD2;
input  wire                    hps_io_hps_io_emac1_inst_RXD3;
inout  wire                    hps_io_hps_io_qspi_inst_IO0;
inout  wire                    hps_io_hps_io_qspi_inst_IO1;
inout  wire                    hps_io_hps_io_qspi_inst_IO2;
inout  wire                    hps_io_hps_io_qspi_inst_IO3;
output wire                         hps_io_hps_io_qspi_inst_SS0;
output wire                         hps_io_hps_io_qspi_inst_CLK;
inout  wire                    hps_io_hps_io_sdio_inst_CMD;
inout  wire                              hps_io_hps_io_sdio_inst_D0;
inout  wire                              hps_io_hps_io_sdio_inst_D1;
output wire                                      hps_io_hps_io_sdio_inst_CLK;
inout  wire                              hps_io_hps_io_sdio_inst_D2;
inout  wire                              hps_io_hps_io_sdio_inst_D3;
inout  wire                              hps_io_hps_io_usb1_inst_D0;
inout  wire                              hps_io_hps_io_usb1_inst_D1;
inout  wire                              hps_io_hps_io_usb1_inst_D2;
inout  wire                              hps_io_hps_io_usb1_inst_D3;
inout  wire                              hps_io_hps_io_usb1_inst_D4;
inout  wire                              hps_io_hps_io_usb1_inst_D5;
inout  wire                              hps_io_hps_io_usb1_inst_D6;
inout  wire                              hps_io_hps_io_usb1_inst_D7;
input  wire                              hps_io_hps_io_usb1_inst_CLK;
output wire                                      hps_io_hps_io_usb1_inst_STP;
input  wire                              hps_io_hps_io_usb1_inst_DIR;
input  wire                              hps_io_hps_io_usb1_inst_NXT;
output wire                                      hps_io_hps_io_spim0_inst_CLK;
output wire                                      hps_io_hps_io_spim0_inst_MOSI;
input  wire                              hps_io_hps_io_spim0_inst_MISO;
output wire                                      hps_io_hps_io_spim0_inst_SS0;
output wire                                      hps_io_hps_io_spim1_inst_CLK;
output wire                                      hps_io_hps_io_spim1_inst_MOSI;
input  wire                              hps_io_hps_io_spim1_inst_MISO;
output wire                                      hps_io_hps_io_spim1_inst_SS0;
input  wire                              hps_io_hps_io_uart0_inst_RX;
output wire                                      hps_io_hps_io_uart0_inst_TX;
inout  wire                              hps_io_hps_io_i2c1_inst_SDA;
inout  wire                              hps_io_hps_io_i2c1_inst_SCL;
inout  wire                              hps_io_hps_io_gpio_inst_GPIO00;
//=======================================================
//  REG/WIRE declarations
//=======================================================

//    For Audio CODEC
wire                                     AUD_CTRL_CLK;    //    For Audio Controller

reg [31:0]                               Cont;
wire                                     VGA_CTRL_CLK;
wire [9:0]                               mVGA_R;
wire [9:0]                               mVGA_G;
wire [9:0]                               mVGA_B;
wire [19:0]                              mVGA_ADDR;
wire                                     DLY_RST;

//    For VGA Controller
```

```
  wire                              mVGA_CLK;
  wire [9:0]                        mRed;
  wire [9:0]                        mGreen;
  wire [9:0]                        mBlue;
  wire                              VGA_Read;   //   VGA data request

  wire [9:0]                        recon_VGA_R;
  wire [9:0]                        recon_VGA_G;
  wire [9:0]                        recon_VGA_B;

  //   For Down Sample
  wire [3:0]                        Remain;
  wire [9:0]                        Quotient;

  wire                              AUD_MUTE;
   wire [7:0]                              VGA_MUSIC_CRL;

  // Drive the LEDs with the switches
  //assign LED = SW;

  // Make the FPGA reset cause an HPS reset
  reg [19:0]                        hps_reset_counter = 20'h0;
  reg                               hps_fpga_reset_n = 0;

  always @(posedge OSC_50_B4A) begin
        if (hps_reset_counter == 20'h ffffff) hps_fpga_reset_n <= 1;
        hps_reset_counter <= hps_reset_counter + 1;
  end

  lab3 u0 (
        .clk_clk                   (OSC_50_B4A),              //              clk.clk
        .reset_reset_n             (hps_fpga_reset_n),        //              reset.reset_n
        .memory_mem_a                     (memory_mem_a),             //
memory.mem_a
        .memory_mem_ba                    (memory_mem_ba),            //              .mem_ba
        .memory_mem_ck                    (memory_mem_ck),            //              .mem_ck
        .memory_mem_ck_n                  (memory_mem_ck_n),              //
.mem_ck_n
        .memory_mem_cke                   (memory_mem_cke),           //              .mem_cke
        .memory_mem_cs_n                  (memory_mem_cs_n),              //
.mem_cs_n
        .memory_mem_ras_n                 (memory_mem_ras_n),             //
.mem_ras_n
        .memory_mem_cas_n                 (memory_mem_cas_n),             //
.mem_cas_n
```

```
        .memory_mem_we_n                 (memory_mem_we_n),              //
.mem_we_n
        .memory_mem_reset_n              (memory_mem_reset_n),           //
.mem_reset_n
        .memory_mem_dq                   (memory_mem_dq),        //              .mem_dq
        .memory_mem_dqs                  (memory_mem_dqs),               //
.mem_dqs
        .memory_mem_dqs_n                (memory_mem_dqs_n),             //
.mem_dqs_n
        .memory_mem_odt                  (memory_mem_odt),       //              .mem_odt
        .memory_mem_dm                   (memory_mem_dm),                //
.mem_dm
        .memory_oct_rzqin                (memory_oct_rzqin),     //              .oct_rzqin
        .hps_io_hps_io_emac1_inst_TX_CLK         (hps_io_hps_io_emac1_inst_TX_CLK), //
                        .hps_0_hps_io.hps_io_emac1_inst_TX_CLK
        .hps_io_hps_io_emac1_inst_TXD0                   (hps_io_hps_io_emac1_inst_TXD0),   //
        .hps_io_emac1_inst_TXD0
        .hps_io_hps_io_emac1_inst_TXD1           (hps_io_hps_io_emac1_inst_TXD1),   //
.hps_io_emac1_inst_TXD1
        .hps_io_hps_io_emac1_inst_TXD2                   (hps_io_hps_io_emac1_inst_TXD2),   //
        .hps_io_emac1_inst_TXD2
        .hps_io_hps_io_emac1_inst_TXD3                   (hps_io_hps_io_emac1_inst_TXD3),   //
        .hps_io_emac1_inst_TXD3
        .hps_io_hps_io_emac1_inst_RXD0                   (hps_io_hps_io_emac1_inst_RXD0),   //
        .hps_io_emac1_inst_RXD0
        .hps_io_hps_io_emac1_inst_MDIO                   (hps_io_hps_io_emac1_inst_MDIO),   //
        .hps_io_emac1_inst_MDIO
        .hps_io_hps_io_emac1_inst_MDC                    (hps_io_hps_io_emac1_inst_MDC),
//              .hps_io_emac1_inst_MDC
        .hps_io_hps_io_emac1_inst_RX_CTL         (hps_io_hps_io_emac1_inst_RX_CTL), //
.hps_io_emac1_inst_RX_CTL
        .hps_io_hps_io_emac1_inst_TX_CTL         (hps_io_hps_io_emac1_inst_TX_CTL), //
.hps_io_emac1_inst_TX_CTL
        .hps_io_hps_io_emac1_inst_RX_CLK         (hps_io_hps_io_emac1_inst_RX_CLK), //
.hps_io_emac1_inst_RX_CLK
        .hps_io_hps_io_emac1_inst_RXD1                   (hps_io_hps_io_emac1_inst_RXD1),   //
        .hps_io_emac1_inst_RXD1
        .hps_io_hps_io_emac1_inst_RXD2                   (hps_io_hps_io_emac1_inst_RXD2),   //
        .hps_io_emac1_inst_RXD2
        .hps_io_hps_io_emac1_inst_RXD3                   (hps_io_hps_io_emac1_inst_RXD3),   //
        .hps_io_emac1_inst_RXD3
        .hps_io_hps_io_qspi_inst_IO0                     (hps_io_hps_io_qspi_inst_IO0),   //
        .hps_io_qspi_inst_IO0
        .hps_io_hps_io_qspi_inst_IO1                     (hps_io_hps_io_qspi_inst_IO1),   //
        .hps_io_qspi_inst_IO1
```

```
          .hps_io_hps_io_qspi_inst_IO2                    (hps_io_hps_io_qspi_inst_IO2),   //
          .hps_io_qspi_inst_IO2
          .hps_io_hps_io_qspi_inst_IO3                    (hps_io_hps_io_qspi_inst_IO3),   //
          .hps_io_qspi_inst_IO3
          .hps_io_hps_io_qspi_inst_SS0                    (hps_io_hps_io_qspi_inst_SS0),   //
          .hps_io_qspi_inst_SS0
          .hps_io_hps_io_qspi_inst_CLK                    (hps_io_hps_io_qspi_inst_CLK),  //
          .hps_io_qspi_inst_CLK
          .hps_io_hps_io_sdio_inst_CMD                    (hps_io_hps_io_sdio_inst_CMD),
//                  .hps_io_sdio_inst_CMD
          .hps_io_hps_io_sdio_inst_D0                     (hps_io_hps_io_sdio_inst_D0),    //
          .hps_io_sdio_inst_D0
          .hps_io_hps_io_sdio_inst_D1                     (hps_io_hps_io_sdio_inst_D1),    //
          .hps_io_sdio_inst_D1
          .hps_io_hps_io_sdio_inst_CLK            (hps_io_hps_io_sdio_inst_CLK),  //
.hps_io_sdio_inst_CLK
          .hps_io_hps_io_sdio_inst_D2             (hps_io_hps_io_sdio_inst_D2),    //
.hps_io_sdio_inst_D2
          .hps_io_hps_io_sdio_inst_D3                     (hps_io_hps_io_sdio_inst_D3),    //
          .hps_io_sdio_inst_D3
          .hps_io_hps_io_usb1_inst_D0                     (hps_io_hps_io_usb1_inst_D0),    //
          .hps_io_usb1_inst_D0
          .hps_io_hps_io_usb1_inst_D1                     (hps_io_hps_io_usb1_inst_D1),    //
          .hps_io_usb1_inst_D1
          .hps_io_hps_io_usb1_inst_D2                     (hps_io_hps_io_usb1_inst_D2),    //
          .hps_io_usb1_inst_D2
          .hps_io_hps_io_usb1_inst_D3                     (hps_io_hps_io_usb1_inst_D3),    //
          .hps_io_usb1_inst_D3
          .hps_io_hps_io_usb1_inst_D4                     (hps_io_hps_io_usb1_inst_D4),    //
          .hps_io_usb1_inst_D4
          .hps_io_hps_io_usb1_inst_D5                     (hps_io_hps_io_usb1_inst_D5),    //
          .hps_io_usb1_inst_D5
          .hps_io_hps_io_usb1_inst_D6                     (hps_io_hps_io_usb1_inst_D6),    //
          .hps_io_usb1_inst_D6
          .hps_io_hps_io_usb1_inst_D7                     (hps_io_hps_io_usb1_inst_D7),    //
          .hps_io_usb1_inst_D7
          .hps_io_hps_io_usb1_inst_CLK            (hps_io_hps_io_usb1_inst_CLK), //
.hps_io_usb1_inst_CLK
          .hps_io_hps_io_usb1_inst_STP            (hps_io_hps_io_usb1_inst_STP),  //
.hps_io_usb1_inst_STP
          .hps_io_hps_io_usb1_inst_DIR            (hps_io_hps_io_usb1_inst_DIR),  //
.hps_io_usb1_inst_DIR
          .hps_io_hps_io_usb1_inst_NXT            (hps_io_hps_io_usb1_inst_NXT),           //
          .hps_io_usb1_inst_NXT
          .hps_io_hps_io_spim0_inst_CLK           (hps_io_hps_io_spim0_inst_CLK),          //
          .hps_io_spim0_inst_CLK
          .hps_io_hps_io_spim0_inst_MOSI              (hps_io_hps_io_spim0_inst_MOSI),   //
.hps_io_spim0_inst_MOSI
          .hps_io_hps_io_spim0_inst_MISO              (hps_io_hps_io_spim0_inst_MISO),   //
.hps_io_spim0_inst_MISO
          .hps_io_hps_io_spim0_inst_SS0               (hps_io_hps_io_spim0_inst_SS0),          //
          .hps_io_spim0_inst_SS0
          .hps_io_hps_io_spim1_inst_CLK               (hps_io_hps_io_spim1_inst_CLK),          //
          .hps_io_spim1_inst_CLK
          .hps_io_hps_io_spim1_inst_MOSI              (hps_io_hps_io_spim1_inst_MOSI),   //
.hps_io_spim1_inst_MOSI
          .hps_io_hps_io_spim1_inst_MISO              (hps_io_hps_io_spim1_inst_MISO),   //
.hps_io_spim1_inst_MISO
          .hps_io_hps_io_spim1_inst_SS0          (hps_io_hps_io_spim1_inst_SS0),       //
.hps_io_spim1_inst_SS0
          .hps_io_hps_io_uart0_inst_RX                          (hps_io_hps_io_uart0_inst_RX), //
          .hps_io_uart0_inst_RX
          .hps_io_hps_io_uart0_inst_TX                          (hps_io_hps_io_uart0_inst_TX), //
          .hps_io_uart0_inst_TX
          .hps_io_hps_io_i2c1_inst_SDA                          (hps_io_hps_io_i2c1_inst_SDA), //
          .hps_io_i2c1_inst_SDA
          .hps_io_hps_io_i2c1_inst_SCL                          (hps_io_hps_io_i2c1_inst_SCL),   //
          .hps_io_i2c1_inst_SCL
          .vga_R (VGA_R),
           .vga_G (VGA_G),
           .vga_B (VGA_B),
           .vga_CLK (VGA_CLK),
           .vga_HS (VGA_HS),
           .vga_VS (VGA_VS),
           .vga_BLANK_n (VGA_BLANK_n),
           .vga_SYNC_n (VGA_SYNC_n),
           .vga_MUSIC_CRL (VGA_MUSIC_CRL),

);
audio_top audio10(
          .OSC_50_B8A(OSC_50_B8A),

          .AUD_ADCLRCK(AUD_ADCLRCK),
          .AUD_ADCDAT(AUD_ADCDAT),
          .AUD_DACLRCK(AUD_DACLRCK),
          .AUD_DACDAT(AUD_DACDAT),
          .AUD_XCK(AUD_XCK),
          .AUD_BCLK(AUD_BCLK),
          .AUD_I2C_SCLK(AUD_I2C_SCLK),
          .AUD_I2C_SDAT(AUD_I2C_SDAT),
          .AUD_MUTE(AUD_MUTE),
          .KEY(KEY),
          .SW(SW),
          .LED(LED),
    .song_crl(VGA_MUSIC_CRL[3:0]),
    .song_begin(VGA_MUSIC_CRL[7])
    //.song_crl(SW),
    //.song_begin(KEY[3])
);

endmodule
```

4. audio_codec.v

```
          module audio_codec (
                    input  clk,
                    input  reset,
                    output [1:0]  sample_end,
                    output [1:0]  sample_req,
                    input  [15:0] audio_output,
                    output [15:0] audio_input,
                    // 1 - left, 0 - right
                    input  [1:0] channel_sel,

                    output AUD_ADCLRCK,
                    input AUD_ADCDAT,
                    output AUD_DACLRCK,
                    output AUD_DACDAT,
                    output AUD_BCLK
          );

          //reg [7:0] lrck_divider;
          reg [9:0] lrck_divider;
          reg [1:0] bclk_divider;

          reg [15:0] shift_out;
          reg [15:0] shift_temp;
          reg [15:0] shift_in;

          //wire lrck = !lrck_divider[7];
          wire lrck = !lrck_divider[9];

          assign AUD_ADCLRCK = lrck;
          assign AUD_DACLRCK = lrck;
          assign AUD_BCLK = bclk_divider[1];
```

```
assign AUD_DACDAT = shift_out[15];

always @(posedge clk) begin
        if (reset) begin
        //lrck_divider <= 8'hff;
          lrck_divider <= 10'h3ff;
        bclk_divider <= 2'b11;
        end else begin
        lrck_divider <= lrck_divider + 1'b1;
        bclk_divider <= bclk_divider + 1'b1;
        end
end

assign sample_end[1] = (lrck_divider == 8'h40);
assign sample_end[0] = (lrck_divider == 8'hc0);
assign audio_input = shift_in;
//assign sample_req[1] = (lrck_divider == 8'hfe);
assign sample_req[1] = (lrck_divider == 10'h3fe);
//assign sample_req[0] = (lrck_divider == 8'h7e);
assign sample_req[0] = (lrck_divider == 10'h1fe);

//wire clr_lrck = (lrck_divider == 8'h7f);
wire clr_lrck = (lrck_divider == 10'h1ff);
//wire set_lrck = (lrck_divider == 8'hff);
wire set_lrck = (lrck_divider == 10'h3ff);

// high right after bclk is set
wire set_bclk = (bclk_divider == 2'b10 && !lrck_divider[6]);
// high right before bclk is cleared
wire clr_bclk = (bclk_divider == 2'b11 && !lrck_divider[6]);

always @(posedge clk) begin
        if (reset) begin
        shift_out <= 16'h0;
        shift_in <= 16'h0;
        shift_in <= 16'h0;
        end else if (set_lrck || clr_lrck) begin
        // check if current channel is selected
        if (channel_sel[set_lrck]) begin
        shift_out <= audio_output;
        shift_temp <= audio_output;
        shift_in <= 16'h0;
        // repeat the sample from the other channel if not
        end else shift_out <= shift_temp;
        end else if (set_bclk == 1) begin
        // only read in if channel is selected
```

```
                if (channel_sel[lrck])
                shift_in <= {shift_in[14:0], AUD_ADCDAT};
                end else if (clr_bclk == 1) begin
                shift_out <= {shift_out[14:0], 1'b0};
                end
        end

        Endmodule
```

5. audio_effects.v

```
module audio_effects (
        input  clk,
        input  sample_end,
        input  sample_req,
        output [15:0] audio_output,
        input  [15:0] audio_input,
        input  [3:0]  control,
   input  [3:0]  song_crl,
   input  song_begin
);


reg [15:0] romdata;
reg [16:0]  index = 16'd0;

reg [15:0] last_sample;
reg [15:0] dat;

wire [15:0] trans1, trans2,trans3,trans4,trans5,trans6,trans7,trans8,trans9,trans10,trans11,trans12;

assign audio_output = dat;

parameter SINE       = 0;
parameter FEEDBACK = 1;

/*ROM1 Computer (.address(index[14:0]),.q(trans1),.clock(clk));
ROM2 Devil1 (.address(index),.q(trans2),.clock(clk));
ROM3 Door4 (.address(index),.q(trans3),.clock(clk));
ROM4 Drips (.address(index[12:0]),.q(trans4),.clock(clk));
*/
ROM5 Death (.address(index[14:0]),.q(trans5),.clock(clk));
//ROM6 Key (.address(index[11:0]),.q(trans6),.clock(clk));
```

```
//ROM7 Leakage (.address(index[14:0]),.q(trans7),.clock(clk));
/*ROM8 Monster1 (.address(index[13:0]),.q(trans8),.clock(clk));
ROM9 Monster2 (.address(index[14:0]),.q(trans9),.clock(clk));
*/
//ROM10 Move2 (.address(index[13:0]),.q(trans10),.clock(clk));
//ROM11 Move3 (.address(index[13:0]),.q(trans11),.clock(clk));
ROM12 Open5 (.address(index[13:0]),.q(trans12),.clock(clk));

always @(posedge clk) begin
        if (sample_end) begin
        last_sample <= audio_input;
        end

        if (sample_req) begin
        if (control[FEEDBACK])
        dat <= last_sample;
        else if (control[SINE]) begin

          if (song_begin == 1'b1)
          index <= 16'd00;
          else begin
                        case (song_crl)
                        //4'b0000 :
                        //begin
                                //if (index < 13'd5552) begin
                                        //index <= index + 1'b1;
                                        //dat <= trans1;
                                //end else
                                        //dat <= 16'd0;
                        //end

                        /*4'b0001 :     //ROM1
                        begin
                                if (index < 15'd18175) begin
                                        index <= index + 1'b1;
                                        dat <= trans1;
                                end else
                                        dat <= 16'd0;
                        end
                        4'b0010 : //ROM2
                        begin
                                if (index < 16'd46079) begin
                                        index <= index + 1'b1;
                                        dat <= trans2;
                                end else
                                        dat <= 16'd0;
```

```
end
4'b0011 : //ROM3
begin
        if (index < 16'd35839) begin
                index <= index + 1'b1;
                dat <= trans3;
        end else
                dat <= 16'd0;
end
4'b0100 : //ROM4
begin
        if (index < 13'd5000) begin
                index <= index + 1'b1;
                dat <= trans4;
        end else
                dat <= 16'd0;
end*/
4'b0101 : //ROM5
begin
        if (index < 15'd24569) begin
                index <= index + 1'b1;
                dat <= trans5;
        end else
                dat <= 16'd0;
end
/*4'b0110 ://ROM6
begin
        if (index < 12'd2559) begin
                index <= index + 1'b1;
                dat <= trans6;
        end else
                dat <= 16'd0;
end

4'b0111 : //ROM7
begin
        if (index < 15'd22775) begin
                index <= index + 1'b1;
                dat <= trans7;
        end else
                dat <= 16'd0;
end
4'b1000 : //ROM8
begin
        if (index < 14'd12791) begin
                index <= index + 1'b1;
```

```
                        dat <= trans8;
                    end else
                        dat <= 16'd0;
                end
                4'b1001 : //ROM9
                begin
                    if (index < 15'd16887) begin
                        index <= index + 1'b1;
                        dat <= trans9;
                    end else
                        dat <= 16'd0;
                end */
                /*4'b1010 : //ROM10
                begin
                    if (index < 14'd8951) begin
                        index <= index + 1'b1;
                        dat <= trans10;
                    end else
                        dat <= 16'd0;
                end*/
                /*4'b1011 : //ROM11
                begin
                    if (index < 14'd8951) begin
                        index <= index + 1'b1;
                        dat <= trans11;
                    end else
                        dat <= 16'd0;
                end*/
                4'b1100 : //ROM12
                begin
                    if (index < 14'd10231) begin
                        index <= index + 1'b1;
                        dat <= trans12;
                    end else
                        dat <= 16'd0;
                end

                default : dat <= 16'd0;
                endcase
  end
  //dat <= trans;

//if (index == 13'd5552)
//index <= 13'd00;
//else
//index <= index + 1'b1;
```

```
        end else
        dat <= 16'd0;
        end
end
//always @(posedge song_begin) begin
                //index <= 13'd00;
//end

Endmodule
```

6. audio_top.v

```
        module audio_top (
                input  OSC_50_B8A,

                inout  AUD_ADCLRCK,
                input  AUD_ADCDAT,
                inout  AUD_DACLRCK,
                output AUD_DACDAT,
                output AUD_XCK,
                inout  AUD_BCLK,
                output AUD_I2C_SCLK,
                inout  AUD_I2C_SDAT,
                output AUD_MUTE,

                input  [3:0] KEY,
                input  [3:0] SW,
                output [3:0] LED,
            input  [3:0] song_crl,
            input  song_begin
        );

        wire reset = !KEY[0];
        wire main_clk;
        wire audio_clk;

        wire [1:0] sample_end;
        wire [1:0] sample_req;
        wire [15:0] audio_output;
        wire [15:0] audio_input;

        clock_pll pll (
                .refclk (OSC_50_B8A),
                .rst (reset),
		.outclk_0 (audio_clk),
		.outclk_1 (main_clk)
);

i2c_av_config av_config (
		.clk (main_clk),
		.reset (reset),
		.i2c_sclk (AUD_I2C_SCLK),
		.i2c_sdat (AUD_I2C_SDAT),
		.status (LED)
);

assign AUD_XCK = audio_clk;
assign AUD_MUTE = (SW != 4'b0);

audio_codec ac (
		.clk (audio_clk),
		.reset (reset),
		.sample_end (sample_end),
		.sample_req (sample_req),
		.audio_output (audio_output),
		.audio_input (audio_input),
		.channel_sel (2'b10),

		.AUD_ADCLRCK (AUD_ADCLRCK),
		.AUD_ADCDAT (AUD_ADCDAT),
		.AUD_DACLRCK (AUD_DACLRCK),
		.AUD_DACDAT (AUD_DACDAT),
		.AUD_BCLK (AUD_BCLK)
);

audio_effects ae (
		.clk (audio_clk),
		.sample_end (sample_end[1]),
		.sample_req (sample_req[1]),
		.audio_output (audio_output),
		.audio_input  (audio_input),
		.control (SW),
    .song_crl (song_crl),
    .song_begin(song_begin)
);

endmodule
```

B. Software

1. Makefile

```
CFLAGS = -Wall

OBJECTS = main.o  fbputchar.o        usbkeyboard.o control.o campaign.o
general_movement.o

TARFILES = Makefile main.c \
        fbputchar.h fbputchar.c \
        usbkeyboard.h usbkeyboard.c\
        control.h control.c\
        campaign.h campaign.c

main : $(OBJECTS)
        cc $(CFLAGS) -o main $(OBJECTS) -lusb-1.0 -pthread



main.o : main.c fbputchar.h usbkeyboard.h control.h
fbputchar.o : fbputchar.c fbputchar.h
usbkeyboard.o : usbkeyboard.c usbkeyboard.h
control.o : control.c control.h
campaign.o : campaign.c campaign.h
general_movement.o : general_movement.c general_movement.h

.PHONY : clean
clean :
        rm -rf *.o *.c~ main
```

2. Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"
```

```
#include "campaign.h"

void music_play(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19+6)/256;
    message[7] = (x*19+6)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 10;
    message[11] = 0;
    message[12] = 0;
    message[13] = 140;
    write_segments(message);
    usleep(50);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19+6)/256;
    message[7] = (x*19+6)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 10;
    message[11] = 0;
    message[12] = 0;
    message[13] = 12;
    write_segments(message);
    usleep(50);


}


int main()
{


    //map_num stores current map number
    //map 1 is the initial map
    int map_num = 1;
    get_map_info(map_num);



    vga_led_arg_t vla;

    static const char filename[] = "/dev/vga_led";

    printf("VGA LED Userspace program started\n");

    if ( (vga_led_fd = open(filename, O_RDWR)) == -1) {
      fprintf(stderr, "could not open %s\n", filename);
      return -1;
    }

    //Open the keyboard
    if ( (keyboard = openkeyboard(&endpoint_address)) == NULL ) {
      printf("Did not find a keyboard\n");
      //exit(1);
    }



    music_play();
            while(1)
                    //general_movement_mode();
                    campaign_mode();

  return 0;
}
```

3. campaign.h

```
#ifndef _CAMPAIGN_H
#define _CAMPAIGN_H
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
```

```
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"
#include "campaign.h"

extern int campaing_mode_flag;

void dialog_01(int);
void dialog_02(int);
void campaign_mode();
void check_obj_outer();
void boss_fight();

#endif
```

4. campaign.c

```
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"
#include "campaign.h"

int campaign_mode_flag;
int map_num;
char possession[];

//dialogs
void dialog_01(int map_num){
  char dia_01[] = "Friday Night [11:50 PM]: Eric is writing a local console for an online
chatroom for CSEE 4840 Lab 2...";
        print_dialog(dia_01);
        print_pic();
  check_skip();
  char dia_02[] = "ERIC: I think I got it done...";
  print_dialog(dia_02);
```

```
            print_pic();
    check_skip();
    char dia_03[] = "ERIC: Let me test this local online chat room...";
    print_dialog(dia_03);
            print_pic();
    check_skip();
    char dia_04[] = "ERIC [TYPES IN CONSOLE]: Hello! O.o";
    print_dialog(dia_04);
            print_pic();
    check_skip();
    char dia_05[] = "COMPUTER: ERIC: Hello! O.o";
    print_dialog(dia_05);
            print_pic();
    check_skip();
    char dia_06[] = "ERIC [FEELING HAPPY]: Yes!! I think it's working!";
    print_dialog(dia_06);
            print_pic();
    check_skip();
    char dia_07[] = "...";
    print_dialog(dia_07);
            print_pic();
    check_skip();
    char dia_08[] = "ERIC: Ah, such relief. Let me walk around and strech a little bit. It's
better for me to not leave the room, just in case ";
    print_dialog(dia_08);
            print_pic();
    check_skip();
    retrieve_map(map_num, 400);
}

void dialog_02(int map_num){
            char dia_01[] = "ERIC: Let me save my code and go home.";
            print_dialog(dia_01);
            print_pic();
            check_skip();

            char dia_02[] = "ERIC: Wait... [Looking at the screen]";
            print_dialog(dia_02);
            print_pic();
            check_skip();

            char dia_03[] = "COMPUTER: 195.4.1.31: Hello Eric.";
            print_dialog(dia_03);
            print_pic();
            check_skip();
```

```
	char dia_04[] = "ERIC [Thinking to himself]: But this is a local console. Only
people in the room can access. But there is nobody here...";
	print_dialog(dia_04);
	print_pic();
	check_skip();

	char dia_05[] = "ERIC [typing back]: who is this?";
	print_dialog(dia_05);
	print_pic();
	check_skip();

	char dia_06[] = "COMPUTER:
@#$^$%$#&^%#^$%#@%$#@!!#%$&#^&^%$&$%...";
	print_dialog(dia_06);
	print_pic();
	check_skip();

	flash_off(7);

	char dia_07[] = "Eric: Whaaaat?? What is going on? Seems like the power is off. So
weird. Let me check the fuse.";
	print_dialog(dia_07);
	print_pic();
	check_skip();



}

void dialog_03(int map_num){
	char dia_01[] = "Eric: Ha. Found it! The fuse is right here.";
	print_dialog(dia_01);
	print_pic();
	check_skip();

	char dia_02[] = "Eric: It's too dark here, I cannot see whether the fuse is damaged.
Maybe I should find a new one and replace it?";
	print_dialog(dia_02);
	print_pic();
	check_skip();

	char dia_03[] = "Eric: I remember there is one in the electric cabinet close to the
entrance.";
	print_dialog(dia_03);
	print_pic();
	check_skip();
}
```

```
void dialog_04(int map_num){
	char dia_01[] = "Eric: Cool, it should be this cabinet, let me try to find the fuse.";
	print_dialog(dia_01);
	print_pic();
	check_skip();

	char dia_02[] = "...";
	print_dialog(dia_02);
	print_pic();
	check_skip();

	char dia_03[] = "Eric: Wait what's that sound... Who is screaming so late at
night?";
	print_dialog(dia_03);
	print_pic();
	check_skip();

	char dia_04[] = "Eric: NVM, I foudn the fuse. Let me go try it on.";
	print_dialog(dia_04);
	print_pic();
	check_skip();
}

void dialog_05(int map_num){
	char dia_01[] = "Eric: Oh no. It still does not work. Maybe the whole building has
been power down.";
	print_dialog(dia_01);
	print_pic();
	check_skip();

	char dia_02[] = "...";
	print_dialog(dia_02);
	print_pic();
	check_skip();

	char dia_03[] = "Eric (a little intimidated): Let me check whether other rooms
have power, and see whether there are other people on this floor.";
	print_dialog(dia_03);
	print_pic();
	check_skip();
}


void dialog_06(int map_num){
	char dia_01[] = "Eric: Looks like it is the whole building.";
```

```
        print_dialog(dia_01);
        print_pic();
        check_skip();

        char dia_02[] = "...";
        print_dialog(dia_02);
        print_pic();
        check_skip();

        char dia_03[] = "Eric: It seems to be too silent here.";
        print_dialog(dia_03);
        print_pic();
        check_skip();

        char dia_04[] = "Eric [shouting]: Hello? Anyone?";
        print_dialog(dia_04);
        print_pic();
        check_skip();

        char dia_05[] = "[Silence...]";
        print_dialog(dia_05);
        print_pic();
        check_skip();

        char dia_06[] = "Eric [thinking to himself] : creepy! What do I do...";
        print_dialog(dia_06);
        print_pic();
        check_skip();

}

void campaign_mode(){
        clean_screen(0);
   //print_pic();
        map_num = 1;
        set_start_location(map_num);
        print_map(map_num);
        face_up();

        campaign_mode_flag = 1;
        dialog_01(map_num);
        while(x == 14 || y == 12){
                general_movement(map_num);
                check_obj_outer(map_num);
        }
        while(x != 14 || y != 12 || get_facing_state() != 5|| map_num!=1){
```

```
			general_movement(map_num);
			check_obj_outer(map_num);
		}

		dialog_02(map_num);
		while(x != 32 || y != 5 ){
			general_movement(map_num);
			check_obj_outer(map_num);
		}
		dialog_03(map_num);
		while(x != 2 || y != 9 || get_facing_state() != 8|| map_num!=1){
			general_movement(map_num);
			check_obj_outer(map_num);
		}
		dialog_04(map_num);
		while(x != 32 || y != 5 || get_facing_state() != 5|| map_num!=1){
			general_movement(map_num);
			check_obj_outer(map_num);
		}
		dialog_05(map_num);
		while(map_num!=2){
			general_movement(map_num);
			check_obj_outer(map_num);
		}
		dialog_06(map_num);

}








int dialog_on_flag;
int check_same;

void check_obj_outer(int map_num){
		if (map_num == 1){

			if(((x == 10 && y == 12)||(x == 18 && y == 12)||(x == 22 && y == 12)||(x
== 26 && y == 12)||(x == 30 && y == 12)||(x == 8 && y == 18)||(x == 12 && y == 18)||(x
== 18 && y == 18)||(x == 24 && y == 18)||(x == 12 && y == 6)||(x == 16 && y == 6)||(x
== 24 && y == 6)) && get_facing_state() == 5)
			{
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "Seems like I can only log into one
computer at a time.";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}
			}
			else if(x == 2 && y == 21 && get_facing_state() == 8){
				if (dialog_on_flag == 1){}
				else{
					char electric_cabinet[] = "Cabinet: there are micro-USB
cables, SocKit boards, mice, keyboards, VGA cables...";
					print_dialog(electric_cabinet);
					dialog_on_flag = 1;
				}

			}
			else if(x == 2 && y == 15 && get_facing_state() == 8){
				if (dialog_on_flag == 1){}
				else{
					char electric_cabinet[] = "Empty electric cabinet.";
					print_dialog(electric_cabinet);
					dialog_on_flag = 1;
				}

			}
			else if(x == 12 && y == 21 && get_facing_state() == 2){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "Cabinet: there are micro-USB
cables, SocKit boards, mice, keyboards, VGA cables...";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}

			}
			else if(((x == 25 && y == 24)||(x == 28 && y == 6)||(x == 29 && y == 6))
&& get_facing_state() == 5){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "Locked.";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}
			}
			else if(((x == 30 && y == 23) || (x == 30 && y == 24)) &&
get_facing_state() == 11){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "Old couch: people nap here a lot,
especially during final seasons. Ewwwwww.";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}

			}
			else if(x == 8 && y == 7 && get_facing_state() == 5){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "Damaged FPGA boards. What a
shame! I really hope poeple can be more careful when using them...";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}

			}
			else if(x == 4 && y == 6 && get_facing_state() == 5){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "It might not be a good idea for me to
leave things behind in this room.";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}

			}
			else if(x == 32 && y == 5 && get_facing_state() == 5){
				if (dialog_on_flag == 1){}
				else{
					char computer_off[] = "[WARNING]: Power control for this
room.";
					print_dialog(computer_off);
					dialog_on_flag = 1;
				}

			}

			else if (dialog_on_flag == 1){
				dialog_on_flag = 0;
				retrieve_map(map_num, 400);
```

```
                }
        }
}
```

5. control.h

```
#ifndef _CONTROL_H
#define _CONTROL_H
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"


#define DIALOG_HEIGHT 80
#define FONT_HEIGHT 16
#define FONT_WIDTH 8


//current map information pointer
int (*map)[34];


extern char file_name[5];

//(x,y) eric's current position
extern int x;
extern int y;
extern int map_num;
extern char possession[];

//last map number
extern int last_map_num;
```

```
struct libusb_device_handle *keyboard;
uint8_t endpoint_address;
struct usb_keyboard_packet packet;
char keystate[12];
int transferred;
static const int offset = 0;
extern unsigned char message[14];

extern int if_event;
//extern int dialog_on_flag;


//character facing state
//Default = UP
//Up = U Down = D Left = L Right = R
extern char facing_state;


int vga_led_fd;



//control function declaration
void set_start_location(int);
void get_map_info(int);
int get_facing_state();
void eric_void();
void face_up();
void face_down();
void face_left();
void face_right();

void move_up();
void move_down();
void move_left();
void move_right();
void print_map(int);
void print_dialog(char*);
void clean_screen(int);
void retrieve_map(int, int);
void check_space();
void check_skip();
void general_movement(int map_num);
void dialog_01(int);
void flash_on(int);
void flash_off(int);
void print_pic();
/********************************
*      map builder instruction
*
*      -1 for obstacle
*      0 for path
*      other number for map_switch
*
*
********************************/

static const int map1[26][34] = {
 /* 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 */
   {-1, -1, -1, -1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1, -1, -1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1, -1, -1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1, -1, -1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1,  0, -1},/* 5 */
   {-1, -1,  0,  0,  0,  0,  0,  0, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0, -1, -1,  0,
0,  0,  0,  0,  0,  0},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1,  0,
0,  0,  0,  0,  0,  0},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},/* 10 */
   { 0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0, -1, -1, -1,  0, -1, -1, -1,  0, -1, -1,
-1,  0,  0,  0,  0},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0, -1, -1, -1,  0, -1, -1, -1,  0,  0,  0,
0,  0,  0, -1, -1},
   {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},/* 15 */
  {-1, -1,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0,  0,
0,  0,  0, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1,
0,  0,  0,  0,  0,  0},/* 20 */
  {-1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,
0,  0,  0, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,
0,  0, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,
0,  0, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0, -1, -1, -1},
  {-1, -1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0}
};



static const int map2[26][34] = {
 /* 0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
27  28  29  30  31  32  33 */
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1,  0,
0,  0,  0,  3,  3},/* 15 */
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  3,  3},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  3,  3},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  3,  3},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1}
};

static const int map3[26][34] = {
 /* 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
27  28  29  30  31  32  33 */
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 2,  2,  0,  0,  0, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1,
-1,  0,  0,  4,  4},/* 15 */
  { 2,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  4,  4},
  { 2,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  4,  4},
  { 2,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  4,  4},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1}
};

static const int map4[26][34] = {
 /* 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 */
  { 3,  3,  3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 3,  3,  3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},/* 5 */
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  { 0,  0,  0,  0,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},/* 10 */
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},/* 15 */
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},/* 20 */
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1}
};


static unsigned char font[] = {
   0x00, 0x00, 0x7e, 0xc3, 0x99, 0x99, 0xf3, 0xe7, 0xe7, 0xff, 0xe7, 0xe7, 0x7e, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xdc, 0x00, 0x76, 0xdc, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
   0x00, 0x00, 0x6e, 0xf8, 0xd8, 0xd8, 0xdc, 0xd8, 0xd8, 0xd8, 0xf8, 0x6e, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0xdb, 0xdb, 0xdf, 0xd8, 0xdb, 0x6e, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x88, 0x88, 0xf8, 0x88, 0x88, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08, 0x00,
0x00, 0x00, 0x00,
   0x00, 0xf8, 0x80, 0xe0, 0x80, 0x80, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x70, 0x88, 0x80, 0x88, 0x70, 0x00, 0x3c, 0x22, 0x3c, 0x24, 0x22, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x80, 0x80, 0x80, 0x80, 0xf8, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20, 0x00, 0x00,
0x00, 0x00,
   0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11,
0x44,
   0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa,
0x55, 0xaa,
   0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77,
0xdd, 0x77,
   0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
    0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
0xf0,
    0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f,
0x0f,
    0x00, 0x88, 0xc8, 0xa8, 0x98, 0x88, 0x00, 0x20, 0x20, 0x20, 0x20, 0x3e, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x88, 0x88, 0x50, 0x50, 0x20, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x0e, 0x38, 0xe0, 0x38, 0x0e, 0x00, 0xfe, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0xe0, 0x38, 0x0e, 0x38, 0xe0, 0x00, 0xfe, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x06, 0x0c, 0xfe, 0x18, 0x30, 0xfe, 0x60, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x06, 0x1e, 0x7e, 0xfe, 0x7e, 0x1e, 0x06, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0xc0, 0xf0, 0xfc, 0xfe, 0xfc, 0xf0, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x0c, 0xfe, 0x0c, 0x18, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x60, 0xfe, 0x60, 0x30, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x28, 0x6c, 0xfe, 0x6c, 0x28, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x06, 0x36, 0x66, 0xfe, 0x60, 0x30, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xfe, 0x6e, 0x6c, 0x6c, 0x6c, 0x6c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x3c, 0x3c, 0x3c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x66, 0x66, 0x66, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x10, 0x10, 0x7c, 0xd6, 0xd0, 0xd0, 0x7c, 0x16, 0x16, 0xd6, 0x7c, 0x10, 0x10,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0xc2, 0xc6, 0x0c, 0x18, 0x30, 0x60, 0xc6, 0x86, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x38, 0x6c, 0x6c, 0x38, 0x76, 0xdc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x18, 0x0c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x30, 0x18, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3c, 0xff, 0x3c, 0x66, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7e, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x18, 0x30,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xce, 0xce, 0xd6, 0xd6, 0xe6, 0xe6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0x06, 0x06, 0x3c, 0x06, 0x06, 0x06, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x0c, 0x1c, 0x3c, 0x6c, 0xcc, 0xfe, 0x0c, 0x0c, 0x0c, 0x1e, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfe, 0xc0, 0xc0, 0xc0, 0xfc, 0x06, 0x06, 0x06, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x38, 0x60, 0xc0, 0xc0, 0xfc, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfe, 0xc6, 0x06, 0x06, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x06, 0x06, 0x0c, 0x78, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x18, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x00, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x0c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xde, 0xde, 0xde, 0xdc, 0xc0, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x66, 0x66, 0x66, 0x66, 0xfc, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xc0, 0xc0, 0xc2, 0x66, 0x3c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xf8, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x6c, 0xf8, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68, 0x60, 0x62, 0x66, 0xfe, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xde, 0xc6, 0xc6, 0x66, 0x3a, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x1e, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0xcc, 0xcc, 0xcc, 0x78, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xe6, 0x66, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xf0, 0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0x62, 0x66, 0xfe, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xc6, 0xee, 0xfe, 0xfe, 0xd6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xc6, 0xe6, 0xf6, 0xfe, 0xde, 0xce, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x60, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xde, 0x7c, 0x0c, 0x0e,
0x00, 0x00,
    0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x6c, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x60, 0x38, 0x0c, 0x06, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x7e, 0x7e, 0x5a, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x6c, 0x38, 0x10, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0xee, 0x6c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xc6, 0xc6, 0x6c, 0x7c, 0x38, 0x38, 0x7c, 0x6c, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xfe, 0xc6, 0x86, 0x0c, 0x18, 0x30, 0x60, 0xc2, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x3c, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x3c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00,
0x00, 0x00, 0x00,
   0x00, 0x00, 0x3c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x3c, 0x00, 0x00,
0x00, 0x00,
   0x10, 0x38, 0x6c, 0xc6, 0x00, 0x00, 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, 0x30, 0x30, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0xe0, 0x60, 0x60, 0x78, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc0, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x1c, 0x0c, 0x0c, 0x3c, 0x6c, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x38, 0x6c, 0x64, 0x60, 0xf0, 0x60, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0xcc,
0x78, 0x00,
   0x00, 0x00, 0xe0, 0x60, 0x60, 0x6c, 0x76, 0x66, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x06, 0x06, 0x00, 0x0e, 0x06, 0x06, 0x06, 0x06, 0x06, 0x06, 0x66,
0x66, 0x3c, 0x00,
   0x00, 0x00, 0xe0, 0x60, 0x60, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x70, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x34, 0x18, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xec, 0xfe, 0xd6, 0xd6, 0xd6, 0xd6, 0xc6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x60, 0x60,
0xf0, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0x0c,
0x1e, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x76, 0x66, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0x60, 0x38, 0x0c, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x10, 0x30, 0x30, 0xfc, 0x30, 0x30, 0x30, 0x30, 0x36, 0x1c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0x6c, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0x6c, 0x38, 0x38, 0x38, 0x6c, 0xc6, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x0c,
0xf8, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xcc, 0x18, 0x30, 0x60, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x0e, 0x18, 0x18, 0x18, 0x70, 0x18, 0x18, 0x18, 0x18, 0x0e, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x70, 0x18, 0x18, 0x18, 0x0e, 0x18, 0x18, 0x18, 0x18, 0x70, 0x00, 0x00,
0x00, 0x00,
   0x00, 0x00, 0x76, 0xdc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
   0x00, 0x66, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
};
```

```
#endif
```

6. Control.c

```
//control function definition

#include "control.h"
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"


char file_name[5];
int x = 0;
int y = 0;
int last_map_num = 0;
unsigned char message[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char facing_state = 'U';
int if_event = 0;
char possession[];


/* Read and print the segment values */
void print_segment_info() {
  vga_led_arg_t vla;
  int i;

  for (i = 0 ; i < VGA_LED_DIGITS ; i++) {
    vla.digit = i;
    if (ioctl(vga_led_fd, VGA_LED_READ_DIGIT, &vla)) {
      perror("ioctl(VGA_LED_READ_DIGIT) failed");
      return;
    }
    printf("%02x ", vla.segments);
  }
  printf("\n");
}

/* Write the contents of the array to the display */
void write_segments(const unsigned char segs[8])
{
  vga_led_arg_t vla;
  int i;
  for (i = 0 ; i < VGA_LED_DIGITS ; i++) {
    vla.digit = i;
    vla.segments = segs[i];
    if (ioctl(vga_led_fd, VGA_LED_WRITE_DIGIT, &vla)) {
      perror("ioctl(VGA_LED_WRITE_DIGIT) failed");
      return;
    }
  }
}




//get start location function
void set_start_location(int map_num){
  switch(map_num){
      case 1:
            if (last_map_num < map_num){
            x = 14;
            y = 12;
            }
            else{
                  x = 4;
                  y = 5;
                  face_down();
            }
        break;
      case 2:
        if (last_map_num < map_num){
            x = 6;
            y = 15;
                  face_down();
            }
            else{
                  x = 31;
                  y = 17;
                  face_left();
            }
        break;
      case 3:
        if (last_map_num < map_num){
            x = 2;
            y = 17;
                    face_right();
            }
            else{
                    x = 31;
                    y = 17;
                    face_left();
            }
        break;
      case 4:
        if (last_map_num < map_num){
            x = 1;
            y = 3;
                    face_down();
            }
            else{
                    x = 26;
                    y = 18;
                    face_up();
            }
        break;
    }
}


//assign new map information

void get_map_info(int map_num){
  switch(map_num){
      case 1:
        map = map1;
        break;
      case 2:
        map = map2;
        break;
      case 3:
        map = map3;
        break;
      case 4:
        map = map4;
        break;
    }
```

```
}

int get_facing_state(){
    int temp;
    switch(facing_state){
      case 'U':
        temp = 5;
        return temp;
        break;
      case 'D':
        temp = 2;
        return temp;
        break;
      case 'L':
        temp = 8;
        return temp;
        break;
      case 'R':
        temp = 11;
        return temp;
        break;
    }
}




//eric void
void eric_void(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = 0;
    message[7] = 0;
    message[8] = 0;
    message[9] = 0;
    message[10] = 5;
    message[11] = 0;
    write_segments(message);
    usleep(100000);
}
```

```
//character facing
void face_up(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 5;
    message[11] = 0;
    write_segments(message);
    facing_state = 'U';
    usleep(100);
}

void face_down(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 2;
    message[11] = 0;
    write_segments(message);
    facing_state = 'D';
    usleep(100);
}

void face_left(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 8;
    message[11] = 0;
    write_segments(message);
    facing_state = 'L';
    usleep(100);
}

void face_right(){
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 11;
    message[11] = 0;
    write_segments(message);
    facing_state = 'R';
    usleep(100);
}


void move_up(){
  if (facing_state != 'U'){
    face_up();
  }
  else{
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18-6)/256;
    message[9] = (y*18-6)%256;
    message[10] = 4;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18-12)/256;
    message[9] = (y*18-12)%256;
    message[10] = 6;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18-18)/256;
    message[9] = (y*18-18)%256;
    message[10] = 5;
    message[11] = 0;
    write_segments(message);
    usleep(100);
    y -= 1;
  }
}


void move_down(){
  if (facing_state != 'D'){
    face_down();
  }
  else{
         printf("23333333");
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18+6)/256;
    message[9] = (y*18+6)%256;
    message[10] = 1;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18+12)/256;
    message[9] = (y*18+12)%256;
    message[10] = 3;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18+18)/256;
    message[9] = (y*18+18)%256;
    message[10] = 2;
    message[11] = 0;
    write_segments(message);
    usleep(100);
    y += 1;
  }
}

void move_left(){
  if (facing_state != 'L'){
    face_left();
  }
  else{
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19-6)/256;
    message[7] = (x*19-6)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 7;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19-13)/256;
    message[7] = (x*19-13)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 9;
    message[11] = 0;
    write_segments(message);
    usleep(50000);
    message[0] = 0;
    message[1] = 0;
    message[2] = 0;
    message[3] = 0;
    message[4] = 0;
    message[5] = 0;
    message[6] = (x*19-19)/256;
    message[7] = (x*19-19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = 8;
    message[11] = 0;
    write_segments(message);
    usleep(100);
    x -= 1;
  }
}

void move_right(){
```

```
if (facing_state != 'R'){
  face_right();
}
else{
  message[0] = 0;
  message[1] = 0;
  message[2] = 0;
  message[3] = 0;
  message[4] = 0;
  message[5] = 0;
  message[6] = (x*19+6)/256;
  message[7] = (x*19+6)%256;
  message[8] = (y*18)/256;
  message[9] = (y*18)%256;
  message[10] = 10;
  message[11] = 0;
  write_segments(message);
  usleep(50000);
  message[0] = 0;
  message[1] = 0;
  message[2] = 0;
  message[3] = 0;
  message[4] = 0;
  message[5] = 0;
  message[6] = (x*19+13)/256;
  message[7] = (x*19+13)%256;
  message[8] = (y*18)/256;
  message[9] = (y*18)%256;
  message[10] = 12;
  message[11] = 0;
  write_segments(message);
  usleep(50000);
  message[0] = 0;
  message[1] = 0;
  message[2] = 0;
  message[3] = 0;
  message[4] = 0;
  message[5] = 0;
  message[6] = (x*19+19)/256;
  message[7] = (x*19+19)%256;
  message[8] = (y*18)/256;
  message[9] = (y*18)%256;
  message[10] = 11;
  message[11] = 0;
  write_segments(message);
  usleep(100);
```

```
    x += 1;
  }
}

void general_movement(int map_num){

    libusb_interrupt_transfer(keyboard, endpoint_address,
            (unsigned char *) &packet, sizeof(packet),
            &transferred, 0);
    if (transferred == sizeof(packet)) {
    printf("\n");
    printf("%02x %02x %02x\n", packet.modifiers, packet.keycode[0],packet.keycode[1]);
    }

            //space
    if (packet.keycode[0] == 0x2c){
            print_dialog(possession);
    }

    //right
    else if (packet.keycode[0] == 0x4f && x <= 31 && map[y][x+1] != -1){
       move_right();
    }
    //left
    else if (packet.keycode[0] == 0x50 && x >= 1 && map[y][x-1] != -1){
       move_left();
    }
    //down
    else if (packet.keycode[0] == 0x51 && y <= 25 && map[y+1][x] != -1){
       move_down();
    }
    //up
    else if (packet.keycode[0] == 0x52 && y >= 1 && map[y-1][x] != -1){
       move_up();
    }
    //change facing when encounter obstacle
    //right
    else if (packet.keycode[0] == 0x4f){
         face_right();
    }
    //left
    else if (packet.keycode[0] == 0x50){
         face_left();
    }
    //down
    else if (packet.keycode[0] == 0x51){
        face_down();
    }
    //up
    else if (packet.keycode[0] == 0x52){
        face_up();
    }





    printf("\nx: %d, y: %d", x, y);
          printf("\nmap_num = %d", map_num);
          printf("\nget_facing_state: %d", get_facing_state());




    if (map[y][x] != 0 && map[y][x] != -1){
          printf("switched map");
      eric_void();
      last_map_num = map_num;
      map_num = map[y][x];
      print_map(map_num);
      get_map_info(map_num);
      set_start_location(map_num);
    }

   // if ( if_event != 0){
    // break;
   // }

}


//send map information and print map
//NOTE: print_map does will set Eric's location to (0,0) (except begining of the game)
//which means Eric will not print out when print map is called
//in order to print Eric's sprite, please use set_start_location()

void print_map(int map_num){
   if (last_map_num != 0){
           x = 0;
           y = 0;
    }
    FILE * fp;
    char * line = NULL;
    size_t len = 0;
```

```
ssize_t read;
int count = 0;
char temp[] = "0000";

int r = 0;
int g = 0;
int b = 0;
int addr_1 = 0;
int addr_2 = 0;
int addr_3 = 0;

switch(map_num){
  case 1:
    strcpy(file_name, "1.mif");
    break;
  case 2:
    strcpy(file_name, "2.mif");
    break;
  case 3:
    strcpy(file_name, "3.mif");
    break;
  case 4:
    strcpy(file_name, "4.mif");
    break;
}


fp = fopen(file_name, "r");
if (fp == NULL)
    exit(EXIT_FAILURE);


while ((read = getline(&line, &len, fp)) != -1) {
  if (count >= 9 && line[0] != 'E'){
        int i = offset;
    for (i; i <= offset+3; i++){
        temp[i-offset] = line[i];
        if (temp[i-offset] == ' '){
          temp[i-offset] = '0';
        }
    }
    int data = atoi(temp);
    int address = count - 9;
r = data/256;
g = (data%256)/16;
b = data%16;
```

```
    addr_1 = address/65536;
    addr_2 = (address%65536)/256;
    addr_3 = address%256;
    message[0] = r;
    message[1] = g;
    message[2] = b;
    message[3] = addr_1;
    message[4] = addr_2;
    message[5] = addr_3;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = get_facing_state();
    message[11] = 1;
    write_segments(message);

      }
      count++;
    }

    fclose(fp);

    message[11] = 0;
    write_segments(message);


}

void print_pic(){
    FILE * fp;
    char * line = NULL;
    size_t len = 0;
    ssize_t read;
    int count = 0;
    char temp[] = "0000";

    int r = 0;
    int g = 0;
    int b = 0;
    int addr_1 = 0;
    int addr_2 = 0;
    int addr_3 = 0;

    fp = fopen("eric_gu.mif", "r");
    if (fp == NULL)
```

```
        exit(EXIT_FAILURE);


    while ((read = getline(&line, &len, fp)) != -1) {
      if (count >= 9 && line[0] != 'E'){
            int i = offset;
        for (i; i <= offset+3; i++){
            temp[i-offset] = line[i];
            if (temp[i-offset] == ' '){
              temp[i-offset] = '0';
            }
        }
        int data = atoi(temp);
        int address = 640*400+560+((count-9)/80)*640+(count-9)%80;
    r = data/256;
    g = (data%256)/16;
    b = data%16;
    addr_1 = address/65536;
    addr_2 = (address%65536)/256;
    addr_3 = address%256;
    message[0] = r;
    message[1] = g;
    message[2] = b;
    message[3] = addr_1;
    message[4] = addr_2;
    message[5] = addr_3;
    message[6] = (x*19)/256;
    message[7] = (x*19)%256;
    message[8] = (y*18)/256;
    message[9] = (y*18)%256;
    message[10] = get_facing_state();
    message[11] = 1;
    write_segments(message);
    message[11] = 0;
    write_segments(message);


      }
      count++;
    }

    fclose(fp);



}
```

```
void clean_screen(int start_height){
   int address = start_height * 640;
   int addr_1 = 0;
   int addr_2 = 0;
   int addr_3 = 0;

   for (address; address < 480*640; address++){
          addr_1 = address/65536;
          addr_2 = (address%65536)/256;
          addr_3 = address%256;
          message[0] = 0;
          message[1] = 0;
          message[2] = 0;
          message[3] = addr_1;
          message[4] = addr_2;
          message[5] = addr_3;
          message[6] = (x*19)/256;
          message[7] = (x*19)%256;
          message[8] = (y*18)/256;
          message[9] = (y*18)%256;
          message[10] = get_facing_state();
          message[11] = 1;
          write_segments(message);
   }
   message[11] = 0;
   write_segments(message);
}

void flash_on(int times){
          int i;
   for (i=0; i < times; i++){
          message[12] = 1;
          write_segments(message);
                   usleep(200000);
          message[12] = 0;
          write_segments(message);
                   usleep(200000);

   }
}

void flash_off(int times){
          int i;
   for (i=0; i < times; i++){
```

```
            message[12] = 0;
            write_segments(message);
                    usleep(100000);
            message[12] = 1;
            write_segments(message);
                    usleep(100000);

    }
}


void print_dialog(char *msg){

    int r = 0;
    int g = 0;
    int b = 0;
    int addr_1 = 0;
    int addr_2 = 0;
    int addr_3 = 0;
    int address = 0;
    char *msgp = msg;
    int pixel;
    int dialog_x, dialog_y, dialog_col, dialog_row;
    dialog_col = 0;
    dialog_row = 0;
    unsigned char pixels, *pixelp = font + FONT_HEIGHT * (*msg);

    clean_screen(480 - DIALOG_HEIGHT);


    while(*msgp != '\0'){
    pixelp = font + FONT_HEIGHT * (*msgp);
      for (dialog_y = 0 ; dialog_y < FONT_HEIGHT ; dialog_y++) {
        pixels = *pixelp++;
        for (dialog_x = 0 ; dialog_x < FONT_WIDTH ; dialog_x++, pixels <<= 1){
          if (pixels & 0x80) {
            pixel = 0x0FFF;
          }
          else {
            pixel = 0x0000;
          }
          address = (480 - DIALOG_HEIGHT + dialog_row * FONT_HEIGHT) * 640 +
dialog_y * 640 + dialog_col * FONT_WIDTH + dialog_x;
          r = pixel/256;
          g = (pixel%256)/16;
          b = pixel%16;
```

```
            addr_1 = (address >> 16);
            addr_2 = ((address%65536)>>8);
            addr_3 = address%256;
            message[0] = r;
            message[1] = g;
            message[2] = b;
            message[3] = addr_1;
            message[4] = addr_2;
            message[5] = addr_3;
            message[6] = (x*19)/256;
            message[7] = (x*19)%256;
            message[8] = (y*18)/256;
            message[9] = (y*18)%256;
            message[10] = get_facing_state();
            message[11] = 1;
            write_segments(message);
            message[11] = 0;
            write_segments(message);
        }
                    }
        msgp++;
        if (dialog_col < 60){
          dialog_col++;
        }
        else{
          dialog_col = 0;
          dialog_row++;
        }
      }

      message[11] = 0;
      write_segments(message);

    }



    void retrieve_map(int map_num, int height){
      FILE * fp;
      char * line = NULL;
      size_t len = 0;
      ssize_t read;
      int count = 0;
      char temp[] = "0000";

      int r = 0;
```

```
int g = 0;
int b = 0;
int addr_1 = 0;
int addr_2 = 0;
int addr_3 = 0;

switch(map_num){
  case 1:
    strcpy(file_name, "1.mif");
    break;
  case 2:
    strcpy(file_name, "2.mif");
    break;
  case 3:
    strcpy(file_name, "3.mif");
    break;
  case 4:
    strcpy(file_name, "4.mif");
    break;
}


fp = fopen(file_name, "r");
if (fp == NULL)
    exit(EXIT_FAILURE);


while ((read = getline(&line, &len, fp)) != -1) {
  if (count >= (9 + 640 * height) && line[0] != 'E'){
          int i = offset;
    for (i; i <= offset+3; i++){
          temp[i-offset] = line[i];
          if (temp[i-offset] == ' '){
            temp[i-offset] = '0';
          }
    }
    int data = atoi(temp);
    int address = count - 9;
r = data/256;
g = (data%256)/16;
b = data%16;
addr_1 = address/65536;
addr_2 = (address%65536)/256;
addr_3 = address%256;
message[0] = r;
message[1] = g;
```

```
   message[2] = b;
   message[3] = addr_1;
   message[4] = addr_2;
   message[5] = addr_3;
   message[6] = (x*19)/256;
   message[7] = (x*19)%256;
   message[8] = (y*18)/256;
   message[9] = (y*18)%256;
   message[10] = get_facing_state();
   message[11] = 1;
   write_segments(message);

    }
    count++;
   }

   fclose(fp);

   message[11] = 0;
   write_segments(message);


}


void check_skip(){
   while(1){
         libusb_interrupt_transfer(keyboard, endpoint_address,(unsigned char *) &packet,
sizeof(packet),&transferred, 0);
         if (transferred == sizeof(packet)) {
           printf("\n");
           printf("%02x %02x %02x\n", packet.modifiers,
packet.keycode[0],packet.keycode[1]);
         }
             if (packet.keycode[0] == 0x49 || packet.keycode[0] == 0x50 ||
packet.keycode[0] == 0x51 || packet.keycode[0] == 0x52){
           break;
             }
   }

}

void check_space(){
   while(1){
         libusb_interrupt_transfer(keyboard, endpoint_address,(unsigned char *) &packet,
sizeof(packet),&transferred, 0);
```

```
            if (transferred == sizeof(packet)) {
              printf("\n");
              printf("%02x %02x %02x\n", packet.modifiers,
packet.keycode[0],packet.keycode[1]);
            }
                if (packet.keycode[0] == 0x2c){
              break;
                }
    }

}
```

7. General_movement.h

```
#ifndef _GENERAL_MOVEMENT_H
#define _GENERAL_MOVEMENT_H
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"
#include "campaign.h"

extern int general_movement_mode_flag;

void general_movement_mode();
#endif
```

8. General_mobement.c

```
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "usbkeyboard.h"
#include "fbputchar.h"
#include "control.h"
#include "campaign.h"

int campaign_mode_flag;
int map_num;
int general_movement_mode_flag;



void general_movement_mode(){
        campaign_mode_flag = 0;
        general_movement_mode_flag = 1;
        clean_screen(0);
  //print_pic();
        map_num = 1;
        set_start_location(map_num);
        print_map(map_num);
        face_up();
        while(1)
                general_movement(map_num);
}
```

9. fbputchar.h

```
#ifndef _FBPUTCHAR_H
#  define _FBPUTCHAR_H



#define FBOPEN_DEV -1          /* Couldn't open the device */
#define FBOPEN_FSCREENINFO -2  /* Couldn't read the fixed info */
#define FBOPEN_VSCREENINFO -3  /* Couldn't read the variable info */
#define FBOPEN_MMAP -4         /* Couldn't mmap the framebuffer memory */
#define FBOPEN_BPP -5          /* Unexpected bits-per-pixel */

extern int fbopen(void);
extern void fbputchar(char, int, int);
extern int fbputs(const char *, int, int);
extern void fbputsn(const char *, int, int, int);
extern void clearscreen();
extern void clearrow();

#endif
```

10. Fbputchar.c

```
/*
 * fbputchar: Framebuffer character generator
 *
 * Assumes 32bpp
 *
 * References:
 *
 *
http://web.njit.edu/all_topics/Prog_Lang_Docs/html/qt/emb-framebuffer-howto.html
 * http://www.diskohq.com/docu/api-reference/fb_8h-source.html
 */

#include "fbputchar.h"
#include <linux/types.h>
#include <linux/stat.h>
#include <fcntl.h>
#include <linux/mman.h>
#include <linux/ioctl.h>

#include <linux/fb.h>

#define FBDEV "/dev/fb0"

#define FONT_WIDTH 8
#define FONT_HEIGHT 16
#define BITS_PER_PIXEL 32

struct fb_var_screeninfo fb_vinfo;
struct fb_fix_screeninfo fb_finfo;
unsigned char *framebuffer;
static unsigned char font[];

/*
 * Open the framebuffer to prepare it to be written to.  Returns 0 on success
 * or one of the FBOPEN_... return codes if something went wrong.
 */
int fbopen()
```

```
{
 int fd = open(FBDEV, O_RDWR); /* Open the device */
 if (fd == -1) return FBOPEN_DEV;

 if (ioctl(fd, FBIOGET_FSCREENINFO, &fb_finfo)) /* Get fixed info about fb */
   return FBOPEN_FSCREENINFO;

 if (ioctl(fd, FBIOGET_VSCREENINFO, &fb_vinfo)) /* Get varying info about fb */
   return FBOPEN_VSCREENINFO;

 if (fb_vinfo.bits_per_pixel != 32) return FBOPEN_BPP; /* Unexpected */

 framebuffer = mmap(0, fb_finfo.smem_len, PROT_READ | PROT_WRITE,
                               MAP_SHARED, fd, 0);
 if (framebuffer == (unsigned char *)-1) return FBOPEN_MMAP;

 return 0;
}

/*
 * Draw the given character at the given row/column.
 * fbopen() must be called first.
 */
void fbputchar(char c, int row, int col)
{
 int x, y;
 unsigned char pixels, *pixelp = font + FONT_HEIGHT * c;
 unsigned char *pixel, *left = framebuffer +
   (row * FONT_HEIGHT + fb_vinfo.yoffset) * fb_finfo.line_length +
   (col * FONT_WIDTH + fb_vinfo.xoffset) * BITS_PER_PIXEL / 8;
 for (y = 0 ; y < FONT_HEIGHT ; y++, left += fb_finfo.line_length) {
   pixels = *pixelp++;
   pixel = left;
   for (x = 0 ; x < FONT_WIDTH ; x++, pixels <<= 1, pixel += 4)
     if (pixels & 0x80) {
           pixel[0] = 255; /* Blue */
       pixel[1] = 255; /* Green */
       pixel[2] = 255; /* Red */
       pixel[3] = 0;
     } else {
           pixel[0] = 0;
       pixel[1] = 0;
       pixel[2] = 0;
       pixel[3] = 0;
     }
 }
}


void clearscreen()
{


            int x, y;
  unsigned char pixels, *pixelp = font + FONT_HEIGHT ;
  unsigned char *pixel, *left = framebuffer +
    (fb_vinfo.yoffset) * fb_finfo.line_length +
    (fb_vinfo.xoffset) * BITS_PER_PIXEL / 8;
  for (y = 0 ; y < 768 ; y++, left += fb_finfo.line_length) {
    pixels = *pixelp++;
    pixel = left;
    for (x = 0 ; x < 1024 ; x++, pixels <<= 1, pixel += 4)
     {
       pixel[0] = 0;
       pixel[1] = 0;
       pixel[2] = 0;
       pixel[3] = 0;
      }
  }
}


/*
 * Draw the given string at the given row/column.
 * String must fit on a single line: wrap-around is not handled.
 */
int fbputs(const char *s, int row, int col)
{
  char c;
  while ((c = *s++) != 0) fbputchar(c, row, col++);
  return row;
}

void fbputsn(const char *s, int row, int col, int numb)
{
  char c;
  while ((c = *s++) != 0 && col < numb)
            fbputchar(c, row, col++);
}

/* 8 X 16 console font from /lib/kbd/consolefonts/lat0-16.psfu.gz
```

```
od --address-radix=n --width=16 -v -t x1 -j 4 -N 2048 lato-16.psfu

*/

static unsigned char font[] = {
  0x00, 0x00, 0x7e, 0xc3, 0x99, 0x99, 0xf3, 0xe7, 0xe7, 0xff, 0xe7, 0xe7, 0x7e, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xdc, 0x00, 0x76, 0xdc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x6e, 0xf8, 0xd8, 0xd8, 0xdc, 0xd8, 0xd8, 0xd8, 0xf8, 0x6e, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0xdb, 0xdb, 0xdf, 0xd8, 0xdb, 0x6e, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x88, 0x88, 0xf8, 0x88, 0x88, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00,
  0x00, 0xf8, 0x80, 0xe0, 0x80, 0x80, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x70, 0x88, 0x80, 0x88, 0x70, 0x00, 0x3c, 0x22, 0x3c, 0x24, 0x22, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x80, 0x80, 0x80, 0xf8, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00,
  0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44,
  0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa,
  0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
  0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f,
  0x00, 0x88, 0xc8, 0xa8, 0x98, 0x88, 0x00, 0x20, 0x20, 0x20, 0x20, 0x3e, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x88, 0x88, 0x50, 0x50, 0x20, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x0e, 0x38, 0xe0, 0x38, 0x0e, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xe0, 0x38, 0x0e, 0x38, 0xe0, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x06, 0x0c, 0xfe, 0x18, 0x30, 0xfe, 0x60, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x06, 0x1e, 0x7e, 0xfe, 0x7e, 0x1e, 0x06, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xc0, 0xf0, 0xfc, 0xfe, 0xfc, 0xf0, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x0c, 0xfe, 0x0c, 0x18, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x60, 0xfe, 0x60, 0x30, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x28, 0x6c, 0xfe, 0x6c, 0x28, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x06, 0x36, 0x66, 0xfe, 0x60, 0x30, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xfe, 0x6e, 0x6c, 0x6c, 0x6c, 0x6c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
  0x00, 0x00, 0x18, 0x3c, 0x3c, 0x3c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x66, 0x66, 0x66, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x10, 0x10, 0x7c, 0xd6, 0xd0, 0xd0, 0x7c, 0x16, 0x16, 0xd6, 0x7c, 0x10, 0x10,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xc2, 0xc6, 0x0c, 0x18, 0x30, 0x60, 0xc6, 0x86, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x38, 0x6c, 0x6c, 0x38, 0x76, 0xdc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x18, 0x0c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x30, 0x18, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3c, 0xff, 0x3c, 0x66, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7e, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x18, 0x30, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xce, 0xce, 0xd6, 0xd6, 0xe6, 0xe6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0x06, 0x06, 0x3c, 0x06, 0x06, 0x06, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x0c, 0x1c, 0x3c, 0x6c, 0xcc, 0xfe, 0x0c, 0x0c, 0x0c, 0x1e, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfe, 0xc0, 0xc0, 0xc0, 0xfc, 0x06, 0x06, 0x06, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x38, 0x60, 0xc0, 0xc0, 0xfc, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfe, 0xc6, 0x06, 0x06, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x06, 0x06, 0x0c, 0x78, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x18, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x0c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xde, 0xde, 0xde, 0xdc, 0xc0, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x66, 0x66, 0x66, 0x66, 0xfc, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xc0, 0xc0, 0xc2, 0x66, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xf8, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x6c, 0xf8, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68, 0x60, 0x62, 0x66, 0xfe, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xde, 0xc6, 0xc6, 0x66, 0x3a, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x1e, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0xcc, 0xcc, 0xcc, 0x78, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xe6, 0x66, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xf0, 0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0x62, 0x66, 0xfe, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xee, 0xfe, 0xfe, 0xd6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xe6, 0xf6, 0xfe, 0xde, 0xce, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x60, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xde, 0x7c, 0x0c, 0x0e,
0x00, 0x00,
  0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x6c, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x60, 0x38, 0x0c, 0x06, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x7e, 0x7e, 0x5a, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x6c, 0x38, 0x10, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0xee, 0x6c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xc6, 0xc6, 0x6c, 0x7c, 0x38, 0x38, 0x7c, 0x6c, 0xc6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xfe, 0xc6, 0x86, 0x0c, 0x18, 0x30, 0x60, 0xc2, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x3c, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x3c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x10, 0x38, 0x6c, 0xc6, 0x00, 0x00, 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, 0x30, 0x30, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0xe0, 0x60, 0x60, 0x78, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc0, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x1c, 0x0c, 0x0c, 0x3c, 0x6c, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x38, 0x6c, 0x64, 0x60, 0xf0, 0x60, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0xcc,
0x78, 0x00,
  0x00, 0x00, 0xe0, 0x60, 0x60, 0x6c, 0x76, 0x66, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x06, 0x06, 0x00, 0x0e, 0x06, 0x06, 0x06, 0x06, 0x06, 0x06, 0x66, 0x66,
0x3c, 0x00,
  0x00, 0x00, 0xe0, 0x60, 0x60, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0xe6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x70, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x34, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xec, 0xfe, 0xd6, 0xd6, 0xd6, 0xd6, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x60, 0x60,
0xf0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0x0c,
0x1e, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x76, 0x66, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0x60, 0x38, 0x0c, 0xc6, 0x7c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x10, 0x30, 0x30, 0xfc, 0x30, 0x30, 0x30, 0x30, 0x36, 0x1c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0x6c, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0x6c, 0x38, 0x38, 0x38, 0x6c, 0xc6, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x0c,
0xf8, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xcc, 0x18, 0x30, 0x60, 0xc6, 0xfe, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x0e, 0x18, 0x18, 0x18, 0x70, 0x18, 0x18, 0x18, 0x18, 0x0e, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x70, 0x18, 0x18, 0x18, 0x0e, 0x18, 0x18, 0x18, 0x18, 0x70, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x76, 0xdc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x66, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00,
0x00, 0x00,
};
```

11. usbkeyboard.c

```
#include "usbkeyboard.h"

#include <stdio.h>
#include <stdlib.h>

/* References on libusb 1.0 and the USB HID/keyboard protocol
 *
 * http://libusb.org
 * http://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/
 * http://www.usb.org/developers/devclass_docs/HID1_11.pdf
 * http://www.usb.org/developers/devclass_docs/Hut1_11.pdf
 */

/*
 * Find and return a USB keyboard device or NULL if not found
 * The argument con
 *
 */
struct libusb_device_handle *openkeyboard(uint8_t *endpoint_address) {
  libusb_device **devs;
  struct libusb_device_handle *keyboard = NULL;
  struct libusb_device_descriptor desc;
  ssize_t num_devs, d;
  uint8_t i, k;

  /* Start the library */
  if ( libusb_init(NULL) < 0 ) {
        fprintf(stderr, "Error: libusb_init failed\n");
        exit(1);
  }

  /* Enumerate all the attached USB devices */
  if ( (num_devs = libusb_get_device_list(NULL, &devs)) < 0 ) {
        fprintf(stderr, "Error: libusb_get_device_list failed\n");
        exit(1);
  }

  /* Look at each device, remembering the first HID device that speaks
        the keyboard protocol */

  for (d = 0 ; d < num_devs ; d++) {
        libusb_device *dev = devs[d];
        if ( libusb_get_device_descriptor(dev, &desc) < 0 ) {
        fprintf(stderr, "Error: libusb_get_device_descriptor failed\n");
        exit(1);
        }

        if (desc.bDeviceClass == LIBUSB_CLASS_PER_INTERFACE) {
        struct libusb_config_descriptor *config;
        libusb_get_config_descriptor(dev, 0, &config);
        for (i = 0 ; i < config->bNumInterfaces ; i++)
    for ( k = 0 ; k < config->interface[i].num_altsetting ; k++ ) {
      const struct libusb_interface_descriptor *inter =
        config->interface[i].altsetting + k ;
      if ( inter->bInterfaceClass == LIBUSB_CLASS_HID &&
        inter->bInterfaceProtocol == USB_HID_KEYBOARD_PROTOCOL) {
        int r;
        if ((r = libusb_open(dev, &keyboard)) != 0) {
        fprintf(stderr, "Error: libusb_open failed: %d\n", r);
        exit(1);
        }
        if (libusb_kernel_driver_active(keyboard,i))
        libusb_detach_kernel_driver(keyboard, i);
        //libusb_set_auto_detach_kernel_driver(keyboard, i);
        if ((r = libusb_claim_interface(keyboard, i)) != 0) {
        fprintf(stderr, "Error: libusb_claim_interface failed: %d\n", r);
        exit(1);
        }
        *endpoint_address = inter->endpoint[0].bEndpointAddress;
        goto found;
      }
    }
        }
  }

 found:
  libusb_free_device_list(devs, 1);

  return keyboard;
}
```

12. usbkeyboard.h

```
#ifndef _USBKEYBOARD_H
#define _USBKEYBOARD_H

#include <libusb-1.0/libusb.h>

#define USB_HID_KEYBOARD_PROTOCOL 1

/* Modifier bits */
#define USB_LCTRL  (1 << 0)
#define USB_LSHIFT (1 << 1)
#define USB_LALT   (1 << 2)
#define USB_LGUI   (1 << 3)
#define USB_RCTRL  (1 << 4)
#define USB_RSHIFT (1 << 5)
#define USB_RALT   (1 << 6)
#define USB_RGUI   (1 << 7)
```

```
struct usb_keyboard_packet {
  uint8_t modifiers;
  uint8_t reserved;
  uint8_t keycode[6];
};

/* Find and open a USB keyboard device.  Argument should point to
   space to store an endpoint address.  Returns NULL if no keyboard
   device was found. */
extern struct libusb_device_handle *openkeyboard(uint8_t *);

#endif
```

13. vga_led.h

```
#ifndef _VGA_LED_H
#define _VGA_LED_H

#include <linux/ioctl.h>

#define VGA_LED_DIGITS 14

typedef struct {
  unsigned char digit;    /* 0, 1, .. , VGA_LED_DIGITS - 1 */
  unsigned char segments; /* LSB is segment a, MSB is decimal point */
} vga_led_arg_t;

#define VGA_LED_MAGIC 'q'

/* ioctls and their arguments */
#define VGA_LED_WRITE_DIGIT _IOW(VGA_LED_MAGIC, 1, vga_led_arg_t *)
#define VGA_LED_READ_DIGIT  _IOWR(VGA_LED_MAGIC, 2, vga_led_arg_t *)

#endif
```

14. vga_led.c

```
/*
 * Device driver for the VGA LED Emulator
 *
 * A Platform device implemented using the misc subsystem
 *
 * Stephen A. Edwards
 * Columbia University
 *
 * References:
 * Linux source: Documentation/driver-model/platform.txt
 *             drivers/misc/arm-charlcd.c
 * http://www.linuxforu.com/tag/linux-device-drivers/
 * http://free-electrons.com/docs/
 *
 * "make" to build
 * insmod vga_led.ko
 *
 * Check code style with
 * checkpatch.pl --file --no-tree vga_led.c
 */

#include <linux/module.h>
#include <linux/init.h>
#include <linux/errno.h>
#include <linux/version.h>
#include <linux/kernel.h>
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/slab.h>
#include <linux/io.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/fs.h>
#include <linux/uaccess.h>
#include "vga_led.h"

#define DRIVER_NAME "vga_led"

/*
 * Information about our device
 */
struct vga_led_dev {
	struct resource res; /* Resource: our registers */
	void __iomem *virtbase; /* Where registers can be accessed in memory */
	u8 segments[VGA_LED_DIGITS];
} dev;

/*
 * Write segments of a single digit
 * Assumes digit is in range and the device information has been set up
 */
```

```
static void write_digit(int digit, u8 segments)
{
	iowrite8(segments, dev.virtbase + digit);
	dev.segments[digit] = segments;
}

/*
 * Handle ioctl() calls from userspace:
 * Read or write the segments on single digits.
 * Note extensive error checking of arguments
 */
static long vga_led_ioctl(struct file *f, unsigned int cmd, unsigned long arg)
{
	vga_led_arg_t vla;

	switch (cmd) {
	case VGA_LED_WRITE_DIGIT:
		if (copy_from_user(&vla, (vga_led_arg_t *) arg,
				   sizeof(vga_led_arg_t)))
			return -EACCES;
		if (vla.digit > 14)
			return -EINVAL;
		write_digit(vla.digit, vla.segments);
		break;

	case VGA_LED_READ_DIGIT:
		if (copy_from_user(&vla, (vga_led_arg_t *) arg,
				   sizeof(vga_led_arg_t)))
			return -EACCES;
		if (vla.digit > 14)
			return -EINVAL;
		vla.segments = dev.segments[vla.digit];
		if (copy_to_user((vga_led_arg_t *) arg, &vla,
				 sizeof(vga_led_arg_t)))
			return -EACCES;
		break;

	default:
		return -EINVAL;
	}

	return 0;
}

/* The operations our device knows how to do */
static const struct file_operations vga_led_fops = {
	.owner		= THIS_MODULE,
	.unlocked_ioctl = vga_led_ioctl,
};

/* Information about our device for the "misc" framework -- like a char dev */
static struct miscdevice vga_led_misc_device = {
	.minor		= MISC_DYNAMIC_MINOR,
	.name		= DRIVER_NAME,
	.fops		= &vga_led_fops,
};

/*
 * Initialization code: get resources (registers) and display
 * a welcome message
 */
static int __init vga_led_probe(struct platform_device *pdev)
{
	//static unsigned char welcome_message[VGA_LED_DIGITS] = {
	//	0x3E, 0x7D, 0x77, 0x08, 0x38, 0x79, 0x5E, 0x00};
	int i, ret;

	/* Register ourselves as a misc device: creates /dev/vga_led */
	ret = misc_register(&vga_led_misc_device);

	/* Get the address of our registers from the device tree */
	ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
	if (ret) {
		ret = -ENOENT;
		goto out_deregister;
	}

	/* Make sure we can use these registers */
	if (request_mem_region(dev.res.start, resource_size(&dev.res),
			       DRIVER_NAME) == NULL) {
		ret = -EBUSY;
		goto out_deregister;
	}

	/* Arrange access to our registers */
	dev.virtbase = of_iomap(pdev->dev.of_node, 0);
	if (dev.virtbase == NULL) {
		ret = -ENOMEM;
		goto out_release_mem_region;
	}

	/* Display a welcome message */
	//for (i = 0; i < VGA_LED_DIGITS; i++)
		//write_digit(i, welcome_message[i]);

	return 0;

out_release_mem_region:
	release_mem_region(dev.res.start, resource_size(&dev.res));
out_deregister:
	misc_deregister(&vga_led_misc_device);
	return ret;
}

/* Clean-up code: release resources */
static int vga_led_remove(struct platform_device *pdev)
{
	iounmap(dev.virtbase);
	release_mem_region(dev.res.start, resource_size(&dev.res));
	misc_deregister(&vga_led_misc_device);
	return 0;
}

/* Which "compatible" string(s) to search for in the Device Tree */
#ifdef CONFIG_OF
static const struct of_device_id vga_led_of_match[] = {
	{ .compatible = "altr,vga_led" },
	{},
};
MODULE_DEVICE_TABLE(of, vga_led_of_match);
#endif

/* Information for registering ourselves as a "platform" driver */
static struct platform_driver vga_led_driver = {
	.driver	= {
		.name	= DRIVER_NAME,
		.owner = THIS_MODULE,
		.of_match_table = of_match_ptr(vga_led_of_match),
	},
	.remove		= __exit_p(vga_led_remove),
};

/* Called when the module is loaded: set things up */
static int __init vga_led_init(void)
{
	pr_info(DRIVER_NAME ": init\n");
	return platform_driver_probe(&vga_led_driver, vga_led_probe);
}
```

```
/* Called when the module is unloaded: release resources */
static void __exit vga_led_exit(void)
{
	platform_driver_unregister(&vga_led_driver);
	pr_info(DRIVER_NAME ": exit\n");
}

module_init(vga_led_init);
module_exit(vga_led_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Stephen A. Edwards, Columbia University");
MODULE_DESCRIPTION("VGA 7-segment LED Emulator");
```