LEGEND

- **Intellectual Property of Gerry O'Brien is in GREEN**
- **Intellectual Property of Tyler Upchurch and Evan Strack is in YELLOW**

```
-- Original Creation of LCD workings by - Gerry O'Brien
-- Work taken from "http://www.digital-circuitry.com/Projects_LCD_DISPLAYS.htm"

-- Binverter Game
-- Tyler Upchurch and Evan Strack
-- Miami University (C) 2017
-- ECE 287 Final Project


LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.all;
USE  IEEE.STD_LOGIC_ARITH.all;
USE  IEEE.STD_LOGIC_UNSIGNED.all;


ENTITY Binverter IS
  GENERIC(
    num_hex_digits : integer := 2
  );

  PORT(
    reset             : IN     std_logic;  -- Map this Port to a Switch within your [Port Declarations / Pin Planer]
    clock_50          : IN     std_logic;  -- Using the DE2 50Mhz Clk, in order to Genreate the 400Hz signal... clk_count_400hz reset count value must be set to:  <= x"0F424"

    lcd_rs            : OUT    std_logic;
    lcd_e             : OUT    std_logic;
    lcd_rw            : OUT    std_logic;
    lcd_on            : OUT    std_logic;
    lcd_blon          : OUT    std_logic;


    data_bus_0        : INOUT  STD_LOGIC;
    data_bus_1        : INOUT  STD_LOGIC;
    data_bus_2        : INOUT  STD_LOGIC;
    data_bus_3        : INOUT  STD_LOGIC;
    data_bus_4        : INOUT  STD_LOGIC;
```

```
    data_bus_5        : INOUT  STD_LOGIC;
    data_bus_6        : INOUT  STD_LOGIC;
    data_bus_7        : INOUT  STD_LOGIC;

    Hex_Display_Data_0 : IN    STD_LOGIC;
    Hex_Display_Data_1 : IN    STD_LOGIC;
    Hex_Display_Data_2 : IN    STD_LOGIC;
    Hex_Display_Data_3 : IN    STD_LOGIC;
    Hex_Display_Data_4 : IN    STD_LOGIC;
    Hex_Display_Data_5 : IN    STD_LOGIC;
    Hex_Display_Data_6 : IN    STD_LOGIC;
    Hex_Display_Data_7 : IN    STD_LOGIC;

                resetSW16 : in  std_logic;  -- control the reset of the game to reset level
                skipToLevel11, skipToLevel21, skipToLevel30, skipToFinalLoss : in std_logic;  -
- buttons to act as asynchronous resets to test different features in the FSM
                SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9, SW10, SW11,
SW12, SW13, enterGuess, startGame  : in  std_logic;
                LEDG0, LEDG1, LEDG2, LEDG3, LEDG4, LEDG5, LEDG6, LEDG7  : out
std_logic; -- output green lights
                LEDR0, LEDR1, LEDR2, LEDR3, LEDR4, LEDR5, LEDR6, LEDR7, LEDR8,
LEDR9, LEDR10  : out std_logic; -- output red lights
                LEDR11, LEDR12, LEDR13, LEDR14, LEDR15, LEDR16, LEDR17  : out
std_logic -- output red lights

   );

END Binverter;


ARCHITECTURE Binverter_arch OF Binverter IS

        type character_string is array ( 0 to 31 ) of STD_LOGIC_VECTOR( 7 downto 0 );

        type game_type is (BuggedState, ResetState, L1, L2, L3, L4, L5, L6, L7, L8, L9, L10,
L11, L12, L13, L14, L15, L16, L17, L18, L19, L20, L21, L22, L23, L24, L25, L26, L27, L28,
L29, L30, FailIntermediate, FailState, CorrectState, FinalWin, FinalLoss);  -- enumeration to
hold our states

        type level_type is (Level_1, Level_2, Level_3, Level_4, Level_5, Level_6, Level_7,
Level_8, Level_9, Level_10, Level_11, Level_12, Level_13, Level_14, Level_15, Level_16,
Level_17, Level_18, Level_19, Level_20, Level_21, Level_22, Level_23, Level_24, Level_25,
Level_26, Level_27, Level_28, Level_29, Level_30);

        signal gameState : game_type := ResetState;
        signal levelState : level_type := Level_1;
```

```
        shared variable lifeCounter : natural range 0 to 255;

        signal counter : std_logic_vector(24 downto 0);                    -- signal that does the
counting for 1 second
        signal redLightCounter : std_logic_vector(24 downto 0);      -- signal that does the
counting for 1 second
        signal greenLightCounter : std_logic_vector(24 downto 0);  -- signal that does the
counting for 1 second
        shared variable delay3sIsOver : boolean := false;                  -- delay program for
5s on Success State
        shared variable delay10sIsOver : boolean := false;                 -- delay program for
10s on Fail State
        signal delay_3s : std_logic_vector(5 downto 0);                    -- signal to control
delay for Correct State
        signal delay_10s : std_logic_vector(10 downto 0);                  -- signal to control
delays for Fail State
        signal REDLIGHT_CONTROLLER : std_logic;                                    -- to
drive the LED for red light blinking
        signal GREENLIGHT_CONTROLLER : std_logic;                                  -- to
drive the LED for green light blinking
        shared variable guessEntered : boolean := false;

        signal lcd_display_level1life3          : character_string;
        signal lcd_display_level1life2          : character_string;
        signal lcd_display_level1life1          : character_string;
        signal lcd_display_level2life3          : character_string;
        signal lcd_display_level2life2          : character_string;
        signal lcd_display_level2life1          : character_string;
        signal lcd_display_level3life3          : character_string;
        signal lcd_display_level3life2          : character_string;
        signal lcd_display_level3life1          : character_string;
        signal lcd_display_level4life3          : character_string;
        signal lcd_display_level4life2          : character_string;
        signal lcd_display_level4life1          : character_string;
        signal lcd_display_level5life3          : character_string;
        signal lcd_display_level5life2          : character_string;
        signal lcd_display_level5life1          : character_string;
        signal lcd_display_level6life3          : character_string;
        signal lcd_display_level6life2          : character_string;
        signal lcd_display_level6life1          : character_string;
        signal lcd_display_level7life3          : character_string;
        signal lcd_display_level7life2          : character_string;
        signal lcd_display_level7life1          : character_string;
        signal lcd_display_level8life3          : character_string;
        signal lcd_display_level8life2          : character_string;
```

```
signal lcd_display_level8life1          : character_string;
signal lcd_display_level9life3          : character_string;
signal lcd_display_level9life2          : character_string;
signal lcd_display_level9life1          : character_string;
signal lcd_display_level10life3         : character_string;
signal lcd_display_level10life2         : character_string;
signal lcd_display_level10life1         : character_string;
signal lcd_display_level11life3         : character_string;
signal lcd_display_level11life2         : character_string;
signal lcd_display_level11life1         : character_string;
signal lcd_display_level12life3         : character_string;
signal lcd_display_level12life2         : character_string;
signal lcd_display_level12life1         : character_string;
signal lcd_display_level13life3         : character_string;
signal lcd_display_level13life2         : character_string;
signal lcd_display_level13life1         : character_string;
signal lcd_display_level14life3         : character_string;
signal lcd_display_level14life2         : character_string;
signal lcd_display_level14life1         : character_string;
signal lcd_display_level15life3         : character_string;
signal lcd_display_level15life2         : character_string;
signal lcd_display_level15life1         : character_string;
signal lcd_display_level16life3         : character_string;
signal lcd_display_level16life2         : character_string;
signal lcd_display_level16life1         : character_string;
signal lcd_display_level17life3         : character_string;
signal lcd_display_level17life2         : character_string;
signal lcd_display_level17life1         : character_string;
signal lcd_display_level18life3         : character_string;
signal lcd_display_level18life2         : character_string;
signal lcd_display_level18life1         : character_string;
signal lcd_display_level19life3         : character_string;
signal lcd_display_level19life2         : character_string;
signal lcd_display_level19life1         : character_string;
signal lcd_display_level20life3         : character_string;
signal lcd_display_level20life2         : character_string;
signal lcd_display_level20life1         : character_string;
signal lcd_display_level21life3         : character_string;
signal lcd_display_level21life2         : character_string;
signal lcd_display_level21life1         : character_string;
signal lcd_display_level22life3         : character_string;
signal lcd_display_level22life2         : character_string;
signal lcd_display_level22life1         : character_string;
signal lcd_display_level23life3         : character_string;
signal lcd_display_level23life2         : character_string;
signal lcd_display_level23life1         : character_string;
```

```
        signal lcd_display_level24life3                 : character_string;
        signal lcd_display_level24life2                 : character_string;
        signal lcd_display_level24life1                 : character_string;
        signal lcd_display_level25life3                 : character_string;
        signal lcd_display_level25life2                 : character_string;
        signal lcd_display_level25life1                 : character_string;
        signal lcd_display_level26life3                 : character_string;
        signal lcd_display_level26life2                 : character_string;
        signal lcd_display_level26life1                 : character_string;
        signal lcd_display_level27life3                 : character_string;
        signal lcd_display_level27life2                 : character_string;
        signal lcd_display_level27life1                 : character_string;
        signal lcd_display_level28life3                 : character_string;
        signal lcd_display_level28life2                 : character_string;
        signal lcd_display_level28life1                 : character_string;
        signal lcd_display_level29life3                 : character_string;
        signal lcd_display_level29life2                 : character_string;
        signal lcd_display_level29life1                 : character_string;
        signal lcd_display_level30life3                 : character_string;
        signal lcd_display_level30life2                 : character_string;
        signal lcd_display_level30life1                 : character_string;
        signal lcd_display_convertdectobin              : character_string;
        signal lcd_display_converthextobin              : character_string;
        signal lcd_display_convertocttobin              : character_string;
        signal lcd_display_levelFail            : character_string;
        signal lcd_display_levelPass            : character_string;
        signal lcd_display_PERFECTFinalWin      : character_string;
        signal lcd_display_LOSTONFIRSTROUND : character_string;
        signal lcd_display_finalWin             : character_string;
        signal lcd_display_finalLoss            : character_string;
        signal lcd_display_bugMessage                   : character_string;
        signal lcd_display_resetMessage     : character_string;

        type state_type is (hold, func_set, display_on, mode_set, print_string,
                line2, return_home, drop_lcd_e, reset1, reset2,
                 reset3, display_off, display_clear);

        signal state, next_command      : state_type;

        signal lcd_display_string       : character_string;

        signal data_bus_value, next_char   : STD_LOGIC_VECTOR(7 downto 0);
        signal clk_count_400hz          : STD_LOGIC_VECTOR(19 downto 0);

        signal char_count               : STD_LOGIC_VECTOR(4 downto 0);
        signal clk_400hz_enable,lcd_rw_int : std_logic;
```

```
        signal Hex_Display_Data         : STD_LOGIC_VECTOR(7 DOWNTO 0);
        signal data_bus                 : STD_LOGIC_VECTOR(7 downto 0);

BEGIN



--=================================================--
-- SIGNAL STD_LOGIC_VECTORS assigned to OUTPUT PORTS
--=================================================--
Hex_Display_Data(0) <= Hex_Display_Data_0;
Hex_Display_Data(1) <= Hex_Display_Data_1;
Hex_Display_Data(2) <= Hex_Display_Data_2;
Hex_Display_Data(3) <= Hex_Display_Data_3;
Hex_Display_Data(4) <= Hex_Display_Data_4;
Hex_Display_Data(5) <= Hex_Display_Data_5;
Hex_Display_Data(6) <= Hex_Display_Data_6;
Hex_Display_Data(7) <= Hex_Display_Data_7;

data_bus_0 <= data_bus(0);
data_bus_1 <= data_bus(1);
data_bus_2 <= data_bus(2);
data_bus_3 <= data_bus(3);
data_bus_4 <= data_bus(4);
data_bus_5 <= data_bus(5);
data_bus_6 <= data_bus(6);
data_bus_7 <= data_bus(7);



-- ASCII hex values for LCD Display
-- Enter Live Hex Data Values from hardware here

-- LCD DISPLAYS THE FOLLOWING:

----------------------------
--| Count=XX |
--| DE2 |
----------------------------

-- Lives: 3
--x"4C",x"49",x"56",x"45",x"53",x"3A",x"33"
-- Lives: 2
```

```
--x"4C",x"49",x"56",x"45",x"53",x"3A",x"32"
-- Lives: 1
--x"4C",x"49",x"56",x"45",x"53",x"3A",x"31"

-- CONVERT: dec4321
--
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"34",x"33",x"32",
x"31"

-- General structure of the level strings
-- Line 1  LEVEL:30 LIVES:3
-- Line 2  CONVERT: dec4321


-- Level 1
lcd_display_level1life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"31");
lcd_display_level1life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"31");
lcd_display_level1life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"31");

-- Level 2
lcd_display_level2life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"36");
lcd_display_level2life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",
```

```
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"36");
lcd_display_level2life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"20",
x"36");

-- Level 3
lcd_display_level3life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"31",
x"37");
lcd_display_level3life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"31",
x"37");
lcd_display_level3life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"31",
x"37");

-- Level 4
lcd_display_level4life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"38",
x"36");
lcd_display_level4life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"38",
x"36");
lcd_display_level4life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",
```

```
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"20",x"38",
x"36");

-- Level 5
lcd_display_level5life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"31",x"31",
x"32");
lcd_display_level5life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"31",x"31",
x"32");
lcd_display_level5life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"31",x"31",
x"32");

-- Level 6
lcd_display_level6life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"33",x"34",
x"31");
lcd_display_level6life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"33",x"34",
x"31");
lcd_display_level6life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"33",x"34",
x"31");

-- Level 7
```

```
lcd_display_level7life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"38",x"39",
x"31");
lcd_display_level7life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"38",x"39",
x"31");
lcd_display_level7life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"20",x"38",x"39",
x"31");

-- Level 8
lcd_display_level8life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"32",x"33",x"36",
x"38");
lcd_display_level8life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"32",x"33",x"36",
x"38");
lcd_display_level8life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"32",x"33",x"36",
x"38");

-- Level 9
lcd_display_level9life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"35",x"34",x"35",
x"30");
```

```
lcd_display_level9life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"35",x"34",x"35",
x"30");
lcd_display_level9life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"20",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"35",x"34",x"35",
x"30");

-- Level 10
lcd_display_level10life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"38",x"37",x"36",
x"31");
lcd_display_level10life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"38",x"37",x"36",
x"31");
lcd_display_level10life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"64",x"65",x"63",x"38",x"37",x"36",
x"31");

-- Level 11
lcd_display_level11life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"39");
lcd_display_level11life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"39");
```

```
lcd_display_level11life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"39");

-- Level 12
lcd_display_level12life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"43");
lcd_display_level12life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"43");
lcd_display_level12life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"20",
x"43");

-- Level 13
lcd_display_level13life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"33",
x"32");
lcd_display_level13life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"33",
x"32");
lcd_display_level13life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"20",x"33",
x"32");
```

```
-- Level 14
lcd_display_level14life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"41",
x"35");
lcd_display_level14life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"41",
x"35");
lcd_display_level14life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"41",
x"35");

-- Level 15
lcd_display_level15life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"46",
x"34");
lcd_display_level15life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"46",
x"34");
lcd_display_level15life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"31",x"46",
x"34");

-- Level 16
lcd_display_level16life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",
```

```
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"46",x"41",
x"32");
lcd_display_level16life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"46",x"41",
x"32");
lcd_display_level16life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"20",x"46",x"41",
x"32");

-- Level 17
lcd_display_level17life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"38",
x"41");
lcd_display_level17life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"38",
x"41");
lcd_display_level17life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"38",
x"41");

-- Level 18
lcd_display_level18life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"45",x"41",
x"41");
lcd_display_level18life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",
```

```
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"45",x"41",
x"41");
lcd_display_level18life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"45",x"41",
x"41");

-- Level 19
lcd_display_level19life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"46",x"39",
x"41");
lcd_display_level19life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"46",x"39",
x"41");
lcd_display_level19life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"32",x"46",x"39",
x"41");

-- Level 20
lcd_display_level20life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"46",
x"46");
lcd_display_level20life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"46",
x"46");
lcd_display_level20life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",
```

```
x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"68",x"65",x"78",x"33",x"46",x"46",
x"46");

-- Level 21
lcd_display_level21life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"20",
x"35");
lcd_display_level21life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"20",
x"35");
lcd_display_level21life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"31",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"20",
x"35");

-- Level 22
lcd_display_level22life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"31",
x"31");
lcd_display_level22life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"31",
x"31");
lcd_display_level22life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"32",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"31",
x"31");

-- Level 23
```

```
lcd_display_level23life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"35",
x"33");
lcd_display_level23life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"35",
x"33");
lcd_display_level23life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"33",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"20",x"35",
x"33");

-- Level 24
lcd_display_level24life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"31",x"31",
x"33");
lcd_display_level24life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"31",x"31",
x"33");
lcd_display_level24life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"34",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"31",x"31",
x"33");

-- Level 25
lcd_display_level25life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"36",x"37",
x"32");
```

```
lcd_display_level25life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"36",x"37",
x"32");
lcd_display_level25life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"35",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"36",x"37",
x"32");

-- Level 26
lcd_display_level26life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"37",x"34",
x"31");
lcd_display_level26life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"37",x"34",
x"31");
lcd_display_level26life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"31",x"36",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"20",x"37",x"34",
x"31");

-- Level 27
lcd_display_level27life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"31",x"30",x"37",
x"36");
lcd_display_level27life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"31",x"30",x"37",
x"36");
```

```
lcd_display_level27life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"37",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"31",x"30",x"37",
x"36");

-- Level 28
lcd_display_level28life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"32",x"37",x"34",
x"35");
lcd_display_level28life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"32",x"37",x"34",
x"35");
lcd_display_level28life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"38",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"32",x"37",x"34",
x"35");

-- Level 29
lcd_display_level29life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"36",x"37",x"32",
x"34");
lcd_display_level29life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"36",x"37",x"32",
x"34");
lcd_display_level29life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"32",x"39",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"36",x"37",x"32",
x"34");
```

```
-- Level 30
lcd_display_level30life3 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"33",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"33",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"37",x"37",x"31",
x"32");
lcd_display_level30life2 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"33",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"32",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"37",x"37",x"31",
x"32");
lcd_display_level30life1 <=
(x"4C",x"45",x"56",x"45",x"4C",x"3A",x"33",x"30",x"20",x"4C",x"49",x"56",x"45",x"53",x"3
A",x"31",

x"43",x"4F",x"4E",x"56",x"45",x"52",x"54",x"3A",x"20",x"6F",x"63",x"74",x"37",x"37",x"31",
x"32");

-- Convert ___ to ___ strings
lcd_display_convertdectobin <=
(x"20",x"20",x"20",x"47",x"65",x"74",x"20",x"52",x"65",x"61",x"64",x"79",x"21",x"20",x"20",
x"20",

x"43",x"6F",x"6E",x"76",x"65",x"72",x"74",x"20",x"64",x"65",x"63",x"54",x"4F",x"62",x"69",
x"6E");
lcd_display_converthextobin <=
(x"20",x"20",x"20",x"47",x"65",x"74",x"20",x"52",x"65",x"61",x"64",x"79",x"21",x"20",x"20",
x"20",

x"43",x"6F",x"6E",x"76",x"65",x"72",x"74",x"20",x"68",x"65",x"78",x"54",x"4F",x"62",x"69",
x"6E");
lcd_display_convertocttobin <=
(x"20",x"20",x"20",x"47",x"65",x"74",x"20",x"52",x"65",x"61",x"64",x"79",x"21",x"20",x"20",
x"20",

x"43",x"6F",x"6E",x"76",x"65",x"72",x"74",x"20",x"6F",x"63",x"74",x"54",x"4F",x"62",x"69",
x"6E");

-- Intermediate Fail and Correct strings
lcd_display_levelFail <=
(x"20",x"20",x"20",x"49",x"4E",x"43",x"4f",x"52",x"52",x"45",x"43",x"54",x"21",x"20",x"20",
x"20",
```

```
x"20",x"20",x"20",x"54",x"52",x"59",x"20",x"41",x"47",x"41",x"49",x"4E",x"21",x"20",x"20",
x"20");
lcd_display_levelPass <=
(x"43",x"4F",x"4E",x"47",x"52",x"41",x"54",x"55",x"4C",x"41",x"54",x"49",x"4F",x"4E",x"53
",x"21",

x"20",x"20",x"20",x"4E",x"45",x"58",x"54",x"20",x"4C",x"45",x"56",x"45",x"4C",x"20",x"20",
x"20");

-- Final Win and Final Loss strings
lcd_display_PERFECTFinalWin <=
(x"43",x"4F",x"4E",x"47",x"52",x"41",x"54",x"55",x"4C",x"41",x"54",x"49",x"4F",x"4E",x"53
",x"21",

x"20",x"50",x"45",x"52",x"46",x"45",x"43",x"54",x"20",x"47",x"41",x"4D",x"45",x"21",x"21",
x"20");
lcd_display_LOSTONFIRSTROUND <=
(x"55",x"68",x"2E",x"2E",x"20",x"59",x"6F",x"75",x"20",x"6C",x"6F",x"73",x"74",x"20",x"6F
",x"6E",

x"74",x"68",x"65",x"20",x"66",x"69",x"72",x"73",x"74",x"20",x"6C",x"65",x"76",x"65",x"6C",
x"21");
lcd_display_finalWin <=
(x"43",x"4F",x"4E",x"47",x"52",x"41",x"54",x"55",x"4C",x"41",x"54",x"49",x"4F",x"4E",x"53
",x"21",

x"59",x"6F",x"75",x"20",x"77",x"6F",x"6E",x"20",x"74",x"68",x"65",x"20",x"67",x"61",x"6D",
x"65");
lcd_display_finalLoss <=
(x"20",x"20",x"20",x"42",x"65",x"74",x"74",x"65",x"72",x"20",x"6C",x"75",x"63",x"6B",x"20"
,x"20",

x"20",x"20",x"20",x"20",x"6E",x"65",x"78",x"74",x"20",x"74",x"69",x"6D",x"65",x"21",x"20",
x"20");

-- Bugged state message
lcd_display_bugMessage <=
(x"54",x"48",x"45",x"52",x"45",x"20",x"57",x"41",x"53",x"20",x"41",x"20",x"42",x"55",x"47",
x"21",

        x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",
x"20",x"20");

-- Reset state message
```

```
lcd_display_resetMessage <=
(x"52",x"65",x"73",x"65",x"74",x"74",x"69",x"6E",x"67",x"2E",x"2E",x"2E",x"20",x"20",x"20
",x"20",

x"53",x"57",x"31",x"35",x"20",x"74",x"6F",x"20",x"62",x"65",x"67",x"69",x"6E",x"20",x"20",
x"20");

-------------------------------------------------------------------------------------------
-- BIDIRECTIONAL TRI STATE LCD DATA BUS
  data_bus <= data_bus_value when lcd_rw_int = '0' else "ZZZZZZZZ";

-- LCD_RW PORT is assigned to it matching SIGNAL
 lcd_rw <= lcd_rw_int;


--------------------------- STATE MACHINE FOR Game playing and LCD message select ---------
--------------------
-------------------------------------------------------------------------------------------------------
PROCESS (gameState, clock_50, resetSW16, SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7,
SW8, SW9, SW10, SW11, SW12, SW13, enterGuess, startGame)
BEGIN

                    if resetSW16 = '1' then   -- asynchronous reset
                              gameState <= ResetState;
                              levelState <= Level_1;
                    elsif skipToLevel11 = '0' then  -- active low - button pressed to go straight to level
11
                              -- Turn off all the lights before entering next state
                              LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <= '0'; LEDR4 <=
'0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                              LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13 <= '0';
LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                              LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <= '0'; LEDG4 <=
'0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                              gameState <= L11;
                              levelState <= Level_11;
                    elsif skipToLevel21 = '0' then  -- active low - button pressed to go straight to level
21
                              -- Turn off all the lights
                              LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <= '0'; LEDR4 <=
'0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                              LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13 <= '0';
LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                              LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <= '0'; LEDG4 <=
'0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';
```

```
                gameState <= L21;
                levelState <= Level_21;
            elsif skipToLevel30 = '0' then  -- active low - button pressed to go straight to level
30
                -- Turn off all the lights
                LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <= '0'; LEDR4 <=
'0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13 <= '0';
LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <= '0'; LEDG4 <=
'0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                gameState <= L30;
                levelState <= Level_30;
            elsif skipToFinalLoss = '0' then  -- active low - button pressed to go straight to
Final Loss
                -- Turn off all the lights
                LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <= '0'; LEDR4 <=
'0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13 <= '0';
LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <= '0'; LEDG4 <=
'0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                gameState <= FinalLoss;
                levelState <= Level_30;
            elsif clock_50'event and clock_50 = '1' then  -- rising clock edge

                if redLightCounter < "10111110101111000001000000" then
                        redLightCounter <= redLightCounter + 8;
                        greenLightCounter <= greenLightCounter + 8;
                else
                        REDLIGHT_CONTROLLER <= not
REDLIGHT_CONTROLLER;
                        GREENLIGHT_CONTROLLER <= not
GREENLIGHT_CONTROLLER;
                        redLightCounter <= (others => '0');
                        greenLightCounter <= (others => '0');
                end if;

                if counter < "10111110101111000001000000" then
                        counter <= counter + 1;
                else
                        REDLIGHT_CONTROLLER <= not
REDLIGHT_CONTROLLER;
```

```
                                GREENLIGHT_CONTROLLER <= not
GREENLIGHT_CONTROLLER;
                                delay_3s <= delay_3s + 1;
                                delay_10s <= delay_10s + 1;
                                counter <= (others => '0');

                                -- 3s
                                if (delay_3s = 3) then
                                        delay3sIsOver := true;
                                        delay_3s <= (others => '0');
                                end if;

                                -- 10s
                                if (delay_10s = 10) then
                                        delay10sIsOver := true;
                                        delay_10s <= (others => '0');
                                end if;

                                -- Reset these when they are not in use
                                if (guessEntered = false) then
                                        delay3sIsOver := false;
                                        delay10sIsOver := false;
                                        delay_3s <= (others => '0');
                                        delay_10s <= (others => '0');
                                end if;
                        end if;

                CASE (gameState) IS

                        when BuggedState =>
                                next_char <=
lcd_display_bugMessage(CONV_INTEGER(char_count));

                        when ResetState =>
                                if (startGame = '1') then
                                        gameState <= L1;
                                        levelState <= Level_1;
                                else
                                        next_char <=
lcd_display_resetMessage(CONV_INTEGER(char_count));

                                        -- Turn off all the lights
                                        LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <=
'0'; LEDR4 <= '0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                                        LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13
<= '0'; LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
```

```
                                        LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <=
'0'; LEDG4 <= '0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                                        guessEntered := false;

                                        lifeCounter := 3;
                                        levelState <= Level_1;
                                end if;

                        when L1 =>
                                levelState <= Level_1;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level1life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level1life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level1life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base10 1
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L2 =>
                                levelState <= Level_2;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level2life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
```

```
                                        next_char <=
lcd_display_level2life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level2life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base10 6
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '1' AND SW1 = '1' AND SW0 = '0') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
                                    end if;


                                when L3 =>
                                    levelState <= Level_3;

                                    if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level3life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level3life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level3life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base10 17
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                            -- correct guess
```

```
                                        gameState <= CorrectState;
                                    else
                                        gameState <= FailIntermediate;
                                    end if;
                                end if;

                            when L4 =>
                                levelState <= Level_4;

                                if (lifeCounter = 3) then
                                    next_char <=
lcd_display_level4life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                    next_char <=
lcd_display_level4life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                    next_char <=
lcd_display_level4life1(CONV_INTEGER(char_count));
                                else
                                    gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                    guessEntered := true;
                                    -- Number base10 86
                                    if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '1' AND SW1 = '1' AND SW0 = '0') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                    else
                                        gameState <= FailIntermediate;
                                    end if;
                                end if;

                            when L5 =>
                                levelState <= Level_5;

                                if (lifeCounter = 3) then
                                    next_char <=
lcd_display_level5life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                    next_char <=
lcd_display_level5life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
```

```
                                        next_char <=
lcd_display_level5life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base10 112
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '0' AND SW1 = '0' AND SW0 = '0') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
                                    end if;

                                when L6 =>
                                    levelState <= Level_6;

                                    if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level6life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level6life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level6life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base10 341
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='0' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '1') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
```

```
                        end if;

                    when L7 =>
                        levelState <= Level_7;

                        if (lifeCounter = 3) then
                            next_char <=
lcd_display_level7life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                            next_char <=
lcd_display_level7life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                            next_char <=
lcd_display_level7life1(CONV_INTEGER(char_count));
                        else
                            gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                            guessEntered := true;
                            -- Number base10 891
                            if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='1' AND SW8 ='1' AND SW7 ='0' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '1') then
                                -- correct guess
                                gameState <= CorrectState;
                            else
                                gameState <= FailIntermediate;
                            end if;
                        end if;

                    when L8 =>
                        levelState <= Level_8;

                        if (lifeCounter = 3) then
                            next_char <=
lcd_display_level8life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                            next_char <=
lcd_display_level8life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                            next_char <=
lcd_display_level8life1(CONV_INTEGER(char_count));
                        else
                            gameState <= BuggedState;
                        end if;
```

```
                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base10 2368
                                if (SW13 = '0' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='0' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '0' AND SW1 = '0' AND SW0 = '0') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                else
                                        gameState <= FailIntermediate;
                                end if;
                        end if;

                when L9 =>
                        levelState <= Level_9;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level9life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level9life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level9life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base10 5450
                                if (SW13 = '0' AND SW12 ='1' AND SW11 ='0' AND
SW10 ='1' AND SW9 ='0' AND SW8 ='1' AND SW7 ='0' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                else
                                        gameState <= FailIntermediate;
                                end if;
                        end if;

                when L10 =>
                        levelState <= Level_10;
```

```
                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level10life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level10life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level10life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base10 8761
                                if (SW13 = '1' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='1' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                else
                                        gameState <= FailIntermediate;
                                end if;
                        end if;

                when L11 =>
                        levelState <= Level_11;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level11life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level11life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level11life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base16 9
```

```
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L12 =>
                                levelState <= Level_12;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level12life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level12life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level12life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 C
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '1' AND SW1 = '0' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L13 =>
                                levelState <= Level_13;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level13life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
```

```
                                        next_char <=
lcd_display_level13life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level13life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 32
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
                                    end if;

                                when L14 =>
                                    levelState <= Level_14;

                                    if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level14life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level14life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level14life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 1A5
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '1') then
                                            -- correct guess
                                            gameState <= CorrectState;
```

```
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L15 =>
                                levelState <= Level_15;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level15life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level15life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level15life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 1F4
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L16 =>
                                levelState <= Level_16;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level16life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level16life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level16life1(CONV_INTEGER(char_count));
```

```
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 FA2
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='1' AND SW7 ='1' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L17 =>
                                levelState <= Level_17;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level17life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level17life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level17life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 3F8A
                                        if (SW13 = '1' AND SW12 ='1' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='1' AND SW7 ='1' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;
```

```
                when L18 =>
                        levelState <= Level_18;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level18life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level18life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level18life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base16 2EAA
                                if (SW13 = '1' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='0' AND SW7 ='1' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                else
                                        gameState <= FailIntermediate;
                                end if;
                        end if;

                when L19 =>
                        levelState <= Level_19;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level19life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level19life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level19life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
```

```
                                        guessEntered := true;
                                        -- Number base16 2F9A
                                        if (SW13 = '1' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='1' AND SW7 ='1' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L20 =>
                                levelState <= Level_20;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level20life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level20life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level20life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base16 3FFF
                                        if (SW13 = '1' AND SW12 ='1' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '1' AND SW1 = '1' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L21 =>
                                levelState <= Level_21;

                                if (lifeCounter = 3) then
```

```
                                        next_char <=
lcd_display_level21life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level21life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level21life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 5
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '1') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
                                    end if;

                                when L22 =>
                                    levelState <= Level_22;

                                    if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level22life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level22life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level22life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base6 11
```

```
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L23 =>
                                levelState <= Level_23;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level23life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level23life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level23life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base6 53
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L24 =>
                                levelState <= Level_24;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level24life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
```

```
                                        next_char <=
lcd_display_level24life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level24life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 113
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='0' AND SW7 ='0' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '1') then
                                            -- correct guess
                                            gameState <= CorrectState;
                                        else
                                            gameState <= FailIntermediate;
                                        end if;
                                    end if;

                                when L25 =>
                                    levelState <= Level_25;

                                    if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level25life3(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level25life2(CONV_INTEGER(char_count));
                                    elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level25life1(CONV_INTEGER(char_count));
                                    else
                                        gameState <= BuggedState;
                                    end if;

                                    if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 672
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                            -- correct guess
                                            gameState <= CorrectState;
```

```
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L26 =>
                                levelState <= Level_26;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level26life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level26life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level26life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 741
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '0' AND SW1 = '0' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L27 =>
                                levelState <= Level_27;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level27life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level27life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level27life1(CONV_INTEGER(char_count));
```

```
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 1076
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='0' AND SW9 ='1' AND SW8 ='0' AND SW7 ='0' AND SW6 ='0' AND SW5 ='1' AND
SW4 ='1' AND SW3 = '1' AND SW2 = '1' AND SW1 = '1' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when L28 =>
                                levelState <= Level_28;

                                if (lifeCounter = 3) then
                                        next_char <=
lcd_display_level28life3(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 2) then
                                        next_char <=
lcd_display_level28life2(CONV_INTEGER(char_count));
                                elsif (lifeCounter = 1) then
                                        next_char <=
lcd_display_level28life1(CONV_INTEGER(char_count));
                                else
                                        gameState <= BuggedState;
                                end if;

                                if (enterGuess = '1') then
                                        guessEntered := true;
                                        -- Number base8 2745
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='0' AND
SW10 ='1' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='1' AND
SW4 ='0' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '1') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;
```

```
                when L29 =>
                        levelState <= Level_29;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level29life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level29life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level29life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
                                guessEntered := true;
                                -- Number base8 6724
                                if (SW13 = '0' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='0' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='1' AND SW3 = '0' AND SW2 = '1' AND SW1 = '0' AND SW0 = '0') then
                                        -- correct guess
                                        gameState <= CorrectState;
                                else
                                        gameState <= FailIntermediate;
                                end if;
                        end if;

                when L30 =>
                        levelState <= Level_30;

                        if (lifeCounter = 3) then
                                next_char <=
lcd_display_level30life3(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 2) then
                                next_char <=
lcd_display_level30life2(CONV_INTEGER(char_count));
                        elsif (lifeCounter = 1) then
                                next_char <=
lcd_display_level30life1(CONV_INTEGER(char_count));
                        else
                                gameState <= BuggedState;
                        end if;

                        if (enterGuess = '1') then
```

```
                                        guessEntered := true;
                                        -- Number base8 7712
                                        if (SW13 = '0' AND SW12 ='0' AND SW11 ='1' AND
SW10 ='1' AND SW9 ='1' AND SW8 ='1' AND SW7 ='1' AND SW6 ='1' AND SW5 ='0' AND
SW4 ='0' AND SW3 = '1' AND SW2 = '0' AND SW1 = '1' AND SW0 = '0') then
                                                -- correct guess
                                                gameState <= CorrectState;
                                        else
                                                gameState <= FailIntermediate;
                                        end if;
                                end if;

                        when FailIntermediate =>
                                if (lifeCounter > 3) then
                                        lifeCounter := 2;
                                elsif (lifeCounter = 3) then
                                        lifeCounter := 2;
                                elsif (lifeCounter = 2) then
                                        lifeCounter := 1;
                                elsif (lifeCounter = 1) then
                                        lifeCounter := 0;
                                else
                                        lifeCounter := 0;
                                end if;

                                gameState <= FailState;

                        when FailState =>
                                if (lifeCounter = 0) then
                                        gameState <= FinalLoss;
                                end if;

                                next_char <=
lcd_display_levelFail(CONV_INTEGER(char_count));

                                -- Flash red lights
                                LEDR0 <= REDLIGHT_CONTROLLER;
                                LEDR1 <= REDLIGHT_CONTROLLER;
                                LEDR2 <= REDLIGHT_CONTROLLER;
                                LEDR3 <= REDLIGHT_CONTROLLER;
                                LEDR4 <= REDLIGHT_CONTROLLER;
                                LEDR5 <= REDLIGHT_CONTROLLER;
                                LEDR6 <= REDLIGHT_CONTROLLER;
                                LEDR7 <= REDLIGHT_CONTROLLER;
                                LEDR8 <= REDLIGHT_CONTROLLER;
                                LEDR9 <= REDLIGHT_CONTROLLER;
```

```
                LEDR10 <= REDLIGHT_CONTROLLER;
                LEDR11 <= REDLIGHT_CONTROLLER;
                LEDR12 <= REDLIGHT_CONTROLLER;
                LEDR13 <= REDLIGHT_CONTROLLER;
                LEDR14 <= REDLIGHT_CONTROLLER;
                LEDR15 <= REDLIGHT_CONTROLLER;
                LEDR16 <= REDLIGHT_CONTROLLER;
                LEDR17 <= REDLIGHT_CONTROLLER;

                if (delay10sIsOver) then
                        guessEntered := false;
                        -- Turn off all the lights
                        LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <= '0'; LEDR4 <= '0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                        LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13 <= '0'; LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                        LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <= '0'; LEDG4 <= '0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                        CASE (levelState) IS

                                when Level_1 =>
                                        gameState <= L1;
                                when Level_2 =>
                                        gameState <= L2;
                                when Level_3 =>
                                        gameState <= L3;
                                when Level_4 =>
                                        gameState <= L4;
                                when Level_5 =>
                                        gameState <= L5;
                                when Level_6 =>
                                        gameState <= L6;
                                when Level_7 =>
                                        gameState <= L7;
                                when Level_8 =>
                                        gameState <= L8;
                                when Level_9 =>
                                        gameState <= L9;
                                when Level_10 =>
                                        gameState <= L10;
                                when Level_11 =>
                                        gameState <= L11;
                                when Level_12 =>
                                        gameState <= L12;
                                when Level_13 =>
```

```
                                        gameState <= L13;
                                    when Level_14 =>
                                        gameState <= L14;
                                    when Level_15 =>
                                        gameState <= L15;
                                    when Level_16 =>
                                        gameState <= L16;
                                    when Level_17 =>
                                        gameState <= L17;
                                    when Level_18 =>
                                        gameState <= L18;
                                    when Level_19 =>
                                        gameState <= L19;
                                    when Level_20 =>
                                        gameState <= L20;
                                    when Level_21 =>
                                        gameState <= L21;
                                    when Level_22 =>
                                        gameState <= L22;
                                    when Level_23 =>
                                        gameState <= L23;
                                    when Level_24 =>
                                        gameState <= L24;
                                    when Level_25 =>
                                        gameState <= L25;
                                    when Level_26 =>
                                        gameState <= L26;
                                    when Level_27 =>
                                        gameState <= L27;
                                    when Level_28 =>
                                        gameState <= L28;
                                    when Level_29 =>
                                        gameState <= L29;
                                    when Level_30 =>
                                        gameState <= L30;
                                    when others =>
                                        gameState <= BuggedState;

                                end case;

                            end if;

                        when CorrectState =>
                            -- Flash green lights
                            LEDG0 <= GREENLIGHT_CONTROLLER;
                            LEDG1 <= GREENLIGHT_CONTROLLER;
```

```
                                LEDG2 <= GREENLIGHT_CONTROLLER;
                                LEDG3 <= GREENLIGHT_CONTROLLER;
                                LEDG4 <= GREENLIGHT_CONTROLLER;
                                LEDG5 <= GREENLIGHT_CONTROLLER;
                                LEDG6 <= GREENLIGHT_CONTROLLER;
                                LEDG7 <= GREENLIGHT_CONTROLLER;

                                next_char <=
lcd_display_levelPass(CONV_INTEGER(char_count));

                                if (delay3sIsOver) then
                                        guessEntered := false;
                                        -- Turn off all the lights
                                        LEDR0 <= '0'; LEDR1 <= '0'; LEDR2 <= '0'; LEDR3 <=
'0'; LEDR4 <= '0'; LEDR5 <= '0'; LEDR6 <= '0'; LEDR7 <= '0'; LEDR8 <= '0'; LEDR9 <= '0';
                                        LEDR10 <= '0'; LEDR11 <= '0'; LEDR12 <= '0'; LEDR13
<= '0'; LEDR14 <= '0'; LEDR15 <= '0'; LEDR16 <= '0'; LEDR17 <= '0';
                                        LEDG0 <= '0'; LEDG1 <= '0'; LEDG2 <= '0'; LEDG3 <=
'0'; LEDG4 <= '0'; LEDG5 <= '0'; LEDG6 <= '0'; LEDG7 <= '0';

                                        CASE (levelState) IS

                                                when Level_1 =>
                                                        gameState <= L2;
                                                when Level_2 =>
                                                        gameState <= L3;
                                                when Level_3 =>
                                                        gameState <= L4;
                                                when Level_4 =>
                                                        gameState <= L5;
                                                when Level_5 =>
                                                        gameState <= L6;
                                                when Level_6 =>
                                                        gameState <= L7;
                                                when Level_7 =>
                                                        gameState <= L8;
                                                when Level_8 =>
                                                        gameState <= L9;
                                                when Level_9 =>
                                                        gameState <= L10;
                                                when Level_10 =>
                                                        gameState <= L11;
                                                when Level_11 =>
                                                        gameState <= L12;
                                                when Level_12 =>
                                                        gameState <= L13;
```

```
                when Level_13 =>
                        gameState <= L14;
                when Level_14 =>
                        gameState <= L15;
                when Level_15 =>
                        gameState <= L16;
                when Level_16 =>
                        gameState <= L17;
                when Level_17 =>
                        gameState <= L18;
                when Level_18 =>
                        gameState <= L19;
                when Level_19 =>
                        gameState <= L20;
                when Level_20 =>
                        gameState <= L21;
                when Level_21 =>
                        gameState <= L22;
                when Level_22 =>
                        gameState <= L23;
                when Level_23 =>
                        gameState <= L24;
                when Level_24 =>
                        gameState <= L25;
                when Level_25 =>
                        gameState <= L26;
                when Level_26 =>
                        gameState <= L27;
                when Level_27 =>
                        gameState <= L28;
                when Level_28 =>
                        gameState <= L29;
                when Level_29 =>
                        gameState <= L30;
                when Level_30 =>
                        gameState <= FinalWin;
                when others =>
                        gameState <= BuggedState;

            end case;

        end if;

    when FinalWin =>
        if (lifeCounter > 3) then
            -- turn on the green lights
```

```
                                        LEDG0 <= '1'; LEDG1 <= '1'; LEDG2 <= '1'; LEDG3 <=
'1'; LEDG4 <= '1'; LEDG5 <= '1'; LEDG6 <= '1'; LEDG7 <= '1';

                                        next_char <=
lcd_display_PERFECTFinalWin(CONV_INTEGER(char_count));
                                    else
                                        -- turn on the green lights
                                        LEDG0 <= '1'; LEDG1 <= '1'; LEDG2 <= '1'; LEDG3 <=
'1'; LEDG4 <= '1'; LEDG5 <= '1'; LEDG6 <= '1'; LEDG7 <= '1';

                                        next_char <=
lcd_display_finalWin(CONV_INTEGER(char_count));
                                    end if;

                                when FinalLoss =>
                                    CASE (levelState) IS

                                        when Level_1 =>
                                            -- turn on the red lights
                                            LEDR0 <= '1'; LEDR1 <= '1'; LEDR2 <= '1';
LEDR3 <= '1'; LEDR4 <= '1'; LEDR5 <= '1'; LEDR6 <= '1'; LEDR7 <= '1'; LEDR8 <= '1';
LEDR9 <= '1';
                                            LEDR10 <= '1'; LEDR11 <= '1'; LEDR12 <= '1';
LEDR13 <= '1'; LEDR14 <= '1'; LEDR15 <= '1'; LEDR16 <= '1'; LEDR17 <= '1';

                                            next_char <=
lcd_display_LOSTONFIRSTROUND(CONV_INTEGER(char_count));

                                        when others =>
                                            -- turn on the red lights
                                            LEDR0 <= '1'; LEDR1 <= '1'; LEDR2 <= '1';
LEDR3 <= '1'; LEDR4 <= '1'; LEDR5 <= '1'; LEDR6 <= '1'; LEDR7 <= '1'; LEDR8 <= '1';
LEDR9 <= '1';
                                            LEDR10 <= '1'; LEDR11 <= '1'; LEDR12 <= '1';
LEDR13 <= '1'; LEDR14 <= '1'; LEDR15 <= '1'; LEDR16 <= '1'; LEDR17 <= '1';

                                            next_char <=
lcd_display_finalLoss(CONV_INTEGER(char_count));
                                    end case;

                                when others =>
                                    gameState <= ResetState;

                                end case;

                        end if;
```

```
END PROCESS;


--
=============================================================================
--
--======================== CLOCK #1 SIGNALS
=============================--
--
=============================================================================
--
process(clock_50)
begin
     if (rising_edge(clock_50)) then
        if (reset = '0') then
           clk_count_400hz <= x"00000";
           clk_400hz_enable <= '0';
        else
           if (clk_count_400hz <= x"0F424") then
                 clk_count_400hz <= clk_count_400hz + 1;
                 clk_400hz_enable <= '0';
           else
                 clk_count_400hz <= x"00000";
                 clk_400hz_enable <= '1';
           end if;
        end if;
     end if;
end process;
--=============================================================================--


--======================== LCD DRIVER CORE
================================--
--                  STATE MACHINE WITH RESET                        --
--=================================================-----
===============--
process (clock_50, reset)
begin

        if reset = '0' then
           state <= reset1;
           data_bus_value <= x"38"; -- RESET
```

```
            next_command <= reset2;
            lcd_e <= '1';
            lcd_rs <= '0';
            lcd_rw_int <= '0';


        elsif rising_edge(clock_50) then
            if clk_400hz_enable = '1' then


            --=========================================================--
            -- State Machine to send commands and data to LCD DISPLAY
            --=========================================================--
              case state is
              -- Set Function to 8-bit transfer and 2 line display with 5x8 Font size
              -- see Hitachi HD44780 family data sheet for LCD command and timing details


--======================= INITIALIZATION START ============================--
                    when reset1 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38"; -- EXTERNAL RESET
                        state <= drop_lcd_e;
                        next_command <= reset2;
                        char_count <= "00000";

                    when reset2 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38"; -- EXTERNAL RESET
                        state <= drop_lcd_e;
                        next_command <= reset3;

                    when reset3 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38"; -- EXTERNAL RESET
                        state <= drop_lcd_e;
```

```
                    next_command <= func_set;

                -- Function Set
                --==============--
                when func_set =>
                    lcd_e <= '1';
                    lcd_rs <= '0';
                    lcd_rw_int <= '0';
                    data_bus_value <= x"38";  -- Set Function to 8-bit transfer, 2 line display and a
5x8 Font size
                    state <= drop_lcd_e;
                    next_command <= display_off;


                -- Turn off Display
                --==============--
                when display_off =>
                    lcd_e <= '1';
                    lcd_rs <= '0';
                    lcd_rw_int <= '0';
                    data_bus_value <= x"08"; -- Turns OFF the Display, Cursor OFF and Blinking
Cursor Position OFF.....(0F = Display ON and Cursor ON, Blinking cursor position ON)
                    state <= drop_lcd_e;
                    next_command <= display_clear;


                -- Clear Display
                --==============--
                when display_clear =>
                    lcd_e <= '1';
                    lcd_rs <= '0';
                    lcd_rw_int <= '0';
                    data_bus_value <= x"01"; -- Clears the Display
                    state <= drop_lcd_e;
                    next_command <= display_on;


                -- Turn on Display and Turn off cursor
                --==================================--
                when display_on =>
                    lcd_e <= '1';
                    lcd_rs <= '0';
                    lcd_rw_int <= '0';
```

```
                        data_bus_value <= x"0C"; -- Turns on the Display (0E = Display ON, Cursor
ON and Blinking cursor OFF)
                        state <= drop_lcd_e;
                        next_command <= mode_set;

                    -- Set write mode to auto increment address and move cursor to the right
                    --
=================================================================-
-
                    when mode_set =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"06"; -- Auto increment address and move cursor to the
right
                        state <= drop_lcd_e;
                        next_command <= print_string;

--======================= INITIALIZATION END
============================--



--
=================================================================
==--
--              Write ASCII hex character Data to the LCD
--
=================================================================
==--
                    when print_string =>
                        state <= drop_lcd_e;
                        lcd_e <= '1';
                        lcd_rs <= '1';
                        lcd_rw_int <= '0';

                          -- ASCII character to output
                          if (next_char(7 downto 4) /= x"0") then
                            data_bus_value <= next_char;
                          else

                            -- Convert 4-bit value to an ASCII hex digit
                            if next_char(3 downto 0) >9 then
```

```
                        -- ASCII A...F
                          data_bus_value <= x"4" & (next_char(3 downto 0)-9);
                        else

                        -- ASCII 0...9
                          data_bus_value <= x"3" & next_char(3 downto 0);
                        end if;
                      end if;

                    state <= drop_lcd_e;


                    -- Loop to send out 32 characters to LCD Display (16 by 2 lines)
                      if (char_count < 31) AND (next_char /= x"fe") then
                          char_count <= char_count +1;
                      else
                          char_count <= "00000";
                      end if;

                    -- Jump to second line?
                      if char_count = 15 then
                        next_command <= line2;

                    -- Return to first line?
                      elsif (char_count = 31) or (next_char = x"fe") then
                            next_command <= return_home;
                      else
                            next_command <= print_string;
                      end if;

                  -- Set write address to line 2 character 1
                  --======================================--
                  when line2 =>
                      lcd_e <= '1';
                      lcd_rs <= '0';
                      lcd_rw_int <= '0';
                      data_bus_value <= x"c0";
                      state <= drop_lcd_e;
                      next_command <= print_string;


                  -- Return write address to first character position on line 1
                  --=========================================================-
-
                  when return_home =>
```

```
                    lcd_e <= '1';
                    lcd_rs <= '0';
                    lcd_rw_int <= '0';
                    data_bus_value <= x"80";
                    state <= drop_lcd_e;
                    next_command <= print_string;


                -- lcd_e will match clk_CUSTOM_hz_enable line when instructed to go LOW,
however, if the clk_CUSTOM_hz_enable source clock must be a lower count value or it will
reset LOW anyhow.
                -- The next states occur at the end of each command or data transfer to the LCD
                -- Drop LCD E line - falling edge loads inst/data to LCD controller
                --
============================================================================
=======--
                when drop_lcd_e =>
                    lcd_e <= '0';
                    lcd_blon <= '1';
                    lcd_on   <= '1';
                    state <= hold;

                -- Hold LCD inst/data valid after falling edge of E line
                --=================================================--
                when hold =>
                    state <= next_command;
                    lcd_blon <= '1';  -- important
                    lcd_on   <= '1';  -- important
                end case;


            end if;-- CLOSING STATEMENT FOR "IF clk_400hz_enable = '1' THEN"

        end if;-- CLOSING STATEMENT FOR "IF reset = '0' THEN"

end process;

END ARCHITECTURE Binverter_arch;
```