CECS 347 Spring Project 3

Space Invaders

By

Alarik Damrow & Pi Oliver

May 3, 2023

Space Invaders for the TM4C. Very basic version of Space Invaders was programmed for the board.

Introduction

In this project there was a LCD that is to be used with the TM4C to display a game. The game to be made was Space Invaders which is a old classic Atari game that has been around for a long time. The LCD will display space invaders while the board is responsible for all inputs that is given by the user.

Operation

To use the program there is only the need for a TM4C MCU along with an LCD that will show the game and what is next to be done in the game. A potentiometer is also used to move the player model either left or right depending on what the player wants. The TM4C will be programmed to handle the game operations and it is responsible for user inputs that will be taken via the two buttons on the MCU. One button will be responsible for starting the game when the start prompt shows up for the game while the other button will be the shoot button that when game is already initiated it will shoot laser as designed. The LCD will display all graphics based on buttons presses and the player will shoot down enemies. The Enemies are only programmed to reach the end of the screen and to die off if reached or if laser kills the enemy off. Also there is a speaker that was programed to play sounds when enemy has been killed and also when laser has been shot.

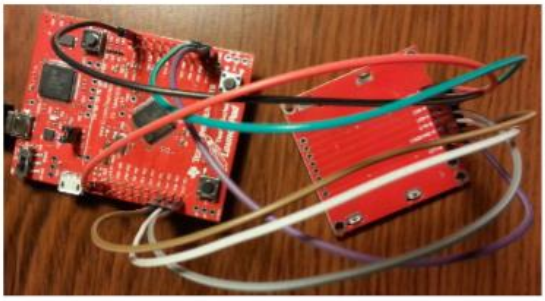
Video Links: **https://youtube.com/shorts/BxAhQ0veS5I?feature=share**

Theory

The main components of this lab were the TM4C along with the LCD and the potentiometer that will all act in unity to make the game happen. The TM4C will have a program that will run the game and it will also record the inputs via its buttons. Switch 2 will be responsible for starting the game when the game prompt is shown and request the player to press the button while Switch 1 will allow the player to shoot a laser during the game run and will only be able to be used in game and nothing else. LCD is connected to Port A on the TM4C to transmit the signals needed to work the LCD to show the game that was developed. The potentiometer is connected to Port E and will signal the TM4C where to place the player model based on the current given by the pot. The Speaker is responsible for play the sounds when the laser is shot and when the laser touches the enemy notifying that the enemy has been killed to the player.

Hardware Design

These are the hardware requirements



Be careful when connecting the backlight, at 3.3V, the back light draws 80 mA. If you want a dimmer back light connect 3.3V to a 100 ohm resistor, and the other end of the resistor to the **BL** pin.

Red SparkFun Nokia 5110 (LCD-10168)

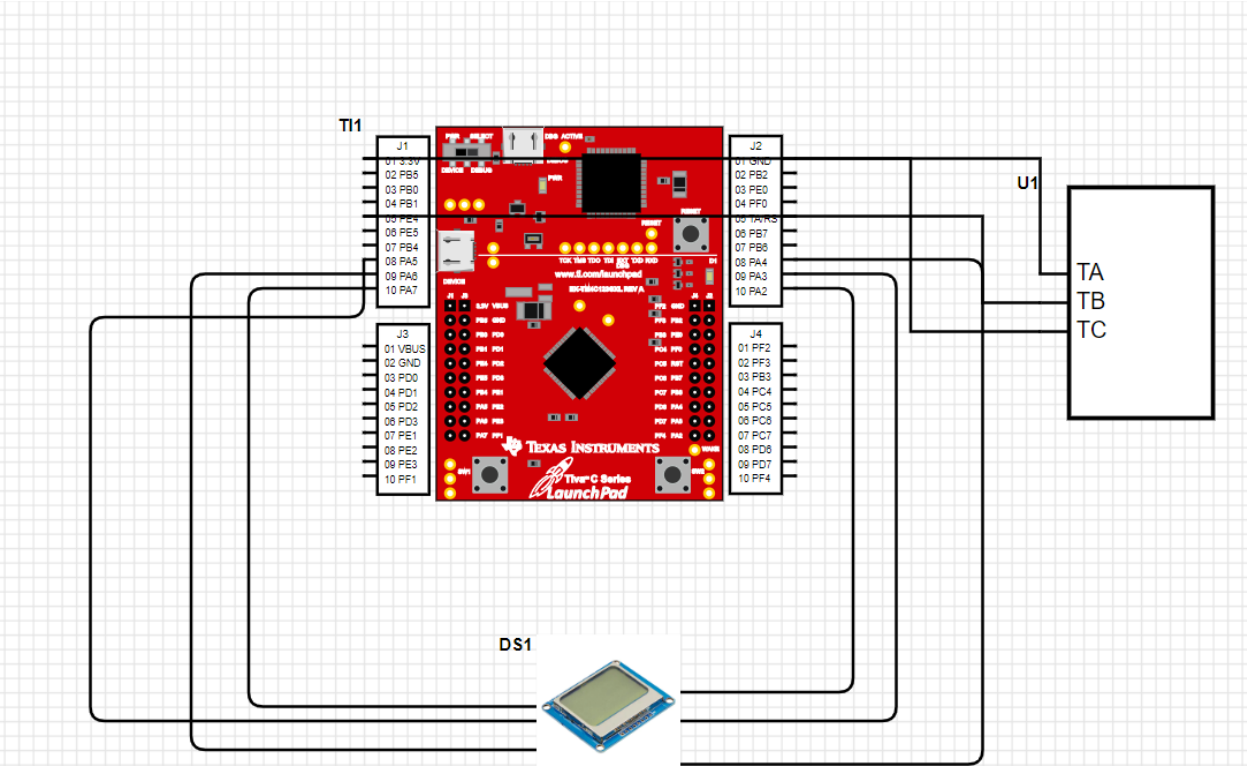
Signal	(Nokia 5110) LaunchPad pin
3.3V	(VCC, pin 1) power
Ground	(GND, pin 2) ground
SSI0Fss	(SCE, pin 3) connected to PA3
Reset	(RST, pin 4) connected to PA7
Data/Command	(D/C, pin 5) connected to PA6
SSI0Tx	(DN, pin 6) connected to PA5
SSI0Clk	(SCLK, pin 7) connected to PA2
back light	(LED, pin 8) not connected

Blue Nokia 5110

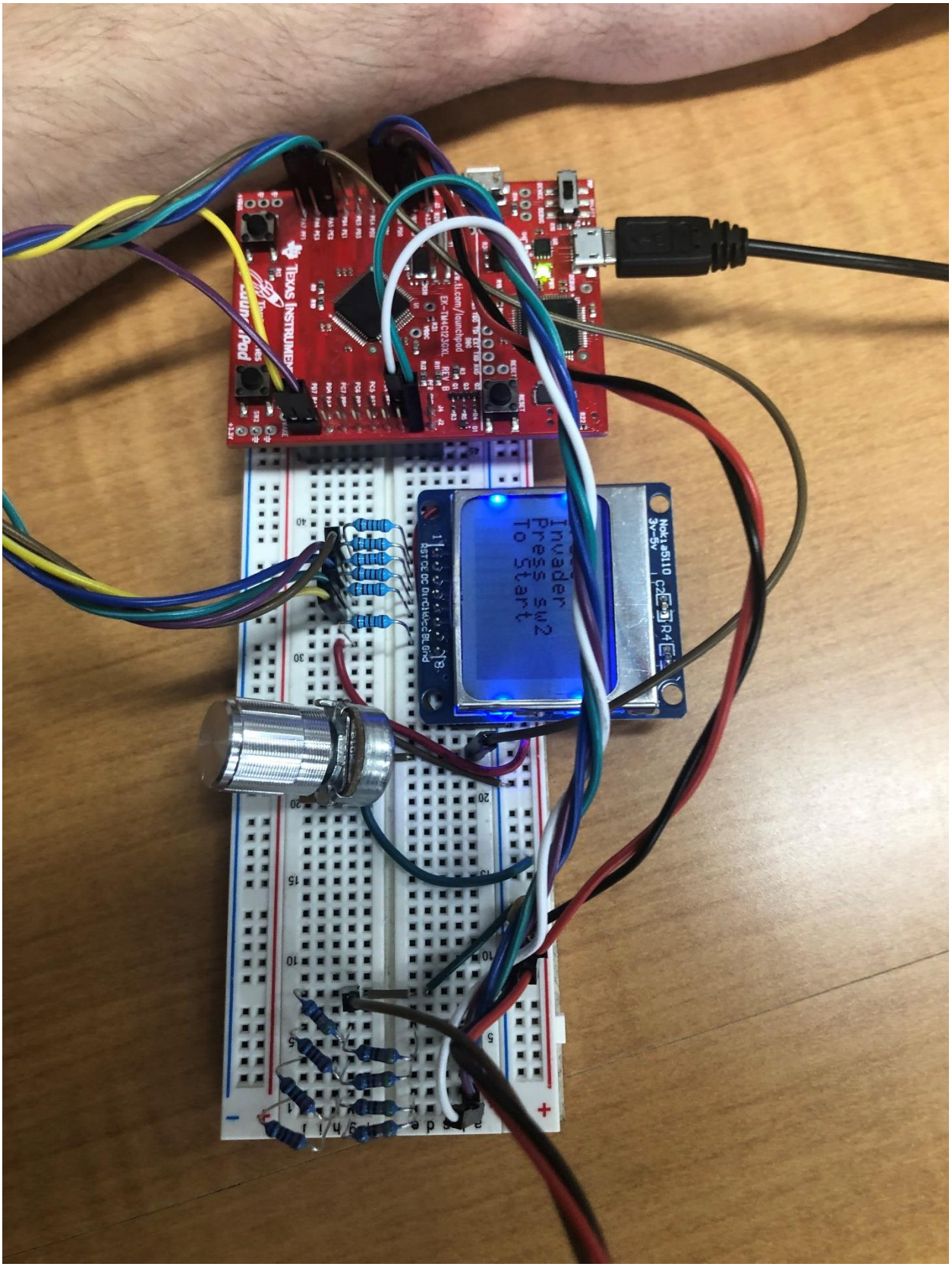
Signal	(Nokia 5110) LaunchPad pin
Reset	(RST, pin 1) connected to PA7
SSI0Fss	(CE, pin 2) connected to PA3
Data/Command	(DC, pin 3) connected to PA6
SSI0Tx	(Din, pin 4) connected to PA5
SSI0Clk	(Clk, pin 5) connected to PA2
3.3V	(Vcc, pin 6) power
back light	(BL, pin 7) not connected
Ground	(Gnd, pin 8) ground

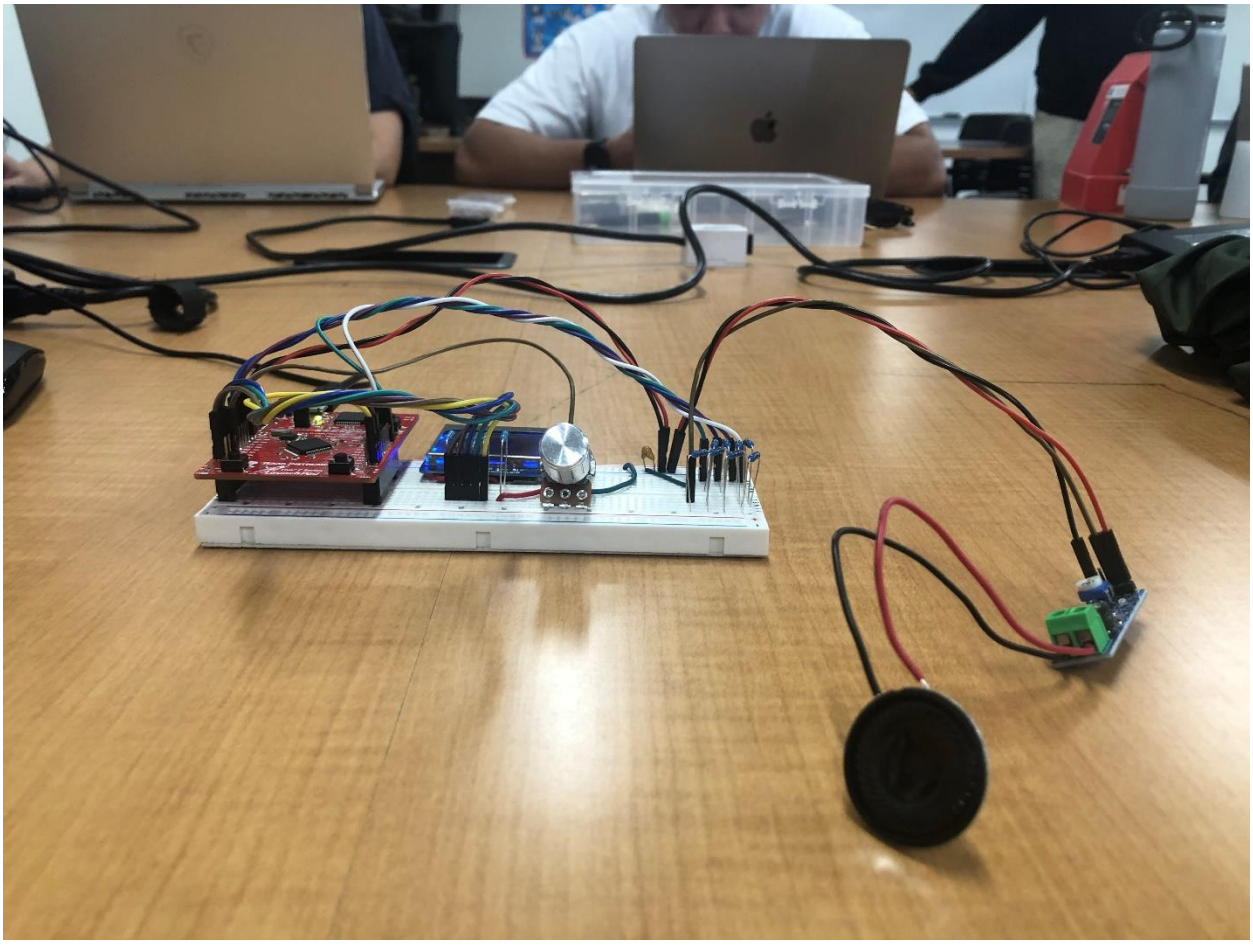
Version	Feature
SpaceInvadersV1.c (5 points)	Start with SpaceInvaders.c, comment out the code not implemented inside supper loop. Implement using PLL to generate an 80MHz system clock. Implement beginning prompt and game over prompt: press sw1 will display the beginning prompt, press sw2 will display the ending prompt. Modules implemented: main, Nokia5110, PLL, and switches. Submit a zip file named SpaceInvadersV1.zip with the following source code: SpaceInvadersV1.c, Nokia5110.c/h, PLL.c/h, and switches.c/h.
SpaceInvadersV2.c (5 points)	Use SysTick timer with interrupt enabled to control 10Hz screen refresh. The game will start with displaying the beginning prompt. Press sw2 to start the game. After the game is started, three different enemies will move from left to right with open and close arm/legs. Each step is 1 pixel to the right. When an enemy reaches the right edge, it will disappear (dead). After all three enemies are dead, display the game end message for 3 seconds (use Systick timer), then go back to display the beginning prompt. Modules implemented: main, Nokia5110, PLL, switch, and SysTick. Submit a zip file named SpaceInvadersV2.zip with the following source code: SpaceInvadersV2.c, Nokia5110.c/h, PLL.c/h, SysTick.c/h, and switches.c/h.
SpaceInvadersV3.c (5 points)	Implement spaceship movement: display spaceship at the bottom. Add ADC to take potentiometer input and covert to position of the spaceship. Modules implemented: main, Nokia5110, PLL, switch, SysTick, and ADC. Submit a zip file named SpaceInvadersV2.zip with the following source code: SpaceInvadersV2.c, Nokia5110.c/h, PLL.c/h, SysTick.c/h, ADC.c/h, and switches.c/h.

We implemented the above hardware requirements according to the schematic we developed. The schematic is listed below:



Hardware is viewable below:





Software Design

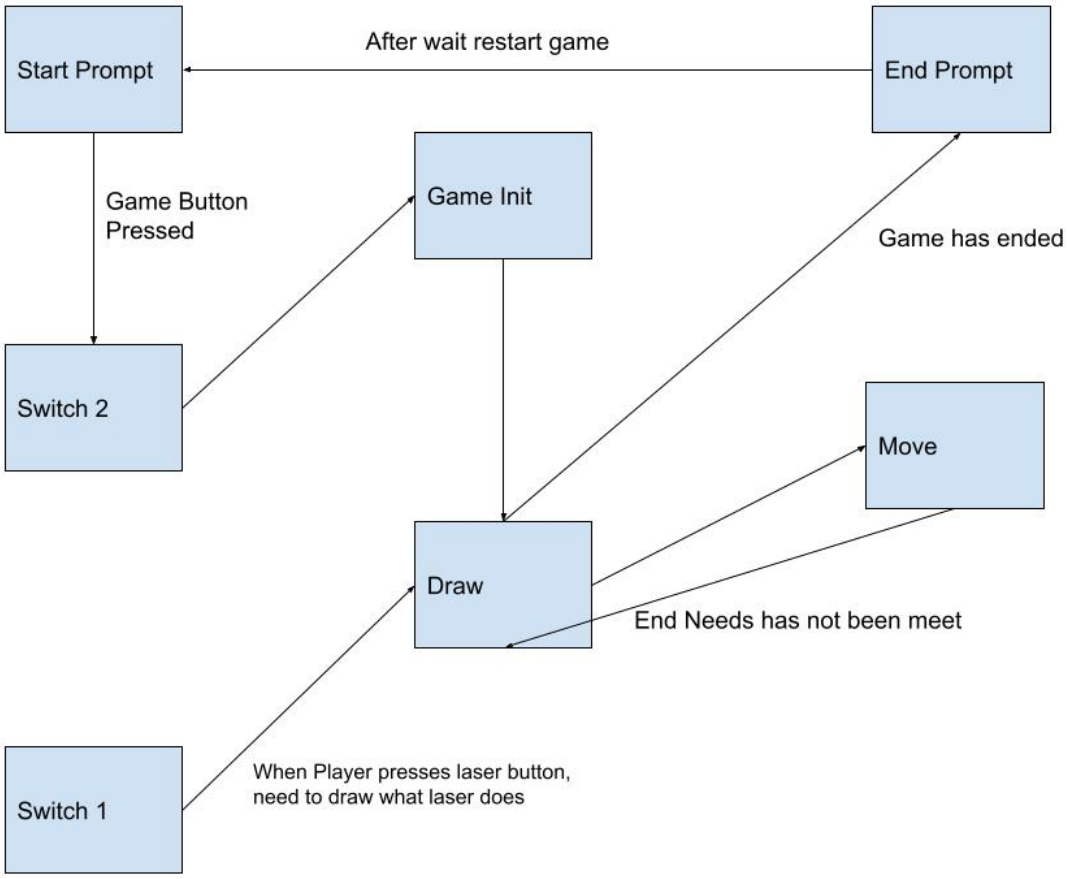
Our software design requirements were as follows:

Deliverable:

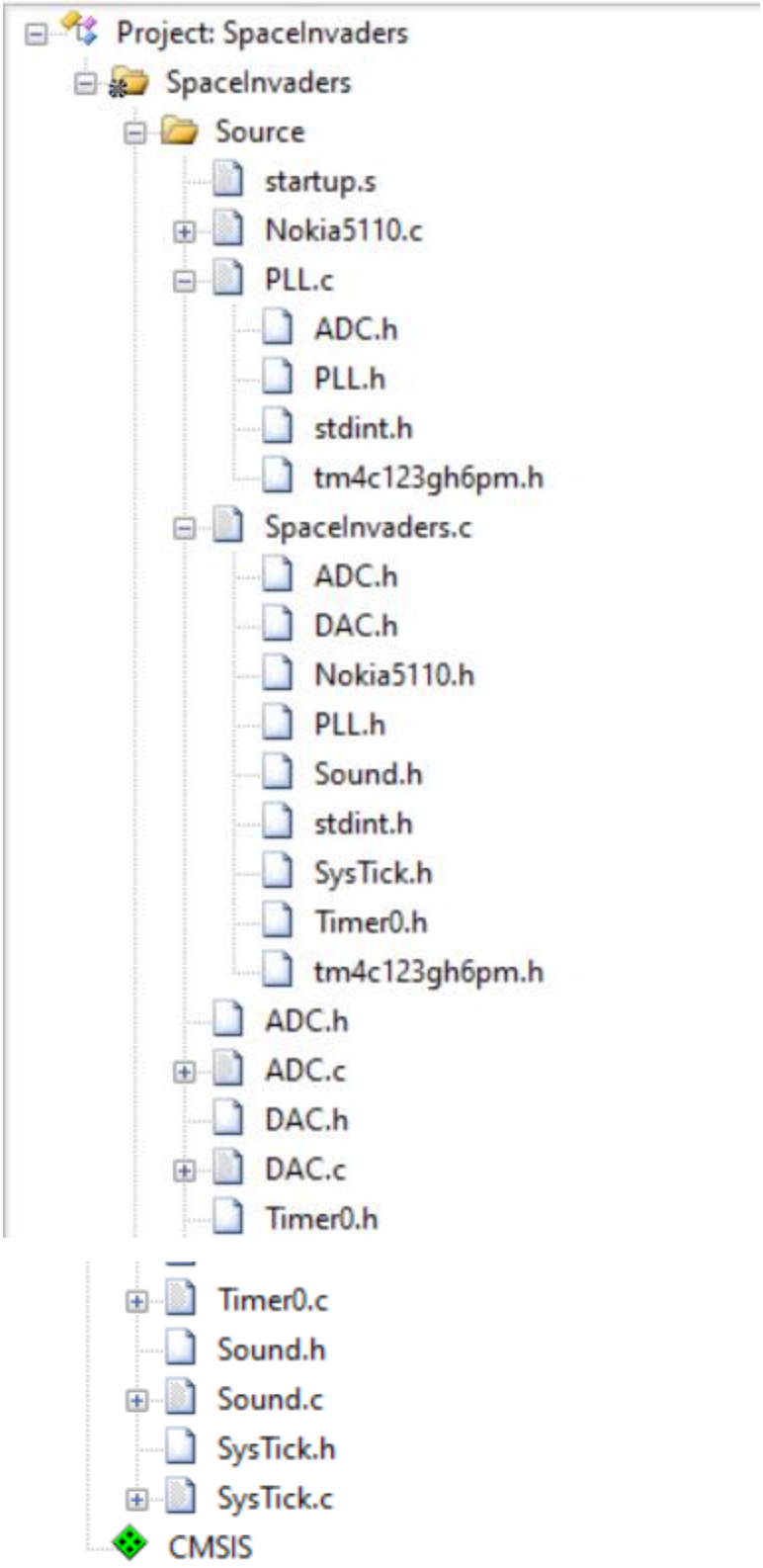
Demonstrate the required functionalities on your embedded system and submit the following items:

1. Project report including all required sections. If video link is provided, put it in Operation section, schematic and picture are required for hardware design section, flowchart of your final version is required for software design section.
2. Source code for this lab: 5 versions
   - Version 1: SpaceInvaderv1.c, Nokia5110.c/h, PLL.c/h, and switches.c/h.
   - Version 2: SpaceInvaderv2.c, Nokia5110.c/h, PLL.c/h, Systick.c/h.
   - Version 3: SpaceInvaderv3.c, ADC.c/h, Nokia5110.c/h, PLL.c/h, Systick.c/h.
   - Version 4: SpaceInvaderv4.c, ADC.c/h, Nokia5110.c/h, PLL.c/h, Systick.c/h.
   - Version 5: SpaceInvaderv5.c, DAC.c/h, sound.c/h, ADC.c/h, Nokia5110.c/h, PLL.c/h,

We implemented the requirements with the following flowchart:



We also implemented the above flowchart with the following modules: SpaceInvaders.c, DAC.c/h, sound.c/h, ADC.c/h, Nokia5110.c/h, PLL.c/h, and Systic.c/.h. Each module has its own respective header file associated with each modules, these header files provide the function prototypes for the modules as is common practice when programming in the C language. The modules relationships and dependencies are shown in the figure below:


Project: SpaceInvaders
SpaceInvaders
Source
startup.s
Nokia5110.c
PLL.c
ADC.h
PLL.h
stdint.h
tm4c123gh6pm.h
SpaceInvaders.c
ADC.h
DAC.h
Nokia5110.h
PLL.h
Sound.h
stdint.h
SysTick.h
Timer0.h
tm4c123gh6pm.h
ADC.h
ADC.c
DAC.h
DAC.c
Timer0.h
Timer0.c
Sound.h
Sound.c
SysTick.h
SysTick.c
CMSIS

The above diagram shows that PLL.c is the module that contains the clock cycles that is to be used throughout the program as needed. SpaceInvaders.c is the main program and it offers the means for the game to be played and also handles the operations to be done in game. ADC.c is responsible for reading sequences that are given by the pot to determine location of player model. Nokia5110.c is responsible for running the LCD that is to be used to show the game. DAC.c is responsible for setting up a DAC needed for the speakers to work. Timer0.c is used to set up a timer for the seven segment display that is to be implemented. Sound.c is used to give speakers sounds needed within the game. SysTick.c is used for a system interrupt that will determine certain actions to be done. Screenshots are shown of each modules in the following:

PLL.c: Clock cycle developer

```
// PLL.c
// Runs on LM4F120/TM4C123
// A software function to change the bus frequency using the PLL.
// Daniel Valvano
// 1/17/2020

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2015
   Program 2.10, Figure 2.37

 Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

#include <stdint.h>
#include "ADC.h"
#include "PLL.h"
#include "tm4c123gh6pm.h"

// The #define statement SYSDIV2 in PLL.h
// initializes the PLL to the desired frequency.
#define SYSDIV2 4
```

```
32
33  // bus frequency is 400MHz/(SYSDIV2+1) = 400MHz/(7+1) = 50 MHz
34  // see the table at the end of this file
35
36  #define SYSCTL_RIS_PLLLRIS      0x00000040  // PLL Lock Raw Interrupt Status
37  #define SYSCTL_RCC_XTAL_M       0x000007C0  // Crystal Value
38  #define SYSCTL_RCC_XTAL_6MHZ    0x000002C0  // 6 MHz Crystal
39  #define SYSCTL_RCC_XTAL_8MHZ    0x00000380  // 8 MHz Crystal
40  #define SYSCTL_RCC_XTAL_16MHZ   0x00000540  // 16 MHz Crystal
41  #define SYSCTL_RCC2_USERCC2     0x80000000  // Use RCC2
42  #define SYSCTL_RCC2_DIV400      0x40000000  // Divide PLL as 400 MHz vs. 200
43                                              // MHz
44  #define SYSCTL_RCC2_SYSDIV2_M   0x1F800000  // System Clock Divisor 2
45  #define SYSCTL_RCC2_SYSDIV2LSB  0x00400000  // Additional LSB for SYSDIV2
46  #define SYSCTL_RCC2_PWRDN2      0x00002000  // Power-Down PLL 2
47  #define SYSCTL_RCC2_BYPASS2     0x00000800  // PLL Bypass 2
48  #define SYSCTL_RCC2_OSCSRC2_M   0x00000070  // Oscillator Source 2
49  #define SYSCTL_RCC2_OSCSRC2_MO  0x00000000  // MOSC
50
51  // configure the system to get its clock from the PLL
52  // SYSDIV = 400/freq -1
53  // bus frequency is 400MHz/(SYSDIV+1)
54  void PLL_Init(uint32_t freq){
55    // 0) configure the system to use RCC2 for advanced features
56    //    such as 400 MHz PLL and non-integer System Clock Divisor
57    SYSCTL_RCC2_R |= SYSCTL_RCC2_USERCC2;
58    // 1) bypass PLL while initializing
59    SYSCTL_RCC2_R |= SYSCTL_RCC2_BYPASS2;
60    // 2) select the crystal value and oscillator source
61    SYSCTL_RCC_R &= ~SYSCTL_RCC_XTAL_M;   // clear XTAL field
62    SYSCTL_RCC_R += SYSCTL_RCC_XTAL_16MHZ;// configure for 16 MHz crystal
```

```
62    SYSCTL_RCC_R += SYSCTL_RCC_XTAL_16MHZ;// configure for 16 MHz crystal
63    SYSCTL_RCC2_R &= ~SYSCTL_RCC2_OSCSRC2_M;// clear oscillator source field
64    SYSCTL_RCC2_R += SYSCTL_RCC2_OSCSRC2_MO;// configure for main oscillator source
65    // 3) activate PLL by clearing PWRDN
66    SYSCTL_RCC2_R &= ~SYSCTL_RCC2_PWRDN2;
67    // 4) set the desired system divider and the system divider least significant bit
68    SYSCTL_RCC2_R |= SYSCTL_RCC2_DIV400;  // use 400 MHz PLL
69    SYSCTL_RCC2_R = (SYSCTL_RCC2_R&~0x1FC00000) // clear system clock divider field
70                    + (freq<<22);      // configure for 80 MHz clock
71    // 5) wait for the PLL to lock by polling PLLLRIS
72    while((SYSCTL_RIS_R&SYSCTL_RIS_PLLLRIS)==0){};
73    // 6) enable use of PLL by clearing BYPASS
74    SYSCTL_RCC2_R &= ~SYSCTL_RCC2_BYPASS2;
75  }
```

```
31
32 /*
33  SYSDIV2   Divisor   Clock (MHz)
34   0          1        reserved
35   1          2        reserved
36   2          3        reserved
37   3          4        reserved
38   4          5        80.000
39   5          6        66.667
40   6          7        reserved
41   7          8        50.000
42   8          9        44.444
43   9          10       40.000
44   10         11       36.364
45   11         12       33.333
46   12         13       30.769
47   13         14       28.571
48   14         15       26.667
49   15         16       25.000
50   16         17       23.529
51   17         18       22.222
52   18         19       21.053
53   19         20       20.000
54   20         21       19.048
55   21         22       18.182
56   22         23       17.391
57   23         24       16.667
58   24         25       16.000
59   25         26       15.385
60   26         27       14.815
61   27         28       14.286
62   28         29       13.793
63   29         30       13.333
64   30         31       12.903
```

65	31	32	12.500
66	32	33	12.121
67	33	34	11.765
68	34	35	11.429
69	35	36	11.111
70	36	37	10.811
71	37	38	10.526
72	38	39	10.256
73	39	40	10.000
74	40	41	9.756
75	41	42	9.524
76	42	43	9.302
77	43	44	9.091
78	44	45	8.889
79	45	46	8.696
80	46	47	8.511
81	47	48	8.333
82	48	49	8.163
83	49	50	8.000
84	50	51	7.843
85	51	52	7.692
86	52	53	7.547
87	53	54	7.407
88	54	55	7.273
89	55	56	7.143
90	56	57	7.018
91	57	58	6.897
92	58	59	6.780
93	59	60	6.667
94	60	61	6.557
95	61	62	6.452
96	62	63	6.349
97	63	64	6.250

98	64	65	6.154
99	65	66	6.061
100	66	67	5.970
101	67	68	5.882
102	68	69	5.797
103	69	70	5.714
104	70	71	5.634
105	71	72	5.556
106	72	73	5.479
107	73	74	5.405
108	74	75	5.333
109	75	76	5.263
110	76	77	5.195
111	77	78	5.128
112	78	79	5.063
113	79	80	5.000
114	80	81	4.938
115	81	82	4.878
116	82	83	4.819
117	83	84	4.762
118	84	85	4.706
119	85	86	4.651
120	86	87	4.598
121	87	88	4.545
122	88	89	4.494
123	89	90	4.444
124	90	91	4.396
125	91	92	4.348
126	92	93	4.301
127	93	94	4.255
128	94	95	4.211

```
 95        96        4.167
 96        97        4.124
 97        98        4.082
 98        99        4.040
 99        100       4.000
 100       101       3.960
 101       102       3.922
 102       103       3.883
 103       104       3.846
 104       105       3.810
 105       106       3.774
 106       107       3.738
 107       108       3.704
 108       109       3.670
 109       110       3.636
 110       111       3.604
 111       112       3.571
 112       113       3.540
 113       114       3.509
 114       115       3.478
 115       116       3.448
 116       117       3.419
 117       118       3.390
 118       119       3.361
 119       120       3.333
 120       121       3.306
 121       122       3.279
 122       123       3.252
 123       124       3.226
 124       125       3.200
 125       126       3.175
 126       127       3.150
 127       128       3.125
*/
```

SpaceInvaders.c: Responsible for the game that is to be displayed and played

```
// SpaceInvaders.c
// Runs on TM4C123
// Min He
// November 15, 2022

// Reference:
// http://www.spaceinvaders.de/
// sounds at http://www.classicgaming.cc/classics/spaceinvaders/sounds.php
// http://www.classicgaming.cc/classics/spaceinvaders/playguide.php
/* The original example code comes the books
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2013

   "Embedded Systems: Introduction to Arm Cortex M Microcontrollers",
   ISBN: 978-1469998749, Jonathan Valvano, copyright (c) 2013

 Copyright 2013 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

 // ******* Required Hardware I/O connections*******************
// Slide pot pin 1 connected to ground
```

```
// Slide pot pin 1 connected to ground
// Slide pot pin 2 connected to PE2/AIN1
// Slide pot pin 3 connected to pne side of the 1k resistor
// other side of the 1k resistor is connected to +3.3V
// Onboard sw1(left push button): fire button
// Onboard sw2(right push button): game start button

// Blue Nokia 5110
// ---------------
// Signal        (Nokia 5110) LaunchPad pin
// Reset         (RST, pin 1) connected to PA7
// SSI0Fss       (CE,  pin 2) connected to PA3
// Data/Command  (DC,  pin 3) connected to PA6
// SSI0Tx        (Din, pin 4) connected to PA5
// SSI0Clk       (Clk, pin 5) connected to PA2
// 3.3V          (Vcc, pin 6) power
// back light    (BL,  pin 7) not connected, consists of 4 white LEDs which draw ~80mA total
// Ground        (Gnd, pin 8) ground

// Red SparkFun Nokia 5110 (LCD-10168)
// -----------------------------------
// Signal        (Nokia 5110) LaunchPad pin
// 3.3V          (VCC, pin 1) power
// Ground        (GND, pin 2) ground
// SSI0Fss       (SCE, pin 3) connected to PA3
// Reset         (RST, pin 4) connected to PA7
// Data/Command  (D/C, pin 5) connected to PA6
// SSI0Tx        (DN,  pin 6) connected to PA5
// SSI0Clk       (SCLK, pin 7) connected to PA2
// back light    (LED, pin 8) not connected, consists of 4 white LEDs which draw ~80mA total
```

```

#include "tm4c123gh6pm.h"
#include "Nokia5110.h"
#include "PLL.h"
#include <stdint.h>
#include "ADC.h"
#include "Sound.h"
#include "DAC.h"
#include "Timer0.h"
#include "SysTick.h"  // for SysTick_Init()
//#include "Switches.h" // optional module for the two onboard switches

// enemy ship that starts at the top of the screen (arms/mouth closed)
// width=16 x height=10
const unsigned char SmallEnemyPointA[][200] = {{
// SmallEnemy30PointA
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0xF0, 0xF0, 0x00, 0x00, 0x00
 0xFF, 0x0F, 0xF0, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF},

 { // SmallEnemy20PointA
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0xFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x0F, 0xF0, 0x0F, 0xF0, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0, 0x0F, 0xFF, 0x00, 0x00, 0x00
 0xF0, 0x0F, 0xF0, 0x0F, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00
```


```
 0xF0, 0x0F, 0xF0, 0x0F, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00
```


```
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF},

 { // SmallEnemy10PointA
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x0F, 0xF0, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0xF0, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0xF0, 0xFF, 0xFF, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF}};


const unsigned char SmallEnemyPointB[][200] = {{
// enemy ship that starts at the top of the screen (arms/mouth open)
// width=16 x height=10, SmallEnemy30PointB
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x0F, 0x00, 0x00, 0xF0, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0xFF, 0xFF, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00
 0xFF, 0x0F, 0xF0, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF},

// enemy ship that starts in the middle of the screen (arms apart)
// width=16 x height=10, SmallEnemy20PointB
{
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x0F, 0xF0, 0x00, 0x0F, 0xF0, 0x0F, 0xF0, 0x0F, 0xF0, 0x00, 0x00, 0x00, 0xFF, 0xF0, 0x0F, 0xFF, 0x00, 0x00, 0x00
 0xF0, 0x0F, 0xF0, 0x0F, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF},

 // enemy ship that starts at the bottom of the screen (arms up)
// width=16 x height=10,SmallEnemy10PointB
{
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x00
 0xF0, 0xFF, 0xFF, 0x0F, 0xFF, 0x00, 0x00, 0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0x0F, 0x00, 0x00, 0xF0, 0x0F, 0x00, 0x00, 0xF0, 0x0F, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF}};


// image of the player's ship
// includes two blacked out columns on the left and right sides of the image to prevent smearing when moved 2 pixels to the left or right
// width=18 x height=8
const unsigned char PlayerShip0[] = {
 0x42, 0x4D, 0xD6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x12, 0x00, 0x00, 0x00, 0x08
 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0xC4, 0x0E, 0x00, 0x00, 0xC4, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x00
 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0A, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0A, 0xAA
 0x00, 0x00, 0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF


  // a missile in flight
```

```
// includes on+e blacked out row on the top, bottom, and right of the image to prevent smearing when moved 1 pixel down, up, or left
// width=4 x height=9
const unsigned char Missile0[] = {
 0x42, 0x4D, 0x9A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x09
 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00
};

  // a missile in flight
// includes one blacked out row on the top, bottom, and left of the image to prevent smearing when moved 1 pixel down, up, or right
// width=4 x height=9
const unsigned char Missile1[] = {
 0x42, 0x4D, 0x9A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x09
 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00
};

// blank space to cover a missile after it hits something
// width=4 x height=9
const unsigned char Missile2[] = {
 0x42, 0x4D, 0x9A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x09
 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

// a laser burst in flight
// includes one blacked out row on the top and bottom of the image to prevent smearing when moved 1 pixel up or down
// width=2 x height=9
const unsigned char Laser0[] = {
 0x42, 0x4D, 0x9A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x09
 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0xAA, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00
};

// blank space to cover a laser after it hits something
// width=2 x height=9
const unsigned char Laser1[] = {
 0x42, 0x4D, 0x9A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x09
 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

// small explosion best used for the demise of an enemy
// width=16 x height=10
const unsigned char SmallExplosion0[] = {
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x0F, 0x00, 0x0F, 0x00, 0xF0, 0x00, 0x00, 0x0F, 0x00, 0xF0, 0xF0, 0x0F, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x0F
 0xF0, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0xF0, 0x00, 0xF0, 0xF0, 0x00, 0xF0, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF};

// blank space following the small explosion for the demise of an enemy
// width=16 x height=10
const unsigned char SmallExplosion1[] = {
 0x42, 0x4D, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x0A
 0x00, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0xC0
 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
 0x00, 0x00, 0x00, 0x00, 0x00, 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};


// initial values for piano major tones:
// Assume system clock is 16MHz
const unsigned long Tone_Tab[] =
// Note name: C, D, E, F, G, A, B, C'
// Offset:0, 1, 2, 3, 4, 5, 6, 7
{30534*2,27211*2,24242*2,22923*2,20408*2,18182*2,16194*2,15289*2}; // C4 Major notes
//{15289*2,13621*2,12135*2,11454*2,10204*2,9091*2,8099*2,7645*2}; // C5 Major notes
//{7645*2,6810*2,6067*2,5727*2,5102*2,4545,4050*2,3822*2}; // C6 Major notes

#define NUM_VALs  16  // Assume 3-bit DAC is used, that will give 16 values for one period.
#define NUM_NOTEs 8   // number of notes to be played repeatedly

enum game_status{OVER,ON};
enum life_status{DEAD, ALIVE};
enum enemy_posture{CLOSE, OPEN};
```

```
235
236 #define PLAYERW     ((unsigned char)PlayerShip0[18])
237 #define PLAYERH     ((unsigned char)PlayerShip0[22])
238 #define ENEMY10W    16
239 #define LASERH      9
240 #define LASERW      2
241 #define BULLETH     LASERH
242 #define BULLETW     LASERW
243
244 #define ENEMYW  18
245 #define ENEMYH  10
246 #define MAX_ENEMYX    MAX_X-ENEMYW
247 #define MAX_ENEMYY    MAX_Y-ENEMYH
248 #define MAX_LASERY    LASERH
249 #define MAX_BULLETY   BULLETH
250
251 struct State {
252   unsigned long x;      // x coordinate
253   unsigned long y;      // y coordinate
254   const unsigned char *image; // ptr->image
255   long life;            // 0=dead, 1=alive
256 };
257 typedef struct State STyp;
258 STyp Enemy[4];
259 STyp PlayerShip;
260 STyp Bullet;
261 STyp Laser;
262 STyp SmallExplosion;
263
```

```
263
264 // Function prototypes
265 extern void EnableInterrupts(void);  // defined in startup.s
266 extern void DisableInterrupts(void); // defined in startup.s
267
268 void SysTick_Init(void);
269 void Delay100ms(unsigned long count);
270 void Game_Init(void);
271 void Move(void);
272 void Draw(void);
273 void Start_Prompt(void);
274 void End_Prompt(void);
275 void Switch_Init(void);
276 void System_Init(void);
277
278 // global variables used for game control
279 volatile uint8_t SW1_pressed = 0;
280 volatile uint8_t SW2_pressed = 0;
281 uint8_t time_to_draw=0;
282 uint8_t game_s=OVER;
283 uint8_t score=0;
284
285 int main(void){
286   ADC_Init();
287   System_Init();
288   DAC_Init();
289   Sound_Init();
290   while(1){
291     SysTick_Init();
292     Start_Prompt();
293     while(game_s == OVER){};
```

```
294       Game_Init(); // all sprites: 3 sprites
295       Draw();
296       while(game_s == ON){
297         if(time_to_draw){
298           Move();
299           Draw();
300           time_to_draw = 0;
301         }
302       }
303       End_Prompt();
304       SysTick_Wait1ms(3000);
305     }
306   }
307
308 void System_Init(void){
309     DisableInterrupts();
310     PLL_Init(Bus80MHz);                   // set system clock to 80 MHz
311     SysTick_Init();
312     Switch_Init();
313     ADC_Init();
314     Nokia5110_Init();
315     Nokia5110_ClearBuffer();
316     Nokia5110_DisplayBuffer();      // draw buffer
317     //game_s = OVER;
318     EnableInterrupts();
319   }
320
321   // Display the game start prompt
```

```
321  // Display the game start prompt
322 void Start_Prompt(void){
323    Nokia5110_Clear();
324    Nokia5110_SetCursor(0, 0);
325    Nokia5110_OutString("Space");
326    Nokia5110_SetCursor(0, 1);
327    Nokia5110_OutString("Invader");
328    Nokia5110_SetCursor(0, 2);
329    Nokia5110_OutString("Press sw2");
330    Nokia5110_SetCursor(0, 3);
331    Nokia5110_OutString("To Start");
332  }
333
334  // Display the game end prompt for 2 seconds
335 void End_Prompt(void){
336    Nokia5110_Clear();
337    Nokia5110_SetCursor(0, 0);
338    Nokia5110_OutString("Game Over");
339    Nokia5110_SetCursor(0, 1);
340    Nokia5110_OutString("Nice Try!");
341    Nokia5110_SetCursor(0, 2);
342    Nokia5110_OutString("Your Score");
343    Nokia5110_SetCursor(0, 3);
344    Nokia5110_OutUDec(score);
345  }
346
347  // Initialize the game: initialize all sprites and
348  // reset refresh control and game status.
```

```
// Initialize the game: initialize all sprites and
// reset refresh control and game status.
void Game_Init(void){
  time_to_draw = 0;
  //game_s = OVER;
  score = 0; // reset score

  // Version 2: add enemy initialization with close posture.
  uint8_t i;
  for(i=0;i<3;i++){
    Enemy[i].x = 20*i;
    Enemy[i].y = 10;
    Enemy[i].image = SmallEnemyPointA[i];
    Enemy[i].life = 1;
  }

  // Version 3: add player ship initialization
  PlayerShip.x = 42;
  PlayerShip.y = 40;
  PlayerShip.image = PlayerShip0;
  PlayerShip.life = 1;

  // Version 4: Add bullet initialization: you can choose Laser or Missile
  Laser.x = PlayerShip.x;
  Laser.y = PlayerShip.y + 8;
  Laser.image = Laser0;
  Laser.life = 0;

  // Explosion Init
  SmallExplosion.x = Laser.x;
  SmallExplosion.y = Laser.y;
```

```
  //Don't init explosion image yet
  SmallExplosion.life = 0;
  Laser.x = ((Convert(ADC_In())*55) >> 10) + PLAYERW/2;
}

// Update positions for all alive sprites.
void Move(void){
  uint8_t num_life = 0;
  uint8_t i;
  //shoot
  if(SW1_pressed){
    Laser.x = PlayerShip.x + PLAYERW/2;
    Laser.y = PlayerShip.y - PLAYERH + 1;
    Laser.life = 1;
    SW1_pressed = 0;
  }

  // Move Laser
  if(Laser.y > MAX_LASERY){
    Laser.y -= 2;
  }else{
    Laser.life = 0;
  }
  for(i=0;i<3;i++){
    num_life += 1;
    if(Enemy[i].x < MAX_ENEMYX){
      Enemy[i].x += 1;
        if(Enemy[i].x % 2 == 1){//controls flapping
                    Enemy[i].image = SmallEnemyPointB[i];
              }
```

```
          }
          else if(Enemy[i].x % 2 == 0){
                Enemy[i].image = SmallEnemyPointA[i];
          }
    }
    else if (Enemy[i].image == SmallExplosion0)
    {
      Enemy[i].life = 0;
    }
    else{
      Enemy[i].life = 0;
      num_life -= 1;
    }
    if((Laser.x+LASERW >= Enemy[i].x) && (Laser.x <= Enemy[i].x+ENEMYW-4) && (Laser.y <= Enemy[i].y+ENEMYH- 4) && (Laser.life == 1) && (
      Enemy[i].life = 1;
      score += 1;
      Enemy[i].image = SmallExplosion0;
      Sound_Explosion();
      Laser.life = 0;
    }
  }
  PlayerShip.x = (Convert(ADC_In())*55) >> 10;
  if (num_life==0) {
    game_s = OVER;
  }
}

// Update the screen:
// clear display and update the screen with the
// current positions of all sprites that are alive.
```

```
437 void Draw(void){
438   static uint8_t enemy_posture = CLOSE;  // enemy start with close posture: SmallEnemyPointA
439   uint8_t i;
440
441   if (game_s==OVER) return;
442
443   Nokia5110_ClearBuffer();
444   for(i=0;i<3;i++){
445     if(Enemy[i].life == ALIVE){
446         Nokia5110_PrintBMP(Enemy[i].x, Enemy[i].y, Enemy[i].image, 0);
447     }
448       if (Enemy[i].image == SmallExplosion0)
449       {
450         Enemy[i].life = 0;
451         Sound_Killed();
452       }
453       if(Enemy[i].x % 2 == 1){
454         enemy_posture = CLOSE;
455         Enemy[i].image = SmallEnemyPointA[i];
456       }else{
457         enemy_posture = OPEN;
458         Enemy[i].image = SmallEnemyPointB[i];
459     }
460   }
461   Nokia5110_PrintBMP(PlayerShip.x, PlayerShip.y, PlayerShip.image, 1);
462   if(Laser.life == ALIVE){
463     Nokia5110_PrintBMP(Laser.x, Laser.y, Laser.image, 1);
464   }
```

```
464   }
465   Laser.image = Laser0;
466   Nokia5110_DisplayBuffer();
467 }
468
469 // Control screen refresh rate.
470 void SysTick_Handler(void){
471   time_to_draw = 1;
472 }
473
474
475 void GPIOPortF_Handler(void){    // called on touch of either SW1 or SW2
476   if(GPIO_PORTF_RIS_R&0x01){ // SW2
477     GPIO_PORTF_ICR_R = 0x01;
478     SW2_pressed = 1;
479     time_to_draw = 1;
480     Sound_Highpitch();
481     game_s = ON;
482   }
483   if(GPIO_PORTF_RIS_R&0x10){
484     GPIO_PORTF_ICR_R = 0x10;
485     SW1_pressed = 1;
486     Sound_Shoot();
487   }
488 }
489
490 // Initialize the onboard two switches.
491 void Switch_Init(void){
492   volatile unsigned long delay;
493   SYSCTL_RCGC2_R |= 0x00000020;       //activate clock for Port F
```

```
494   while((SYSCTL_RCGC2_R&0x20) == 0){};
495   GPIO_PORTF_LOCK_R = 0x4C4F434B;     //unlock GPIO Port F
496   GPIO_PORTF_CR_R = 0x11;                        //allow changes for PF4 and PF0
497   GPIO_PORTF_DIR_R &= ~0x11;        //make onboard buttons PF4 and PF0 inputs
498   GPIO_PORTF_DEN_R |= 0x11;         //enable digital I/0 on PF4,0
499   GPIO_PORTF_PCTL_R &= ~0x000F000F; //configure as GPIO
500   GPIO_PORTF_AMSEL_R &= ~0x11;      //disable analog functionality on PF4,0
501   GPIO_PORTF_PUR_R |= 0x11;             //enable weak pull-up on PF4,0
502   GPIO_PORTF_IS_R &= ~0x11;             //PF4,PF0 is edge-sensitive
503   GPIO_PORTF_IBE_R &= ~0x11;            //PF4,PF0 is not both edges
504   GPIO_PORTF_IEV_R &= ~0x11;            //PF4,PF0 falling edge event
505   GPIO_PORTF_ICR_R = 0x11;              //clear flags 4,0
506   GPIO_PORTF_IM_R |= 0x11;              //arm interrupt on PF4,PF0
507   NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00400000; //set priority to 2
508   NVIC_EN0_R = 0x40000000;              //enable interrupt 30 in NVIC
509 }
```

ADC.c: Sets up a converter to take a analog signal to make a digital signal

```
1  // ADC.c
2  // Runs on LM4F120/TM4C123
3  // Provide functions that initialize ADC0
4  // Last Modified: 3/6/2015
5  // Student names: change this to your names or look very silly
6  // Last modification date: change this to the last modification date or look very silly
7
8  #include <stdint.h>
9  #include "tm4c123gh6pm.h"
10 #include "ADC.h"
11
12
13 // ADC initialization function
14 // Input: none
15 // Output: none
16
17 // Conversion for ADC
18 uint32_t Convert(uint32_t input){
19   return ((56*(input))+27891)/224;
20 }
21
22
23 void ADC_Init(void){
24 volatile unsigned long delay;
25   SYSCTL_RCGCGPIO_R |= 0x00000010;   // 1) activate clock for Port E
26   delay = SYSCTL_RCGCGPIO_R;         //    allow time for clock to stabilize
27    delay = SYSCTL_RCGCGPIO_R;
28    delay = SYSCTL_RCGCGPIO_R;
29    delay = SYSCTL_RCGCGPIO_R;
30   GPIO_PORTE_DIR_R &= ~0x04;      // 2) make PE2 input
31   GPIO_PORTE_AFSEL_R |= 0x04;     // 3) enable alternate function on PE2
```

```
  GPIO_PORTE_DEN_R &= ~0x04;      // 4) disable digital I/O on PE2
  GPIO_PORTE_AMSEL_R |= 0x04;     // 5) enable analog function on PE2
  SYSCTL_RCGCADC_R |= 0x01;   // 6) activate ADC0
  delay = SYSCTL_RCGCADC_R;
 delay = SYSCTL_RCGCADC_R;
 delay = SYSCTL_RCGCADC_R;
 delay = SYSCTL_RCGCADC_R;

  ADC0_PC_R = 0x01;           // 7) configure for 125K
  ADC0_SSPRI_R = 0x0123;          // 8) Sequencer 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger
  ADC0_SSMUX3_R = (ADC0_SSMUX3_R&0xFFFFFFF0)+1; // 11) channel Ain1 (PE2)
  ADC0_SSCTL3_R = 0x0006; // 12) no TS0 D0, yes IE0 END0
  ADC0_IM_R &= ~0x0008;
  ADC0_ACTSS_R |= 0x0008;         // 13) enable sample sequencer 3

}

//------------ADC_In------------
// Busy-wait Analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
uint32_t ADC_In(void){
 unsigned long result;
  ADC0_PSSI_R = 0x0008;            // 1) initiate SS3
  while((ADC0_RIS_R&0x08)==0){};   // 2) wait for conversion done
  result = ADC0_SSFIFO3_R&0xFFF;   // 3) read result
  ADC0 ISC R = 0x0008;             // 4) acknowledge completion
  return result;


}
```

DAC.c: Creates a converter for digital signals to be converted to analog signals for speakers

```c
// dac.c
// This software configures DAC output
// Lab 6 requires a minimum of 4 bits for the DAC, but you could have 5 or 6 bits
// Runs on LM4F120 or TM4C123
// Program written by: put your names here
// Date Created: 3/6/17
// Last Modified: 1/17/2020
// Lab number: 6
// Hardware connections
// TO STUDENTS "REMOVE THIS LINE AND SPECIFY YOUR HARDWARE********

#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "DAC.h"

#define DAC_DATA (*((volatile unsigned long *)0x4000503C))  // PB0-3: output to DAC circuit

// Code files contain the actual implemenation for public functions
// this file also contains an private functions and private data

// **************DAC_Init*********************
// Initialize 3-bit DAC, called once
// DAC bit 0 on PB0 (least significant bit)
// DAC bit 1 on PB1
// DAC bit 2 on PB2
// DAC bit 3 on PB3 (most significant bit)
// Input: none
// Output: none
void DAC_Init(void){
  unsigned long volatile delay;
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB; // activate port B
  delay = SYSCTL_RCGC2_R;    // allow time to finish activating

  GPIO_PORTB_AMSEL_R &= ~0x0F;      // no analog
  GPIO_PORTB_PCTL_R &= ~0x0000FFFF; // regular function
  GPIO_PORTB_DIR_R |= 0x0F;       // make PB3-0 out
  GPIO_PORTB_AFSEL_R &= ~0x0F;    // disable alt funct on PB3-0
  GPIO_PORTB_DEN_R |= 0x0F;       // enable digital I/O on PB3-0
}

// ***************DAC_Out**********************
// output to DAC
// Input: 4-bit data, 0 to 15
// Input=n is converted to n*3.3V/15
// Output: none
void DAC_Out(uint8_t data){
  DAC_DATA = data;
}
```

Timer0.c: Makes a timer based on board crystal

```c
// Timer0.c
// Runs on LM4F120/TM4C123
// Use TIMER0 in 32-bit periodic mode to request interrupts at a periodic rate
// Daniel Valvano
// Last Modified: 1/17/2020
// You can use this timer only if you learn how it works

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2015
  Program 7.5, example 7.6

 Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "Timer0.h"

void (*PeriodicTask0)(void);   // user function

// ****************** Timer0_Init ****************
// Activate TIMER0 interrupts to run user task periodically
// Inputs:  task is a pointer to a user function
//          period in units (1/clockfreq)
// Outputs: none
void Timer0_Init(void(*task)(void), uint32_t period){
  SYSCTL_RCGCTIMER_R |= 0x01;   // 0) activate TIMER0
  PeriodicTask0 = task;          // user function
  TIMER0_CTL_R = 0x00000000;    // 1) disable TIMER0A during setup
  TIMER0_CFG_R = 0x00000000;    // 2) configure for 32-bit mode
  TIMER0_TAMR_R = 0x00000002;   // 3) configure for periodic mode, default down-count settings
  TIMER0_TAILR_R = period-1;    // 4) reload value
  TIMER0_TAPR_R = 0;            // 5) bus clock resolution
  TIMER0_ICR_R = 0x00000001;    // 6) clear TIMER0A timeout flag
  TIMER0_IMR_R = 0x00000001;    // 7) arm timeout interrupt
  NVIC_PRI4_R = (NVIC_PRI4_R&0x00FFFFFF)|0x80000000; // 8) priority 4
// interrupts enabled in the main program after all devices initialized
// vector number 35, interrupt number 19
  NVIC_EN0_R = 1<<19;           // 9) enable IRQ 19 in NVIC
  TIMER0_CTL_R = 0x00000001;    // 10) enable TIMER0A
}

void Timer0A_Handler(void){
  TIMER0_ICR_R = TIMER_ICR_TATOCINT;// acknowledge TIMER0A timeout
  (*PeriodicTask0)();                // execute user task
}
```

Sound.c: The file that contains sounds to be played on speaker

```
// Sound.c
// Runs on any computer
// Sound assets based off the original Space Invaders and basic
// functions to play them.  Copy these functions and constants
// into your SpaceInvaders.c for ease of sharing your game!
// Jonathan Valvano
// November 19, 2012

#include "DAC.h"
#include "Timer0.h"
#include "Sound.h"
#include "tm4c123gh6pm.h"

const unsigned char shoot[4080] = {
  129, 99, 103, 164, 214, 129, 31, 105, 204, 118, 55, 92, 140, 225, 152, 61, 84, 154, 184, 101,
  75, 129, 209, 135, 47, 94, 125, 207, 166, 72, 79, 135, 195, 118, 68, 122, 205, 136, 64, 106,
  143, 173, 105, 54, 122, 200, 133, 74, 106, 215, 236, 91, 43, 84, 163, 115, 34, 81, 150, 209,
  184, 198, 191, 113, 95, 12, 0, 79, 167, 111, 120, 238, 255, 255, 118, 0, 0, 0, 0, 0,
  89, 236, 255, 255, 255, 255, 190, 0, 0, 0, 20, 170, 161, 137, 255, 255, 255, 181, 37, 0,
  0, 15, 0, 60, 167, 255, 255, 255, 255, 255, 60, 0, 0, 41, 60, 44, 128, 211, 255, 255,
  255, 150, 51, 12, 0, 0, 10, 136, 201, 208, 236, 255, 255, 209, 0, 0, 0, 15, 16, 31,
  128, 255, 255, 255, 255, 252, 144, 0, 0, 0, 77, 68, 75, 171, 255, 255, 255, 233, 139, 53,
  0, 0, 9, 106, 77, 123, 209, 255, 255, 255, 146, 68, 7, 0, 0, 0, 136, 166, 159, 219,
  255, 255, 255, 82, 0, 0, 0, 0, 0, 81, 214, 225, 229, 255, 255, 255, 115, 0, 0, 0,
  75, 63, 47, 126, 209, 255, 255, 255, 222, 119, 54, 0, 0, 0, 112, 133, 103, 185, 255, 255,
  255, 222, 136, 74, 0, 0, 0, 102, 96, 98, 177, 255, 255, 255, 255, 191, 119, 0, 0, 0,
  44, 125, 116, 122, 222, 255, 255, 255, 236, 98, 50, 0, 0, 0, 17, 129, 208, 164, 194, 255,
  255, 255, 198, 19, 0, 0, 0, 0, 40, 154, 205, 191, 209, 255, 255, 255, 147, 74, 20, 0,
  0, 0, 112, 140, 119, 180, 255, 255, 255, 236, 211, 122, 0, 0, 0, 33, 85, 81, 123, 207,
  255, 255, 255, 255, 255, 132, 0, 0, 0, 29, 31, 33, 115, 184, 255, 255, 248, 255, 255, 226,
  33, 0, 0, 0, 55, 44, 58, 128, 215, 255, 243, 255, 255, 255, 226, 36, 0, 0, 0, 12,
  31, 98, 170, 232, 225, 228, 255, 255, 255, 140, 39, 0, 0, 0, 0, 13, 96, 192, 192, 174,
  242, 255, 255, 255, 207, 50, 0, 0, 0, 0, 0, 70, 149, 232, 192, 191, 248, 255, 255, 236,
  140, 40, 0, 0, 0, 0, 24, 118, 208, 167, 163, 224, 255, 255, 255, 209, 137, 63, 10, 0,
  0, 0, 88, 120, 115, 153, 181, 253, 255, 255, 228, 239, 219, 87, 0, 0, 0, 0, 22, 54,
  130, 204, 200, 177, 238, 255, 255, 231, 208, 137, 75, 0, 0, 0, 0, 94, 105, 112, 178, 225,
  204, 215, 255, 255, 255, 201, 112, 44, 10, 0, 0, 0, 0, 91, 101, 115, 167, 214, 245, 232, 216,
  255, 255, 225, 150, 63, 23, 0, 0, 0, 40, 101, 113, 140, 198, 216, 198, 219, 255, 255, 233,
  180, 137, 81, 17, 0, 0, 0, 64, 78, 108, 154, 197, 240, 212, 207, 233, 255, 255, 209, 144,
  71, 36, 0, 0, 0, 22, 103, 106, 136, 170, 200, 248, 221, 212, 233, 255, 255, 195, 112, 37,
  6, 0, 0, 0, 27, 126, 128, 139, 173, 190, 224, 243, 221, 214, 233, 255, 221, 144, 54, 2,
  0, 0, 0, 0, 67, 154, 177, 161, 173, 171, 214, 250, 221, 211, 225, 243, 204, 122, 48, 9,
  3, 0, 0, 2, 75, 154, 167, 160, 177, 202, 218, 195, 195, 221, 249, 235, 188, 143, 105, 53,
  0, 0, 0, 15, 63, 79, 133, 166, 202, 215, 174, 160, 191, 240, 229, 205, 200, 207, 204, 109,
  20, 0, 0, 0, 0, 19, 89, 164, 174, 163, 168, 187, 216, 183, 171, 202, 211, 240, 212, 171,
  154, 113, 68, 0, 0, 0, 0, 0, 16, 54, 115, 152, 200, 214, 178, 157, 176, 201, 192, 177, 195,
  218, 208, 168, 136, 87, 55, 2, 0, 0, 0, 55, 77, 108, 152, 180, 211, 198, 166, 170, 187,
  195, 181, 185, 205, 218, 174, 149, 132, 91, 60, 0, 0, 0, 3, 64, 88, 108, 159, 198, 197,
  181, 167, 177, 194, 173, 159, 184, 190, 214, 202, 160, 153, 140, 103, 34, 0, 0, 3, 6, 19,
  71, 115, 166, 204, 184, 170, 183, 197, 166, 144, 152, 180, 204, 194, 174, 180, 185, 157, 132, 96,
  60, 36, 0, 0, 0, 24, 78, 105, 125, 156, 185, 202, 177, 168, 168, 181, 170, 142, 154, 180,
  201, 181, 167, 168, 171, 170, 128, 85, 44, 30, 16, 0, 3, 22, 58, 120, 137, 149, 173, 176,
  194, 185, 164, 156, 156, 170, 146, 136, 161, 185, 191, 171, 156, 163, 168, 142, 108, 84, 60, 17,
  0, 7, 29, 48, 55, 85, 128, 171, 183, 171, 178, 187, 178, 160, 147, 152, 160, 146, 133, 149,
  168, 188, 170, 154, 159, 159, 156, 130, 109, 96, 68, 23, 5, 13, 23, 50, 71, 81, 106, 146,
  168, 188, 191, 173, 171, 168, 174, 161, 139, 139, 147, 144, 132, 142, 159, 176, 174, 160, 153, 150,
  147, 150, 128, 109, 91, 53, 44, 34, 15, 24, 40, 70, 82, 96, 126, 149, 180, 184, 178, 173,
```

```
174, 177, 166, 144, 144, 149, 150, 137, 126, 129, 143, 149, 153, 156, 163, 168, 149, 139, 136, 135,
136, 118, 106, 92, 74, 46, 26, 31, 44, 55, 60, 75, 91, 119, 146, 156, 166, 173, 183, 183,
168, 161, 159, 159, 156, 142, 139, 143, 143, 132, 128, 130, 139, 144, 137, 143, 152, 163, 160, 147,
140, 133, 136, 129, 116, 112, 118, 112, 89, 68, 50, 46, 53, 53, 58, 67, 84, 105, 105, 113,
136, 156, 170, 171, 173, 176, 177, 167, 160, 153, 153, 152, 140, 139, 142, 140, 140, 130, 132, 129,
137, 133, 128, 128, 135, 140, 142, 143, 147, 150, 146, 136, 130, 129, 126, 128, 116, 111, 113, 120,
119, 87, 46, 13, 0, 6, 53, 96, 139, 161, 200, 255, 255, 174, 82, 58, 55, 105, 219, 208,
82, 71, 70, 112, 235, 204, 79, 70, 71, 120, 239, 207, 79, 68, 67, 118, 242, 204, 78, 65,
65, 112, 242, 207, 78, 64, 60, 109, 243, 211, 77, 58, 57, 103, 245, 216, 81, 60, 58, 99,
232, 225, 82, 53, 58, 88, 221, 236, 98, 53, 58, 81, 205, 250, 118, 51, 55, 72, 183, 255,
139, 51, 57, 63, 167, 255, 164, 55, 54, 57, 143, 255, 191, 58, 53, 55, 115, 255, 214, 75,
51, 55, 96, 232, 243, 96, 44, 54, 78, 200, 255, 133, 47, 57, 63, 163, 255, 177, 53, 55,
55, 122, 255, 215, 64, 47, 51, 98, 238, 245, 98, 46, 60, 75, 191, 255, 150, 46, 53, 55,
142, 255, 201, 58, 48, 50, 101, 243, 243, 91, 44, 51, 70, 190, 255, 152, 46, 54, 58, 129,
255, 212, 68, 43, 50, 92, 222, 255, 116, 43, 54, 61, 164, 255, 187, 51, 46, 53, 102, 248,
246, 94, 41, 54, 65, 176, 255, 174, 48, 46, 54, 113, 255, 236, 81, 40, 54, 72, 177, 255,
168, 47, 48, 58, 111, 252, 245, 89, 46, 51, 71, 176, 255, 180, 50, 46, 54, 101, 235, 255,
106, 41, 54, 64, 152, 255, 207, 63, 39, 51, 88, 202, 255, 140, 40, 48, 55, 120, 255, 240,
82, 40, 53, 68, 163, 255, 191, 51, 46, 50, 94, 214, 255, 136, 40, 53, 57, 120, 255, 242,
88, 41, 54, 70, 156, 255, 207, 60, 43, 53, 85, 190, 255, 166, 46, 47, 60, 101, 222, 255,
125, 41, 48, 61, 122, 255, 243, 95, 37, 51, 70, 144, 255, 221, 67, 41, 54, 75, 164, 255,
200, 55, 43, 53, 84, 184, 255, 177, 50, 46, 57, 95, 201, 255, 156, 47, 47, 60, 101, 216,
255, 139, 44, 50, 61, 112, 224, 255, 126, 44, 51, 65, 113, 229, 255, 122, 39, 48, 63, 115,
236, 255, 120, 47, 48, 64, 119, 231, 255, 118, 43, 50, 68, 116, 229, 255, 128, 46, 47, 67,
112, 219, 255, 135, 47, 53, 63, 108, 205, 255, 154, 50, 48, 61, 98, 192, 255, 170, 50, 50,
58, 92, 173, 255, 195, 57, 39, 46, 77, 149, 255, 208, 65, 34, 51, 78, 137, 255, 233, 79,
41, 50, 72, 120, 232, 255, 112, 43, 47, 67, 106, 201, 255, 160, 47, 47, 60, 92, 170, 255,
200, 63, 46, 55, 82, 142, 255, 242, 91, 48, 50, 75, 118, 212, 255, 146, 48, 47, 63, 101,
167, 255, 201, 64, 47, 57, 84, 136, 243, 253, 106, 44, 55, 71, 111, 188, 255, 177, 55, 47,
55, 91, 146, 252, 243, 92, 47, 54, 77, 113, 191, 255, 168, 54, 48, 63, 92, 144, 250, 243,
```

```
96, 51, 50, 74, 106, 184, 255, 185, 57, 48, 54, 87, 133, 233, 255, 116, 50, 51, 65, 105,
164, 255, 214, 67, 48, 53, 84, 118, 200, 255, 160, 55, 51, 58, 95, 140, 242, 252, 108, 48,
51, 67, 101, 159, 255, 216, 71, 50, 50, 81, 113, 180, 255, 187, 57, 47, 53, 88, 122, 205,
255, 153, 50, 48, 61, 96, 135, 226, 255, 123, 51, 53, 65, 98, 144, 242, 240, 102, 48, 48,
72, 103, 146, 255, 232, 84, 50, 53, 74, 103, 156, 255, 222, 84, 51, 54, 79, 106, 159, 255,
224, 78, 53, 54, 75, 106, 154, 255, 226, 85, 51, 54, 75, 105, 152, 255, 228, 89, 51, 54,
74, 106, 147, 245, 238, 106, 50, 50, 74, 103, 139, 231, 250, 126, 53, 53, 65, 101, 129, 208,
255, 147, 54, 53, 61, 91, 120, 191, 255, 177, 61, 47, 57, 87, 115, 164, 255, 211, 71, 54,
53, 78, 109, 146, 243, 240, 108, 55, 55, 65, 105, 128, 202, 255, 156, 58, 51, 58, 98, 115,
171, 255, 208, 74, 51, 53, 79, 112, 143, 228, 253, 128, 55, 51, 64, 102, 122, 181, 255, 195,
67, 48, 57, 89, 111, 143, 233, 249, 123, 58, 55, 68, 102, 119, 176, 255, 195, 72, 51, 58,
84, 112, 136, 216, 255, 150, 55, 51, 60, 92, 115, 159, 253, 229, 99, 54, 58, 72, 103, 123,
178, 255, 195, 68, 53, 60, 81, 113, 136, 207, 255, 164, 61, 51, 58, 89, 119, 144, 225, 255,
137, 58, 54, 64, 94, 119, 154, 239, 240, 112, 54, 58, 68, 103, 119, 156, 246, 224, 96, 46,
48, 65, 101, 118, 159, 246, 224, 94, 50, 55, 65, 106, 122, 159, 248, 221, 99, 51, 55, 68,
105, 126, 156, 243, 228, 108, 55, 51, 70, 101, 122, 154, 231, 242, 128, 55, 55, 65, 98, 119,
144, 215, 255, 152, 60, 54, 63, 91, 119, 136, 194, 255, 181, 64, 51, 63, 85, 109, 126, 171,
255, 218, 94, 57, 58, 71, 108, 122, 153, 225, 245, 140, 57, 51, 67, 94, 118, 136, 187, 255,
192, 74, 53, 58, 82, 109, 126, 154, 229, 246, 130, 57, 57, 67, 95, 118, 132, 183, 255, 197,
77, 54, 60, 78, 106, 125, 150, 214, 255, 153, 61, 54, 63, 88, 116, 126, 166, 242, 226, 112,
58, 50, 67, 96, 120, 136, 180, 255, 202, 79, 54, 55, 71, 102, 122, 142, 195, 255, 181, 68,
53, 60, 79, 108, 125, 144, 198, 255, 171, 67, 50, 57, 85, 108, 126, 144, 202, 255, 168, 65,
50, 58, 85, 112, 128, 146, 197, 253, 173, 67, 50, 57, 82, 105, 123, 139, 190, 253, 185, 72,
50, 58, 79, 108, 126, 137, 177, 249, 207, 92, 58, 60, 72, 99, 119, 133, 161, 229, 236, 130,
58, 51, 67, 94, 113, 126, 147, 198, 252, 174, 67, 50, 61, 84, 105, 128, 136, 164, 243, 219,
103, 58, 53, 67, 96, 116, 130, 146, 198, 249, 170, 68, 50, 61, 79, 111, 126, 133, 163, 231,
229, 125, 58, 50, 64, 95, 112, 129, 140, 177, 248, 202, 92, 58, 54, 72, 99, 118, 132, 147,
190, 249, 187, 75, 53, 57, 81, 106, 118, 133, 144, 200, 255, 176, 67, 54, 60, 81, 109, 123,
137, 143, 197, 252, 173, 71, 53, 60, 81, 106, 118, 137, 143, 190, 250, 187, 77, 54, 63, 77,
102, 119, 133, 144, 177, 240, 202, 99, 54, 54, 74, 98, 118, 132, 137, 164, 224, 232, 132, 60,
```

```
 48,  63,  94, 115, 126, 140, 152, 195, 249, 178,  72,  55,  55,  78, 109, 119, 135, 144, 168, 225,
228, 129,  63,  50,  63,  91, 112, 129, 139, 144, 188, 239, 200,  92,  60,  57,  71, 101, 119, 133,
143, 149, 197, 249, 180,  72,  53,  57,  70, 101, 113, 129, 136, 144, 197, 238, 167,  68,  50,  55,
 77, 102, 118, 133, 140, 150, 197, 240, 170,  70,  57,  60,  81, 102, 116, 132, 142, 143, 190, 240,
187,  87,  53,  57,  72,  99, 118, 132, 143, 143, 173, 229, 212, 120,  60,  50,  68,  91, 115, 126,
136, 140, 156, 198, 239, 168,  72,  55,  58,  79, 106, 120, 133, 139, 143, 168, 224, 218, 132,  60,
 48,  67,  91, 112, 125, 133, 143, 146, 176, 232, 204, 108,  55,  51,  71,  94, 113, 126, 139, 144,
144, 181, 232, 195, 103,  57,  53,  71,  95, 112, 129, 140, 140, 143, 174, 226, 204, 115,  57,  51,
 71,  94, 112, 125, 135, 143, 143, 164, 209, 224, 146,  64,  51,  60,  79, 108, 123, 133, 140, 140,
150, 184, 229, 187,  89,  54,  54,  72,  96, 116, 129, 139, 137, 142, 160, 197, 229, 156,  68,  51,
 55,  78, 106, 118, 130, 136, 140, 146, 161, 204, 221, 149,  64,  51,  63,  79, 105, 120, 129, 140,
142, 139, 160, 202, 221, 156,  64,  53,  60,  75, 102, 120, 130, 140, 144, 144, 154, 188, 224, 178,
 89,  55,  54,  70,  95, 116, 125, 137, 143, 142, 142, 164, 204, 216, 144,  63,  51,  63,  84, 102,
118, 130, 136, 140, 143, 147, 171, 214, 198, 122,  60,  48,  65,  88, 112, 118, 132, 139, 139, 139,
143, 168, 209, 202, 128,  60,  50,  65,  87, 105, 120, 132, 139, 139, 143, 143, 160, 198, 212, 152,
 70,  54,  60,  79, 101, 118, 130, 137, 137, 137, 142, 146, 174, 208, 194, 116,  57,  51,  71,  89,
111, 123, 128, 137, 140, 143, 147, 146, 177, 211, 184, 109,  57,  51,  70,  92, 111, 123, 136, 137,
137, 143, 143, 147, 171, 202, 198, 128,  61,  50,  63,  85, 108, 120, 132, 139, 140, 143, 143, 140,
152, 184, 208, 163,  89,  54,  54,  74,  96, 112, 128, 132, 140, 139, 143, 142, 140, 157, 184, 205,
159,  84,  58,  53,  72,  98, 116, 129, 139, 143, 142, 146, 140, 137, 147, 176, 201, 180, 111,  57,
 51,  70,  92, 111, 125, 132, 140, 142, 144, 142, 140, 139, 154, 183, 205, 157,  84,  54,  51,  70,
 92, 106, 125, 133, 136, 142, 140, 137, 136, 135, 144, 180, 200, 163,  98,  55,  50,  75,  89, 112,
125, 133, 139, 140, 143, 140, 139, 137, 135, 157, 188, 195, 144,  75,  55,  55,  81,  99, 115, 132,
137, 143, 142, 142, 139, 140, 136, 136, 153, 184, 197, 153,  84,  57,  55,  77,  98, 115, 128, 137,
142, 140, 144, 139, 136, 137, 139, 142, 161, 185, 185, 139,  74,  54,  64,  79, 102, 116, 129, 139,
139, 140, 144, 139, 140, 136, 135, 136, 150, 176, 192, 161,  96,  58,  53,  77,  96, 115, 123, 132,
142, 139, 139, 142, 142, 135, 137, 135, 130, 150, 177, 188, 160, 101,  58,  54,  72,  91, 111, 122,
135, 137, 143, 142, 142, 139, 136, 137, 135, 130, 133, 153, 176, 185, 149,  87,  61,  55,  77,  94,
112, 129, 135, 139, 139, 142, 140, 139, 136, 137, 135, 132, 128, 137, 161, 178, 183, 132,  75,  53,
```

```
 60,  81, 101, 118, 130, 136, 140, 139, 143, 140, 135, 136, 137, 130, 129, 132, 128, 140, 167, 181,
167, 115,  70,  54,  67,  85, 108, 119, 136, 136, 142, 142, 139, 142, 140, 132, 135, 132, 130, 130,
126, 128, 147, 168, 183, 156, 106,  68,  57,  65,  92, 106, 123, 132, 136, 142, 140, 140, 137, 139,
133, 129, 133, 130, 125, 130, 130, 126, 146, 167, 176, 160, 113,  71,  55,  65,  87, 105, 119, 133,
139, 140, 140, 137, 140, 139, 130, 133, 130, 129, 132, 132, 128, 126, 130, 139, 154, 167, 174, 142,
 95,  67,  57,  72,  95, 111, 126, 132, 140, 142, 140, 140, 139, 136, 137, 130, 135, 132, 128, 128,
128, 126, 130, 126, 132, 150, 163, 173, 161, 118,  79,  60,  67,  84, 102, 120, 129, 135, 140, 140,
142, 142, 140, 132, 135, 130, 130, 133, 129, 128, 126, 130, 130, 126, 125, 130, 137, 156, 163, 170,
146, 102,  74,  58,  72,  88, 109, 122, 130, 139, 140, 140, 139, 137, 136, 137, 130, 135, 132, 128,
129, 128, 132, 128, 126, 128, 125, 130, 130, 128, 135, 150, 159, 163, 163, 125,  91,  65,  63,  75,
 95, 115, 123, 132, 140, 142, 143, 140, 139, 130, 132, 129, 128, 128, 123, 129, 129, 125, 123, 126,
126, 126, 125, 129, 125, 125, 125, 126, 135, 143, 154, 159, 157, 135, 102,  78,  61,  71,  89, 106,
120, 129, 137, 139, 143, 142, 142, 139, 139, 136, 132, 128, 133, 128, 128, 132, 128, 126, 132, 126,
126, 130, 128, 126, 126, 126, 128, 128, 128, 125, 129, 130, 139, 152, 152, 156, 150, 126,  99,  82,
 65,  70,  89, 106, 120, 133, 139, 140, 143, 140, 139, 137, 139, 132, 136, 133, 129, 128, 133, 129,
128, 132, 128, 126, 125, 130, 128, 132, 128, 126, 130, 126, 126, 126, 125, 125, 125, 126, 128, 128,
128, 128, 128, 129, 142, 144, 150, 147, 144, 133, 108,  91,  79,  70,  77,  96, 106, 123, 133, 139,
144, 140, 140, 139, 137, 139, 132, 153, 176, 119, 102, 113, 113, 116, 130, 166, 122,  99, 130, 113,
132, 166, 118, 103, 130, 111, 136, 166, 115, 105, 129, 111, 132, 160, 120, 102, 128, 115, 130, 160,
120, 101, 129, 109, 130, 163, 119, 105, 128, 109, 130, 164, 116, 101, 129, 111, 130, 164, 116, 108,
129, 111, 128, 159, 119, 112, 129, 111, 125, 157, 120, 116, 128, 106, 126, 150, 130, 123, 120, 111,
120, 139, 146, 126, 113, 116, 119, 133, 153, 126, 106, 120, 113, 132, 160, 122, 102, 128, 109, 126,
163, 115, 111, 129, 109, 126, 154, 125, 119, 123, 111, 120, 139, 146, 126, 109, 120, 116, 133, 164,
118, 102, 128, 109, 126, 163, 119, 108, 129, 111, 125, 149, 130, 123, 118, 115, 122, 132, 159, 128,
108, 125, 111, 132, 164, 118, 109, 129, 111, 125, 150, 132, 128, 119, 112, 119, 132, 159, 123, 106,
128, 111, 128, 159, 120, 118, 125, 112, 122, 139, 150, 126, 111, 125, 115, 129, 164, 116, 112, 128,
111, 125, 144, 142, 128, 113, 118, 116, 130, 164, 120, 106, 126, 115, 126, 146, 139, 123, 113, 122,
116, 130, 166, 118, 106, 129, 112, 125, 144, 143, 129, 116, 119, 118, 130, 160, 116, 116, 125, 116,
126, 133, 154, 128, 109, 126, 116, 128, 153, 126, 120, 120, 118, 120, 132, 166, 122, 109, 126, 112,
120, 136, 139, 125, 109, 115, 118, 123, 154, 122, 116, 120, 112, 122, 129, 163, 119, 108, 123, 113,
```

```
  125, 136, 152, 126, 112, 122, 118, 122, 150, 132, 125, 119, 116, 116, 130, 157, 119, 118, 123, 118,
  122, 132, 159, 120, 109, 125, 115, 125, 137, 156, 123, 111, 123, 118, 126, 139, 142, 128, 116, 119,
  118, 128, 147, 132, 122, 115, 118, 123, 128, 152, 128, 120, 125, 116, 116, 132, 153, 120, 120, 123,
  118, 122, 128, 157, 122, 119, 125, 119, 123, 129, 159, 116, 116, 126, 119, 123, 129, 159, 116, 116,
  126, 119, 120, 132, 159, 116, 118, 125, 119, 120, 132, 154, 118, 118, 123, 122, 123, 129, 153, 120,
  120, 123, 118, 122, 128, 152, 120, 125, 125, 116, 123, 126, 144, 130, 120, 120, 113, 122, 126, 144,
  137, 122, 118, 118, 123, 120, 135, 147, 125, 119, 119, 122, 120, 132, 154, 122, 115, 123, 118, 119,
  132, 154, 119, 113, 123, 115, 126, 132, 146, 122, 125, 122, 115, 122, 125, 137, 143, 126, 119, 119,
  125, 120, 132, 154, 122, 115, 126, 123, 122, 129, 153, 122, 119, 125, 119, 123, 126, 139, 142, 123,
  119, 119, 125, 118, 132, 154, 118, 118, 125, 116, 126, 126, 140, 132, 125, 123, 116, 123, 120, 132,
  154, 122, 113, 122, 118, 122, 128, 146, 126, 123, 122, 115, 126, 120, 132, 153, 116, 116, 123, 118,
  122, 128, 142, 129, 123, 125, 116, 123, 120, 132, 153, 116, 116, 128, 116, 126, 126, 139, 137, 122,
  120, 119, 122, 120, 132, 149, 120, 125, 125, 116, 129, 122, 129, 153, 122, 119, 125, 119, 128, 123,
  136, 144, 128, 120, 118, 123, 123, 129, 142, 126, 122, 125, 119, 129, 118, 132, 149, 119, 123, 125,
  119, 129, 122, 129, 153, 120, 120, 125, 119, 129, 128, 129, 150, 122, 122, 125, 120, 129, 125, 132,
  139, 122, 120, 119, 125, 125, 129, 140, 137, 128, 120, 120, 126, 126, 126, 137, 137, 123, 122, 125,
  125, 126, 129, 135, 137, 128, 120, 120, 123, 128, 128, 136, 137, 126, 120, 125, 125, 125, 125, 136,
  140, 128, 120, 125, 125, 125, 125, 132, 144, 128, 115, 119, 116, 122, 123, 129, 144, 118, 115, 125,
  118, 126, 120, 130, 144, 118, 122, 128, 116, 128, 122, 129, 140, 120, 125, 125, 119, 128, 123, 129,
  136, 132, 125, 125, 122, 123, 128, 128, 129, 143, 123, 122, 125, 123, 128, 125, 125, 142, 120, 125,
  125, 123, 128, 128, 128, 132, 136, 126, 122, 125, 123, 128, 125, 129, 143, 118, 125, 125, 123, 128,
  128, 128, 132, 136, 126, 122, 125, 123, 128, 128, 129, 142, 119, 125, 125, 123, 128, 128, 128, 129,
  137, 126, 122, 125, 125, 125, 126, 129, 142, 122, 126, 125, 123, 128, 125, 128, 129, 142, 119, 123,
  125, 123, 123, 128, 128, 129, 137, 125, 125, 125, 123, 128, 128, 128, 132, 130, 123, 125, 125, 125,
  126, 126, 126, 136, 126, 125, 125, 123, 123, 128, 128, 128, 137, 120, 123, 123, 123, 125, 129, 128,
  128, 137, 120, 125, 125, 125, 123, 128, 128, 128, 136, 123, 123, 128, 119, 126, 128, 128, 128, 136,
  123, 123, 128, 123, 123, 123, 123, 129, 136, 123, 125, 125, 119, 128, 128, 123, 130, 135, 129, 130,
  120, 125, 125, 125, 125, 125, 129, 135, 125, 125, 125, 125, 123, 128, 128, 128, 137, 120, 123, 128,
  125, 128, 128, 125, 125, 132, 125, 125, 125, 125, 126, 126, 126, 126, 126, 136, 126, 120, 126, 125,
  123, 128, 128, 128, 136, 123, 125, 125, 123, 128, 123, 128, 128, 128, 133, 126, 125, 125, 125, 123,
```

```
  123, 128, 128, 128, 136, 123, 125, 125, 123, 128, 123, 128, 128, 128, 133, 126, 125, 125, 125, 123,
  128, 123, 128, 133, 123, 125, 125, 123, 128, 128, 128, 128, 128, 135, 123, 125, 125, 125, 125, 126,
  126, 126, 130, 135, 125, 125, 125, 125, 125, 126, 126, 126, 130, 130, 125, 125, 125, 125, 126, 126,
  126, 126, 130, 129, 125, 125, 125, 125, 126, 126, 126, 126, 132, 125, 125, 125, 125, 126, 126, 126,
  126, 130, 130, 123, 129, 125, 125, 129, 125, 125, 125, 125, 129, 129, 125, 125, 125, 125, 126, 126,
  126, 130, 129, 130, 126, 125, 125, 125, 125, 126, 129, 125, 129, 135, 125, 125, 129, 125, 125, 125,
  129, 125, 129, 130, 123, 125, 129, 125, 129, 125, 125, 125, 129, 129, 130, 123, 125, 125, 125, 126,
  126, 126, 122, 129, 128, 123, 123, 123, 122, 126, 122, 126, 128, 123, 128, 128, 128, 123, 128, 123,
  128, 128, 128, 128, 128, 129, 129, 125, 129, 125, 125, 125, 125, 126, 126, 126, 130, 125, 125, 125,
  126, 126, 126, 126, 128, 128, 128, 129, 125, 125, 125, 125, 126, 126, 126, 126, 126, 130, 129, 125,
  125, 125, 125, 130, 125, 125, 129, 125, 129, 129, 125, 125, 125, 125, 126, 126, 126, 130, 125, 125,
  129, 125, 125, 125, 125, 130, 125, 125, 125, 125, 130, 125, 129, 125, 129, 129, 125, 125, 129, 125,
  129, 125, 129, 129, 125, 125, 125, 125, 129, 125, 125, 125, 126, 128, 128, 129, 125, 129, 125, 125};

const unsigned char invaderkilled[3377] = {
  128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
  128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
  128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
  128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
  128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 144, 164, 145, 111,
  95, 91, 100, 106, 126, 134, 108, 109, 123, 160, 158, 118, 101, 144, 189, 110, 79, 154, 202, 117,
  16, 96, 208, 187, 65, 100, 192, 175, 93, 0, 106, 253, 137, 17, 140, 242, 103, 1, 134, 255,
  134, 4, 110, 230, 118, 1, 149, 255, 89, 6, 160, 255, 110, 0, 60, 177, 255, 81, 27, 207,
  170, 43, 61, 200, 179, 23, 104, 245, 112, 11, 102, 210, 217, 39, 73, 237, 114, 29, 57, 173,
  255, 91, 0, 145, 255, 138, 10, 62, 171, 252, 84, 0, 172, 244, 46, 47, 207, 190, 74, 0,
  141, 255, 147, 16, 65, 191, 215, 30, 84, 224, 163, 78, 0, 128, 255, 124, 0, 149, 255, 102,
  0, 138, 249, 169, 63, 0, 166, 255, 62, 24, 195, 229, 33, 78, 224, 163, 70, 0, 145, 255,
  100, 0, 161, 255, 92, 0, 163, 255, 95, 0, 134, 246, 191, 77, 0, 125, 255, 130, 0, 88,
  198, 226, 95, 0, 122, 255, 158, 43, 53, 213, 178, 21, 99, 214, 200, 49, 39, 198, 214, 75,
  0, 131, 255, 134, 6, 104, 219, 179, 18, 105, 251, 104, 4, 106, 228, 204, 45, 51, 185, 247,
```

```
76, 0, 178, 244, 39, 52, 199, 220, 85, 0, 105, 255, 182, 35, 99, 254, 96, 0, 118, 207,
243, 84, 0, 137, 255, 124, 0, 140, 253, 121, 43, 19, 162, 255, 147, 15, 60, 180, 255, 62,
26, 196, 222, 91, 0, 135, 255, 96, 0, 178, 255, 76, 6, 179, 255, 80, 0, 166, 255, 108,
0, 110, 255, 92, 0, 122, 228, 215, 57, 24, 181, 252, 75, 0, 118, 255, 134, 1, 133, 214,
188, 65, 0, 163, 255, 95, 0, 143, 255, 126, 34, 48, 193, 255, 65, 22, 208, 201, 50, 31,
209, 199, 42, 70, 190, 251, 57, 11, 190, 221, 87, 0, 140, 255, 129, 27, 88, 209, 202, 54,
28, 186, 251, 94, 0, 144, 255, 82, 4, 190, 255, 60, 19, 202, 220, 42, 55, 212, 186, 35,
78, 221, 169, 27, 95, 213, 189, 75, 0, 146, 255, 113, 0, 169, 255, 83, 0, 118, 241, 164,
25, 107, 202, 212, 79, 0, 133, 255, 153, 42, 62, 216, 164, 7, 148, 229, 154, 54, 0, 185,
255, 65, 9, 197, 232, 54, 34, 189, 244, 80, 0, 113, 238, 210, 40, 90, 214, 159, 51, 2,
197, 218, 59, 31, 192, 255, 70, 8, 168, 255, 122, 0, 117, 235, 92, 7, 130, 234, 188, 16,
124, 229, 121, 46, 15, 178, 255, 104, 0, 142, 255, 87, 0, 156, 254, 131, 25, 100, 212, 183,
64, 6, 164, 255, 107, 8, 97, 179, 255, 99, 0, 121, 229, 155, 44, 87, 214, 181, 60, 35,
175, 255, 119, 15, 50, 167, 255, 98, 0, 171, 241, 113, 14, 97, 238, 127, 19, 159, 252, 91,
0, 162, 251, 126, 21, 102, 212, 172, 62, 15, 174, 255, 90, 0, 163, 229, 169, 49, 14, 189,
231, 78, 0, 147, 255, 127, 31, 120, 221, 148, 36, 86, 227, 127, 20, 131, 215, 195, 38, 70,
205, 178, 66, 0, 167, 255, 103, 5, 139, 215, 175, 73, 0, 176, 255, 122, 15, 104, 213, 164,
68, 0, 179, 255, 73, 22, 197, 245, 65, 0, 175, 255, 88, 0, 167, 242, 116, 39, 41, 198,
242, 67, 12, 171, 255, 105, 0, 149, 224, 142, 29, 88, 195, 198, 63, 0, 192, 241, 66, 10,
180, 255, 90, 0, 177, 250, 63, 24, 191, 236, 85, 0, 135, 247, 98, 29, 134, 222, 197, 50,
52, 200, 209, 58, 19, 196, 230, 59, 20, 175, 255, 103, 0, 158, 239, 80, 0, 148, 209, 206,
71, 0, 192, 234, 56, 28, 189, 255, 85, 0, 149, 237, 135, 51, 38, 174, 255, 92, 0, 145,
211, 150, 63, 7, 189, 255, 61, 17, 184, 255, 119, 21, 52, 182, 240, 63, 0, 190, 245, 86,
4, 126, 229, 169, 62, 55, 195, 246, 55, 13, 192, 243, 65, 0, 181, 251, 118, 42, 38, 187,
255, 68, 0, 186, 255, 131, 45, 29, 182, 255, 83, 0, 148, 211, 187, 81, 0, 181, 255, 84,
9, 158, 240, 91, 11, 181, 249, 99, 19, 117, 229, 107, 33, 121, 215, 206, 38, 87, 213, 153,
47, 47, 211, 177, 32, 119, 225, 128, 24, 145, 236, 91, 31, 121, 208, 215, 61, 0, 181, 255,
85, 2, 165, 228, 99, 28, 92, 213, 194, 39, 108, 211, 164, 66, 0, 189, 255, 109, 42, 53,
202, 227, 62, 20, 179, 255, 99, 20, 97, 192, 219, 59, 1, 190, 255, 84, 23, 83, 186, 255,
62, 0, 191, 236, 66, 0, 193, 242, 59, 28, 208, 212, 56, 30, 181, 255, 65, 8, 198, 221,
46, 31, 204, 203, 42, 63, 213, 176, 53, 49, 194, 251, 60, 6, 167, 246, 92, 19, 133, 199,
211, 60, 0, 170, 252, 126, 32, 104, 209, 149, 41, 81, 204, 192, 42, 74, 218, 159, 49, 73,
193, 250, 78, 0, 170, 235, 67, 15, 177, 252, 87, 31, 126, 195, 243, 68, 0, 180, 230, 64,
8, 158, 242, 130, 40, 109, 191, 223, 64, 0, 170, 228, 151, 47, 38, 200, 213, 63, 0, 152,
243, 167, 39, 124, 222, 105, 54, 43, 190, 255, 57, 14, 175, 235, 142, 28, 91, 194, 171, 46,
23, 201, 240, 69, 7, 143, 239, 145, 34, 157, 238, 79, 41, 86, 190, 255, 59, 0, 191, 234,
112, 40, 43, 197, 248, 57, 3, 194, 255, 70, 17, 191, 232, 83, 13, 107, 209, 190, 54, 41,
209, 217, 52, 25, 205, 239, 54, 20, 192, 251, 113, 29, 43, 190, 253, 52, 0, 145, 235, 165,
34, 133, 224, 116, 33, 94, 212, 165, 33, 99, 230, 109, 23, 182, 233, 110, 32, 106, 218, 148,
30, 128, 232, 86, 15, 191, 240, 96, 42, 73, 201, 252, 45, 6, 208, 238, 53, 0, 157, 247,
130, 41, 104, 190, 243, 42, 0, 176, 222, 140, 44, 6, 201, 255, 57, 0, 169, 216, 175, 49,
16, 217, 203, 32, 72, 213, 185, 45, 51, 226, 176, 36, 89, 201, 250, 69, 0, 177, 227, 53,
21, 152, 245, 149, 39, 94, 200, 250, 59, 0, 134, 216, 146, 27, 144, 224, 121, 45, 46, 214,
240, 31, 64, 225, 157, 40, 91, 205, 206, 43, 16, 212, 254, 75, 0, 160, 236, 78, 37, 118,
215, 215, 35, 68, 209, 196, 46, 2, 209, 241, 60, 1, 174, 249, 80, 38, 139, 213, 221, 43,
0, 182, 242, 134, 20, 100, 205, 161, 49, 5, 209, 255, 44, 34, 223, 219, 44, 21, 187, 255,
64, 25, 136, 225, 168, 15, 134, 204, 139, 34, 36, 210, 219, 50, 6, 216, 237, 34, 58, 209,
222, 53, 0, 188, 230, 140, 36, 65, 227, 142, 41, 86, 218, 211, 29, 82, 218, 165, 37, 51,
206, 239, 34, 3, 220, 214, 41, 33, 203, 255, 54, 4, 172, 232, 98, 25, 145, 212, 137, 28,
93, 209, 173, 42, 23, 224, 207, 27, 94, 219, 179, 50, 6, 198, 255, 97, 26, 149, 222, 92,
34, 103, 212, 212, 39, 27, 211, 252, 69, 0, 160, 225, 89, 30, 132, 219, 163, 37, 75, 218,
195, 39, 36, 228, 196, 32, 84, 226, 152, 25, 130, 209, 151, 39, 48, 228, 167, 40, 69, 209,
251, 38, 22, 214, 224, 35, 2, 170, 241, 104, 26, 161, 204, 178, 41, 0, 207, 250, 90, 23,
118, 224, 112, 33, 144, 222, 137, 51, 53, 206, 254, 55, 4, 183, 214, 132, 46, 5, 213, 255,
42, 30, 172, 231, 167, 32, 45, 213, 202, 17, 86, 227, 113, 49, 85, 213, 252, 29, 32, 229,
189, 40, 32, 191, 254, 83, 13, 165, 220, 73, 35, 108, 222, 208, 34, 63, 219, 213, 35, 14,
195, 235, 137, 43, 0, 201, 255, 45, 34, 159, 229, 161, 30, 83, 220, 162, 41, 54, 205, 248,
38, 19, 177, 224, 170, 31, 25, 226, 198, 31, 71, 215, 236, 38, 0, 206, 237, 52, 31, 216,
```

```
294   217, 34, 22, 218, 249, 74, 16, 156, 222, 62, 28, 216, 241, 68, 17, 142, 220, 89, 19, 191,
295   227, 81, 42, 84, 207, 255, 35, 0, 180, 207, 157, 33, 51, 223, 180, 38, 42, 226, 230, 23,
296   63, 222, 188, 40, 9, 205, 255, 66, 45, 107, 209, 252, 26, 0, 213, 227, 84, 18, 140, 213,
297   132, 51, 24, 208, 255, 65, 39, 213, 222, 40, 26, 115, 215, 242, 27, 21, 224, 225, 48, 11,
298   155, 229, 123, 47, 119, 220, 168, 23, 121, 212, 129, 46, 25, 204, 255, 77, 29, 170, 206, 124,
299   46, 24, 202, 255, 63, 37, 208, 219, 46, 17, 166, 223, 91, 35, 179, 228, 70, 41, 135, 215,
300   214, 22, 20, 230, 192, 28, 83, 214, 191, 30, 32, 227, 207, 30, 71, 229, 169, 40, 61, 199,
301   248, 70, 19, 153, 205, 123, 42, 78, 221, 193, 22, 104, 209, 169, 42, 6, 215, 246, 58, 41,
302   203, 218, 46, 17, 192, 221, 73, 41, 135, 227, 154, 45, 75, 202, 242, 59, 12, 173, 201, 45,
303   44, 149, 232, 171, 40, 72, 208, 230, 27, 10, 215, 203, 38, 45, 217, 226, 30, 31, 217, 210,
304   36, 26, 202, 223, 134, 45, 34, 227, 213, 32, 65, 221, 194, 36, 47, 198, 231, 115, 24, 93,
305   197, 170, 32, 54, 228, 179, 44, 56, 198, 238, 128, 44, 39, 213, 227, 23, 44, 218, 212, 46,
306   20, 163, 220, 104, 61, 101, 212, 245, 43, 26, 106, 201, 236, 30, 15, 213, 218, 55, 46, 122,
307   214, 203, 24, 73, 215, 163, 37, 66, 224, 174, 42, 69, 198, 235, 92, 42, 84, 200, 226, 27,
308   23, 212, 209, 65, 38, 151, 217, 98, 45, 188, 210, 94, 61, 70, 210, 242, 65, 47, 73, 203,
309   247, 27, 38, 145, 206, 219, 30, 16, 155, 224, 152, 44, 116, 203, 177, 37, 28, 198, 227, 82,
310   47, 153, 199, 123, 44, 103, 219, 130, 42, 169, 197, 120, 52, 41, 188, 239, 80, 49, 181, 206,
311   68, 45, 163, 196, 166, 35, 37, 227, 180, 41, 92, 221, 170, 41, 101, 211, 172, 46, 40, 199,
312   223, 53, 60, 138, 218, 182, 27, 106, 201, 130, 49, 49, 188, 234, 115, 46, 137, 203, 88, 49,
313   174, 196, 119, 53, 65, 210, 210, 31, 32, 194, 214, 72, 51, 198, 199, 50, 43, 167, 208, 79,
314   50, 188, 205, 85, 49, 136, 193, 160, 43, 33, 217, 220, 79, 55, 103, 206, 199, 36, 49, 219,
315   189, 41, 76, 213, 198, 36, 38, 183, 216, 99, 62, 89, 202, 234, 47, 40, 167, 198, 94, 45,
316   182, 194, 90, 60, 69, 214, 218, 44, 77, 210, 188, 33, 45, 139, 213, 205, 36, 50, 217, 191,
317   42, 43, 185, 215, 90, 69, 100, 204, 220, 34, 61, 208, 186, 41, 37, 168, 206, 128, 57, 113,
318   205, 172, 41, 68, 220, 163, 50, 83, 206, 215, 48, 41, 162, 194, 124, 62, 70, 190, 224, 65,
319   59, 205, 184, 41, 52, 208, 199, 61, 49, 145, 202, 112, 61, 147, 202, 142, 47, 105, 204, 147,
320   45, 100, 207, 156, 47, 89, 206, 176, 33, 106, 205, 116, 64, 85, 197, 218, 62, 59, 124, 200,
321   179, 33, 92, 207, 137, 49, 110, 194, 170, 40, 59, 219, 168, 47, 93, 206, 186, 39, 52, 149,
322   213, 184, 37, 66, 209, 191, 45, 42, 202, 184, 50, 73, 133, 224, 186, 35, 112, 191, 150, 45,
323   42, 201, 206, 70, 62, 160, 194, 118, 60, 105, 207, 172, 46, 67, 205, 195, 44, 72, 206, 181,
```

```
324   44, 49, 181, 200, 107, 66, 116, 198, 174, 43, 51, 164, 216, 121, 51, 175, 179, 89, 61, 95,
325   210, 158, 53, 100, 205, 186, 39, 51, 169, 206, 119, 65, 88, 183, 211, 59, 60, 200, 176, 59,
326   61, 116, 203, 190, 41, 75, 212, 177, 46, 59, 197, 196, 65, 68, 164, 195, 105, 57, 168, 181,
327   87, 66, 109, 201, 191, 40, 51, 192, 194, 120, 64, 80, 205, 203, 50, 54, 195, 182, 79, 73,
328   99, 210, 190, 50, 83, 203, 192, 45, 56, 174, 188, 86, 65, 196, 176, 46, 79, 208, 169, 49,
329   68, 193, 198, 100, 70, 84, 202, 199, 44, 65, 125, 207, 181, 51, 81, 200, 190, 47, 60, 172,
330   184, 84, 67, 188, 185, 84, 65, 144, 195, 107, 73, 111, 194, 206, 60, 61, 149, 184, 111, 60,
331   167, 177, 104, 69, 80, 206, 192, 58, 79, 199, 173, 55, 83, 140, 205, 175, 29, 47, 181, 184,
332   78, 83, 102, 195, 204, 54, 73, 107, 198, 187, 47, 78, 144, 205, 130, 60, 94, 189, 193, 66,
333   65, 114, 196, 152, 52, 135, 182, 137, 58, 68, 207, 196, 78, 69, 154, 182, 93, 73, 157, 187,
334   104, 77, 103, 182, 207, 69, 72, 192, 170, 64, 63, 166, 177, 90, 85, 105, 188, 204, 67, 75,
335   198, 171, 64, 76, 104, 198, 203, 65, 73, 121, 190, 177, 43, 72, 157, 197, 167, 50, 65, 202,
336   164, 66, 88, 145, 210, 138, 63, 95, 184, 190, 74, 68, 111, 191, 184, 42, 71, 201, 172, 57,
337   82, 149, 198, 174, 47, 64, 187, 180, 71, 87, 193, 159, 56, 82, 195, 173, 62, 82, 184, 179,
338   75, 75, 156, 184, 104, 69, 148, 181, 105, 74, 116, 192, 178, 50, 72, 168, 183, 97, 76, 139,
339   178, 150, 51, 92, 196, 153, 55, 80, 195, 183, 65, 87, 195, 147, 66, 89, 167, 191, 104, 76,
340   108, 191, 162, 53, 82, 189, 178, 67, 79, 117, 194, 189, 48, 70, 150, 185, 116, 69, 161, 166,
341   107, 78, 89, 196, 185, 57, 86, 192, 164, 62, 82, 169, 182, 96, 76, 157, 170, 121, 65, 114,
342   189, 117, 78, 108, 171, 197, 116, 60, 115, 180, 125, 71, 97, 189, 181, 60, 89, 195, 162, 63,
343   76, 149, 184, 120, 70, 162, 168, 80, 84, 155, 184, 105, 80, 110, 175, 184, 68, 83, 150, 177,
344   152, 49, 71, 187, 175, 96, 79, 97, 184, 181, 71, 84, 170, 171, 86, 84, 131, 192, 167, 52,
345   77, 167, 178, 111, 75, 91, 179, 185, 87, 81, 141, 179, 114, 83, 109, 173, 188, 85, 82, 112,
346   186, 170, 55, 91, 192, 163, 60, 78, 157, 181, 109, 84, 138, 185, 149, 61, 83, 171, 182, 117,
347   74, 88, 193, 174, 67, 86, 120, 191, 180, 52, 85, 168, 174, 90, 82, 180, 156, 85, 85, 111,
348   193, 164, 65, 99, 173, 176, 107, 71, 96, 187, 162, 65, 98, 184, 163, 72, 90, 185, 165, 72,
349   83, 156, 175, 123, 73, 116, 189, 133, 80, 115, 188, 159, 59, 86, 133, 197, 157, 63, 109, 182,
350   157, 58, 82, 157, 178, 121, 75, 92, 149, 194, 131, 68, 87, 159, 182, 106, 77, 121, 179, 135,
351   74, 107, 180, 160, 65, 91, 169, 169, 104, 78, 99, 179, 173, 73, 95, 185, 160, 74, 89, 128,
352   182, 163, 57, 93, 190, 159, 70, 86, 175, 159, 82, 98, 146, 186, 153, 63, 83, 180, 166, 86,
```

SysTick.c: Onboard timer and interrupt handler to make certain actions happen

```c
// SysTick.c
// Runs on TM4C123
// By Dr. Min He
// December 10th, 2018

#include <stdint.h>
#include "SysTick.h"
#include "tm4c123gh6pm.h"

#define ONE_MICRO_SECOND          80     // number of machine cycles to generate 1us delay for 80MHz system clock

void SysTick_Start(void){
  NVIC_ST_CTRL_R = 0;
  NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M; // number of counts to wait
  NVIC_ST_CURRENT_R = 0; // any value written to CURRENT clears
  NVIC_ST_CTRL_R |= NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC; // enable SysTick with core clock
}

// Disable Systick timer
void SysTick_Stop(void){
  NVIC_ST_CTRL_R = 0;
}

// Calculate number of machine cycles elapsed
uint32_t SysTick_Get_MC_Elapsed(void){
  return NVIC_ST_RELOAD_R-NVIC_ST_CURRENT_R;
}

// Time delay using busy wait.
// This function assumes 16 MHz system clock.
void SysTick_Waitlus(uint32_t delay){
  NVIC_ST_CTRL_R = 0;
  NVIC_ST_RELOAD_R = delay*ONE_MICRO_SECOND-1; // number of counts to wait
  NVIC_ST_CURRENT_R = 0; // any value written to CURRENT clears
  NVIC_ST_CTRL_R |= NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC;
  while((NVIC_ST_CTRL_R&NVIC_ST_CTRL_COUNT)==0); // wait for count flag
  NVIC_ST_CTRL_R = 0;
}

// Time delay using busy wait.
// This assumes 50 MHz system clock.
void SysTick_Waitlms(uint32_t delay){
  unsigned long i;
  for(i=0; i<delay; i++){
    SysTick_Waitlus(1000);  // wait 1ms (assumes 50 MHz clock)
  }
}

void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                                   // disable SysTick during setup
  NVIC_ST_RELOAD_R = (2000000*4) - 1;            // number of counts to wait 10hz (assuming 80MHz clock)
  NVIC_ST_CURRENT_R = 0;                                // any write to CURRENT clears
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x1FFFFFFF)|0x20000000;    // priority 1
  NVIC_ST_CTRL_R = 0x07;
}
```

Conclusion

We struggled a lot with this project, but we were able to complete it on time and achieve all that was required of the project. Most of the issues came from implementing the enemies and how to make the explosion animation happen when laser hits the enemy. After resolving the enemy issue in game, the rest was straight forward and manageable to do. We were able to complete the project in time and able to do some of the extra credit.