**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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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"3A",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 < "1011111010111100001000000" 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 < "1011111010111100001000000" 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;