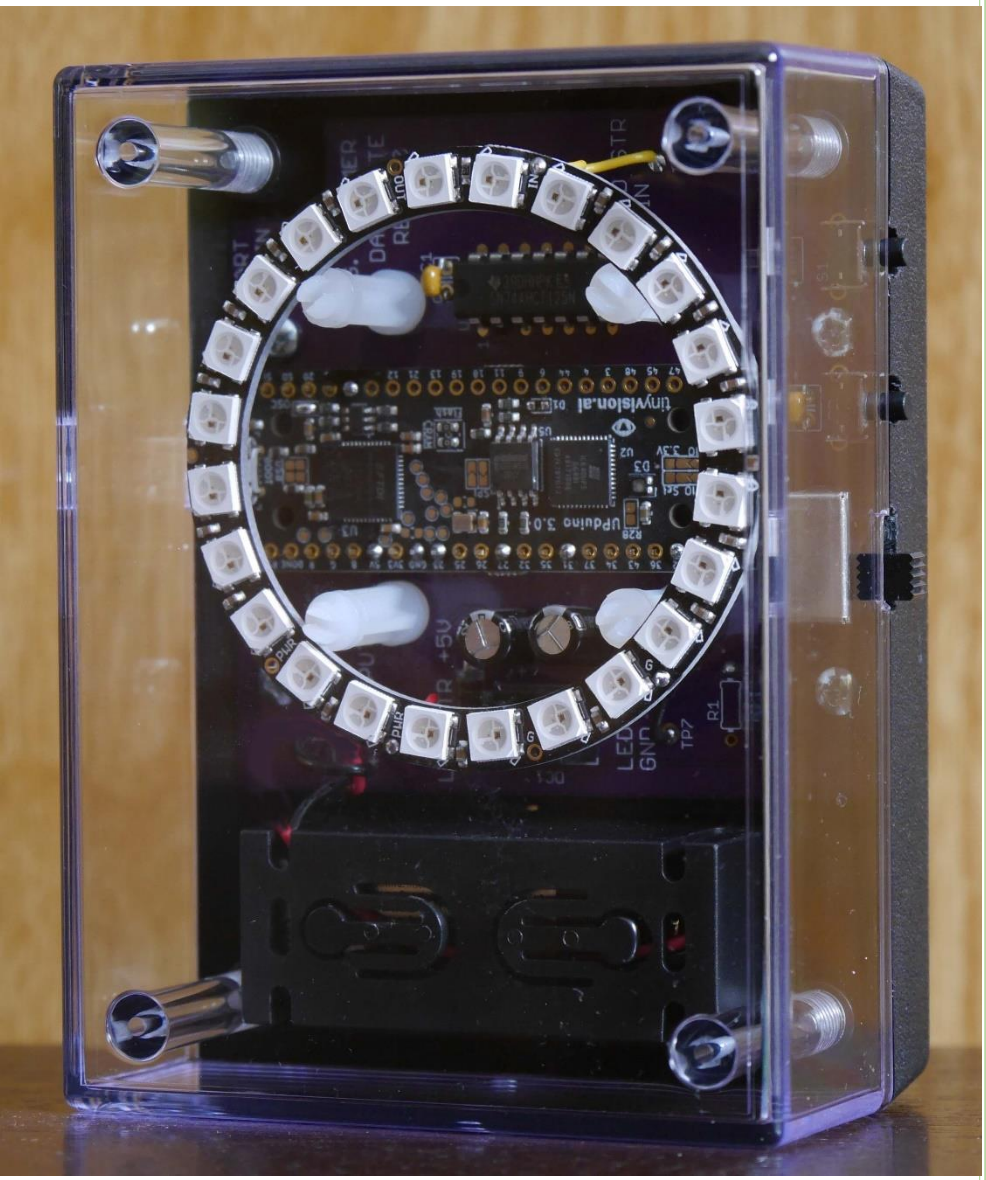
Dazzelite User's Manual



Revision: A.0

Grant Grummer

2/21/2021

Dazzelite User's Manual

Table of Contents

Dazzelite User’s Manual

Introduction

Welcome to the Dazzelite User’s Manual. Dazzelite can be built in many ways. One can build a version on a breadboard, a perfboard or a custom PCB. The perfboard and custom PCB versions can be mounted into a case, making the unit portable. This document is written for the custom PCB version, mounted into a beautiful, portable case with a built-in battery compartment.

Dazzelite uses a tri-color LED ring to produce different light displays. Dazzelite is programmable, so the displays are only limited by your imagination. No previous software knowledge is required. A set of default light displays along with an introduction to Dazzelite can be viewed at the following link:

https://vimeo.com/516437596

A small set of display specific instructions are used to create different display patterns. There are up to 16 possible instructions. Not all of these instructions are defined at the time of this writing. These are available for those who wish to augment the functionality of this project.

The design contains all the electronic hardware needed to drive the LED ring. The brain of the hardware is a Lattice iCE40 Ultra Plus 5K FPGA (Field Programmable Gate Array). It, along with programming and other support logic, is on the UPduino v3.0 low cost daughter board. It’s mounted onto the Dazzelite board. The Dazzelite board also contains switches, power supply and interface hardware. Dazzelite can be powered form the micro USB connector on the daughter board or an external source.

Dazzelite is controlled via three switches. The power slide switch enables/disables power from the external source or enables/disables the FPGA if powered from micro USB connector. The upper pushbutton switch selects one of four display pattern instruction sets. Each set can have up to 256 instructions. The lower pushbutton switch selects different brightness levels.

Dazzelite Design Environment

In order to program Dazzelite, one must first setup a design environment. Refer to the “Dazzelite Installation and Setup” document for instruction on how to set it up. The rest of this document assumes this setup has been completed.

Building the Design

The Lattice Radiant design environment is used to build the design. A full build is performed by clicking on the “Run All” ▶ triangle.   The boxes to the right of the “Run All” triangle all turn green (with a check mark) once the design has been built. See Figure 1.

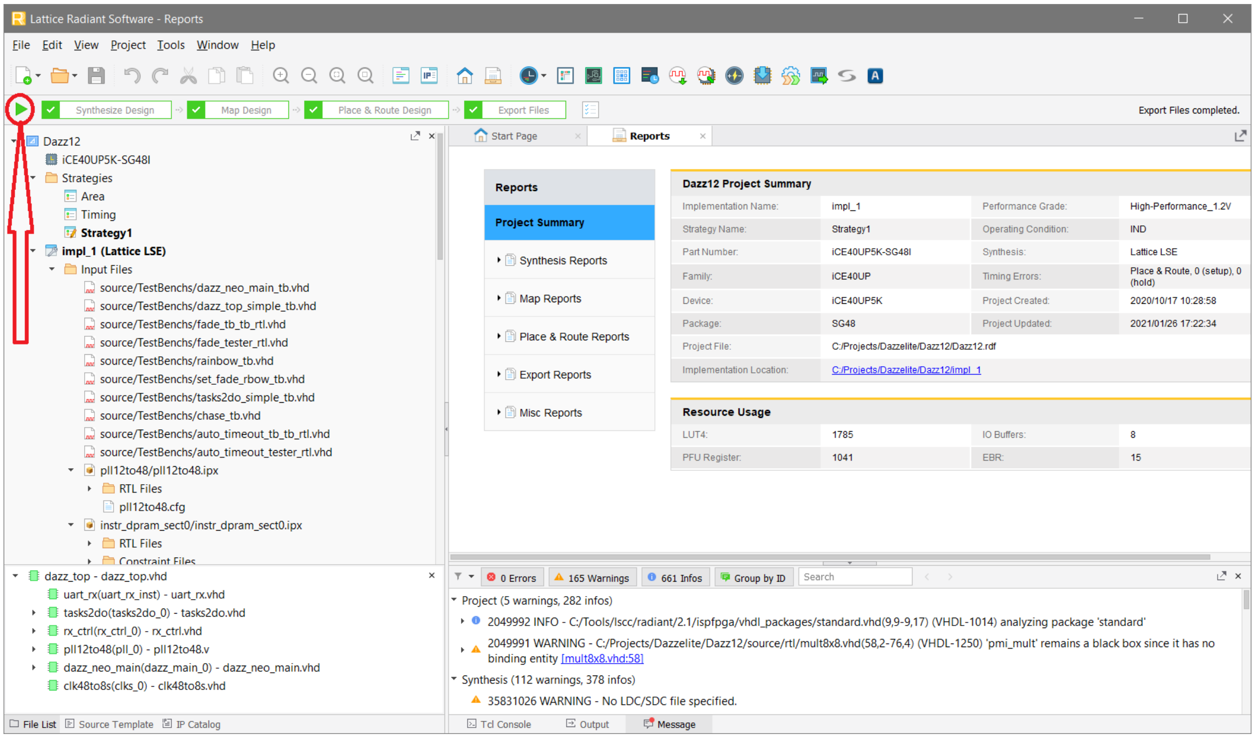


Figure 1: Run All

Custom Displays

A great way to get started with making your own custom displays is to build new display instruction sets. One doesn't need to know anything about FPGAs or software to do this. Use the supplied "Dazzelite_Instruction" spreadsheet to create each instruction. Use a text editor to combine the new instructions into a set of instructions. Refer to the supplied example display instruction sets, in the "Patterns" folder, to see how this looks. Remember to end each instruction set with the Finish instruction!

There are 4 memories in Dazzelite, each contain one display pattern. They are selected via the pattern pushbutton switch. Each memory can be loaded with a new pattern by double clicking on inst_dpram_sect0, 1, 2 or 3 as shown in Figure 2.

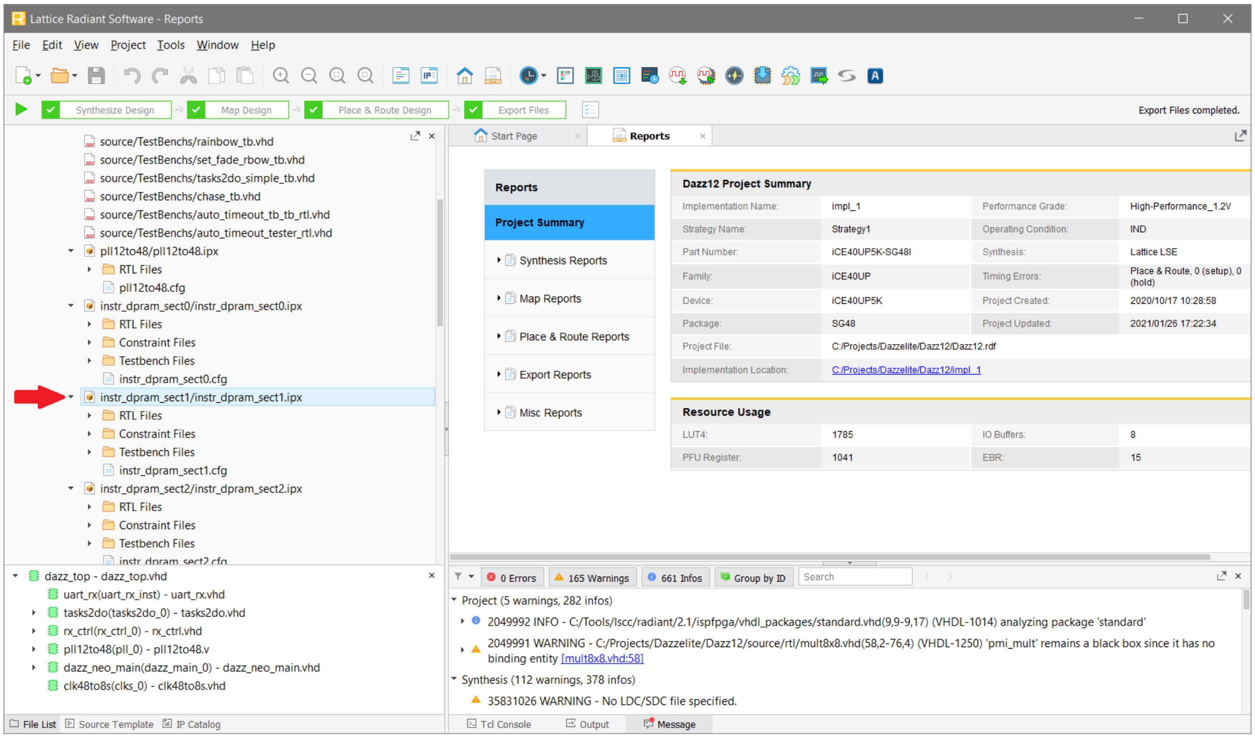


Figure 2: Changing Display Patterns

Replace the current instruction set with your new instruction set by clicking on "…" and navigating to that instruction as highlighted in Figure 3.

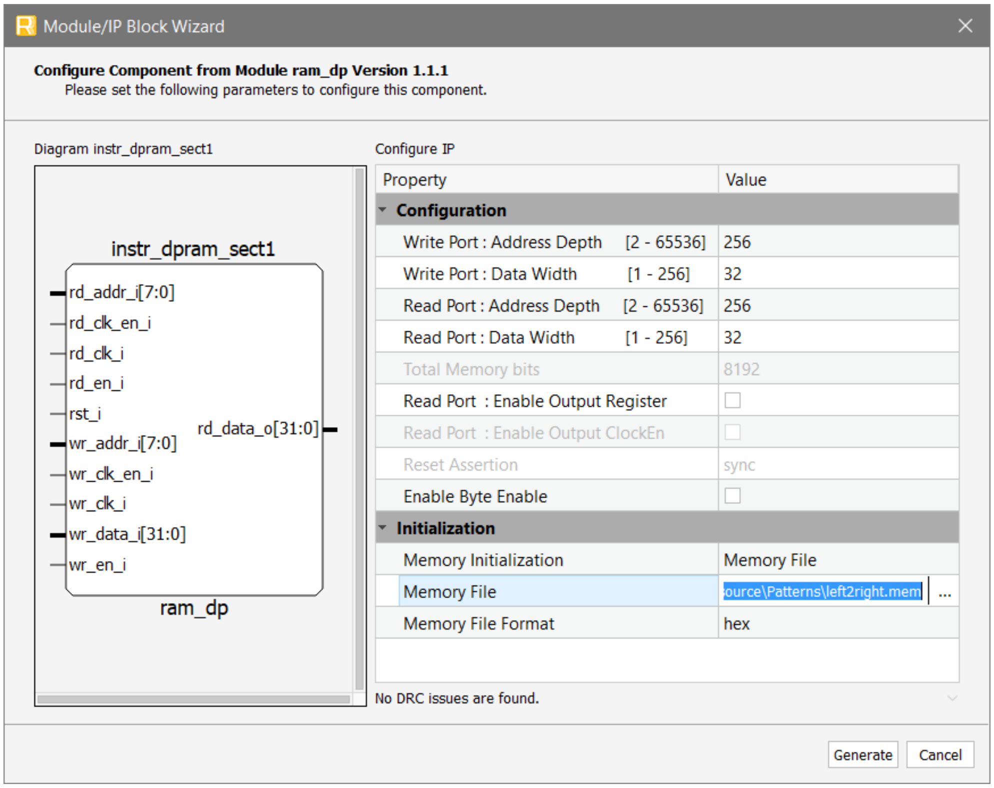


Figure 3: Replacing Display Pattern

Click in the "Generate" button (at the bottom of the window) to replace the display pattern. After that completes, re-build the design as outlined in the Building the Design section.

Programming Dazzelite

Dazzelite can be programmed from the Radiant program or via a standalone programmer.

Programmer Access Through Radiant

From Radiant, click on the programming icon as shown in Figure 4.

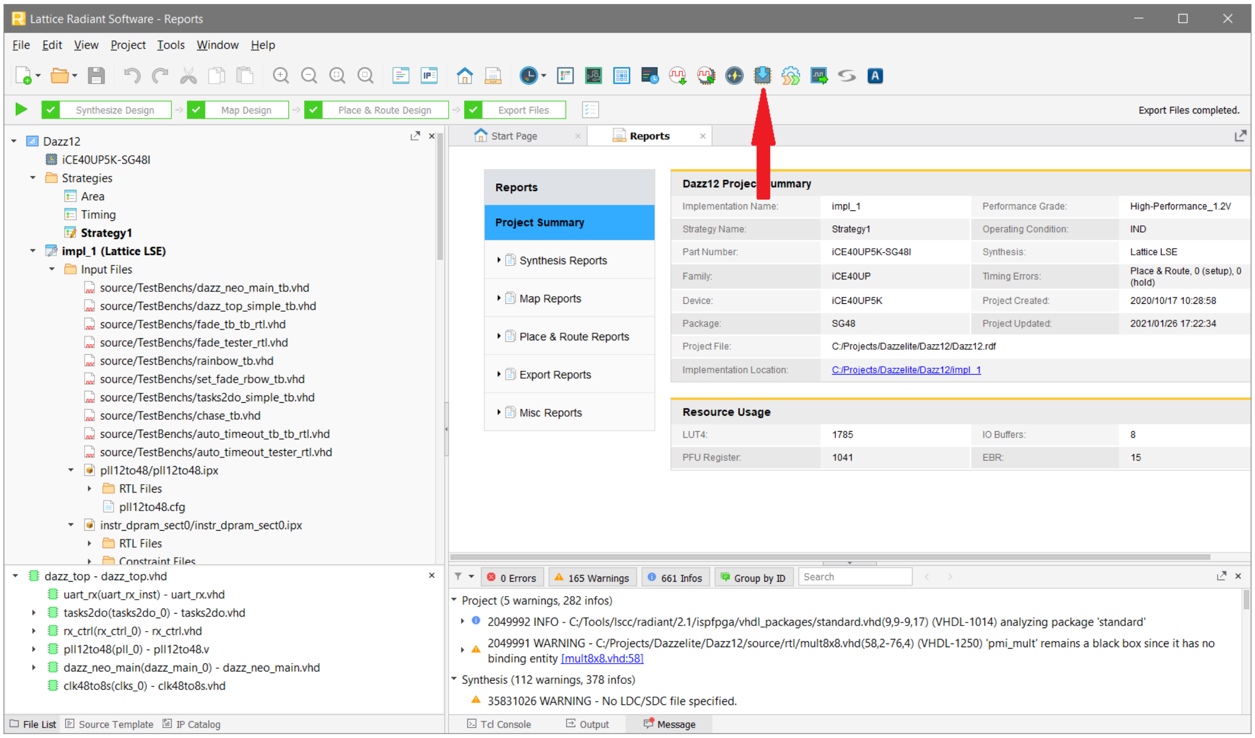


Figure 4: Programming Icon

Standalone Programmers

There are many programmers that can be used to program Dazzelite. One is the Radiant [Programmer Standalone] and another is the Diamond [Programmer Standalone]. They're both very similar.

Programming Steps

The following steps can be used with the Radiant Programmer (either via Radiant or standalone) to program Dazzelite. The Diamond Programmer steps may be slightly different.

1. Connect the micro USB port on the UPDuino v3.0 to your PC
2. Open the Radiant Programmer
3. Click `Detect Cable` then `OK`
4. After scanning, select `Generic JTAG Device` and `Select iCE40 UltraPlus`
5. Under `Device` click iCE40UP3K and change it to iCE40UP5K
6. Under `Operation` double click `Fast Program` and change `Target Memory: ` to `External SPI Flash Memory`
7. Select your `*.hex` programming file under `Programming file`.
8. Configure the following `SPI Flash Options`
   a. Winbond
   b. W25Q32JV
   c. 208mil 8-pin SOIC
9. Click `Load from File` under `SPI Programming` to get load size
10. Click OK
11. Click `Run` -> `Program Device`

Instructions

The Dazzelite_Instruction spreadsheet provides an easy way to create instructions. It allows instructions to be build using decimal number and automatically provides the hexadecimal value (of the instruction) needed by the FPGA. Optionally, a detailed look at each bit, in each instruction, is provided in the rest of this section.

As previously noted, there are up to 16 possible instructions. Some are specifically tailored for the RGB LED ring. Instructions are 32-bits long. All instructions start with a command. Some commands are followed by additional fields. A detailed description of each instruction is presented in the following sections. Note all bits of the instructions are zeroed at powerup unless otherwise noted.

Instruction 0: Blackout Display

Blackouts the entire display.

31 28	27				0
CMD0	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD0	Blackouts all the LEDs in the string(s). All bits must be zero.
[27:0]	Unused	Not defined

Instruction 1: Delay

Adds a programmable time delay. Each additional count adds 5.12 mSeconds. The maximum count is 255 (0xFF). A count a zero creates a random delay.

31 28	27 20	19			0
CMD1	DLY	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD1	Programmable delay. Value = 1
[27:20]	DLY	Delay count. Max = 255 (0xFF). All zeros create a random count.
[19:0]	Unused	Not defined

Instruction 2: Finish

All instruction sets must end with a finish command!

31 28	27				0
CMD2	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function

[31:28]	CMD2	Informs the FPGA that the present instruction set is finished. Value = 2
[27:0]	Unused	Not defined

Instruction 3: Set Random LED

Used to set one LED at a random location/address and/or color. The random color is selected from the 16 colors in the color palette (see Instruction 9: Set Color Palette).

<table>
<tr><td>31 28</td><td>27|26| 25 20</td><td>19 12</td><td>11</td><td></td><td>1 | 0</td></tr>
<tr><td>CMD3</td><td>U | R | ADDR</td><td>BRITE</td><td>Unused</td><td>Unused</td><td>Unused|N</td></tr>
</table>

Bits	**Name**	**Function**
[31:28]	CMD3	Sets one LED. Value = 3
[27]	U	Unused, not defined
[26]	R	Random LED address 0 = bits [25:0] specify address of LED 1 = random LED address generated
[25:20]	ADDR	Specific LED address when [26] = 0. Values 0 to 63 (0x3F)
[19:12]	BRITE	Set the brightness of the LED. Values 0 to 255 (0xFF)
[11:1]	Unused	Not defined
[0]	N	0 = use the same color as used in the last CMD3. That color was the one last selected from the color palette. 1 = use the next color in the color palette.

Instruction 4: Set LED

Used to set one LED's address and color.

<table>
<tr><td>31 28</td><td>27 26 | 25 20</td><td>19 12</td><td>11 8</td><td>7 4</td><td>3 0</td></tr>
<tr><td>CMD4</td><td>U | ADDR</td><td>BRITE</td><td>BLUE</td><td>GREEN</td><td>RED</td></tr>
</table>

Bits	**Name**	**Function**
[31:28]	CMD4	Sets one LED. Value = 4
[27:26]	U	Unused, not defined
[25:20]	ADDR	Specific LED address. Values 0 to 63 (0x3F)
[19:12	BRITE	Set the brightness of the LED. Values 0 to 255 (0xFF)
[11:8]	BLUE	Sets the amount of blue. Value 0 to 15 (0xF)
[7:4]	GREEN	Sets the amount of green. Value 0 to 15 (0xF)
[3:0]	RED	Sets the amount of red. Value 0 to 15 (0xF)

Instruction 5: Chase

Adds an offset to all LED addresses. Increments or decrements that address along with their brightness and color. The offset wraps around at maximum numbers of LEDs per string.

31 28	27 26 | 25 20	19 14|13|12	11		0
CMD5	U | SIZE	Unused | Z | ID	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD5	Increments or decrements every LED's address by the offset size. Value = 5
[27:26]	U	Not defined
[25:20]	SIZE	Offset size. A size of 0 = 1. All others = number entered
[19:14]	Unused	Not defined
[13]	Z	0 = do not zero LEDs address offset 1 = zeros LEDs address offset (ID value ignored)
[12]	ID	0 = decrement offset 1 = increment offset
[11:0]	Unused	Not defined

Instruction 6: Left to Right and Right to Left

Two strings of colors start at the top of the display and make their way to the bottom then back up to the top. As they merge (at the bottom), They are added together and the resulting color is displayed.

31 28	27 23 | 22 20	19 12	11		0
CMD6	Unused | QTY	BRITE	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD6	Runs the algorithm. Value = 6
[27:23]	Unused	Not defined
[22:20]	QTY	Quantity of lights, 1 to 8 per side
[19:12]	BRITE	Set the brightness of the LED. Values 0 to 255 (0xFF)
[11:0]	Unused	Not defined

Instruction 7: Fade In and Out

Fades all LEDs brightness. Increments and then decrements all LEDs brightness.

31 28	27 26|25 24|23 22|21 20	19 13|12	11		0
CMD7	U | LLIM| ULIM| SPD	Unused | EF	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD7	Increments or decrements every LED's address by one. Value = 7
[27:26]	U	Unused, not defined
[25:24]	LLIM	Limits how low the brightness value can go
[23:22]	ULIM	Limits how high the brightness value can go
[21:20]	SPD	Speeds up the incrementing and decrementing
[19:13]	Unused	Not defined

[12]	EF	0 = disable fading once decrement reaches its lowest value 1 = enables fading starting at its lowest value
[11:0]	Unused	Not defined

Instruction 8: Loop

Loop the previous x number of instructions.

31 28	27 20	19 12	11		0
CMD8	LOOPS	QTY	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD8	Repeats previous instructions. Value = 8
[27:20]	LOOPS	Sets the number of times a loop is executed. A value of 255 (0xFF) is a continuous loop.
[19:12]	QTY	Quantity of instruction to repeat. Subtract the start instruction from the loop instruction line number.
[11:0]	Unused	Not defined

Instruction 9: Set Color Palette

Storage location for 16 definable colors. Used in conjunction with Instruction 3: Set Random LED. A set of predefined colors are loaded at powerup. Use this instruction to change them.

31 28	27 24 | 23 20	19 12	11 8	7 4	3 0
CMD9	Unused | CNUM	Unused	BLUE	GREEN	RED

Bits	Name	Function
[31:28]	CMD9	Sets one LED. Value = 9
[27:24]	Unused	Not defined
[23:20]	CNUM	Specific palette address. Values 0 to 15 (0xF)
[19:12]	Unused	Not defined
[11:8]	BLUE	Sets the amount of blue. Value 0 to 15 (0xF)
[7:4]	GREEN	Sets the amount of green. Value 0 to 15 (0xF)
[3:0]	RED	Sets the amount of red. Value 0 to 15 (0xF)

Instruction 10 (0xA): Reserved

Do not use. It's used in other versions of this project.

31 28	27				0
CMD10	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD10	Reserved. Value = 10 (0xA)
[27:0]	Unused	Not defined

Instruction 11 (0xB): Rainbow

Transitions through the 12-bit color spectrum.

31 28	27	13|12	11		0
CMD11	Unused	Unused |ER	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD11	Increments or decrements every LED's address by one. Value = 11 (0xB)
[27:13]	Unused	Not defined
[12]	ER	0 = disable the transitions 1 = enables the transitions
[11:0]	Unused	Not defined

Instruction 12 (0xC): Reserved

Do not use. It's used in other versions of this project.

31 28	27				0
CMD12	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD12	Reserved. Value = 12 (0xC)
[27:0]	Unused	Not defined

Instruction 13 (0xD): Automatic Timeout

Automatically turns the LEDs on and off according to the following settings. The instruction set in use before the time "off" will resume once the time "on" is attained. One should set this up before issuing display instructions.

31 28	27 25 | 24 20	19 17 | 16 12	11	5	4 0
CMD13	Unused | HON	Unused | HOFF	Unused	Unused	HDLY

Bits	Name	Function
[31:28]	CMD13	Sets one LED. Value = 13 (0xD)
[27:25]	Unused	Not defined
[24:20]	HON	Hours on. "0" is always on. Values 0 to 31 (0x1F)
[19:17]	Unused	Not defined
[16:12]	HOFF	Hours off. "0" is once off stay off. Value 0 to 31 (0x1F)
[11:5]	Unused	Not defined
[4:0]	HDLY	Hours delayed (off). Value 0 to 31 (0x1F)

Instruction 14 (0xE): Unused

Available for a new instruction.

31 28	27				0
CMD14	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD14	Unused but available. Value = 14 (0xE)
[27:0]	Unused	Not defined

Instruction 15 (0xF): Unused

Available for a new instruction.

31 28	27				0
CMD15	Unused	Unused	Unused	Unused	Unused

Bits	Name	Function
[31:28]	CMD15	Unused but available. Value = 15 (0xF)
[27:0]	Unused	Not defined

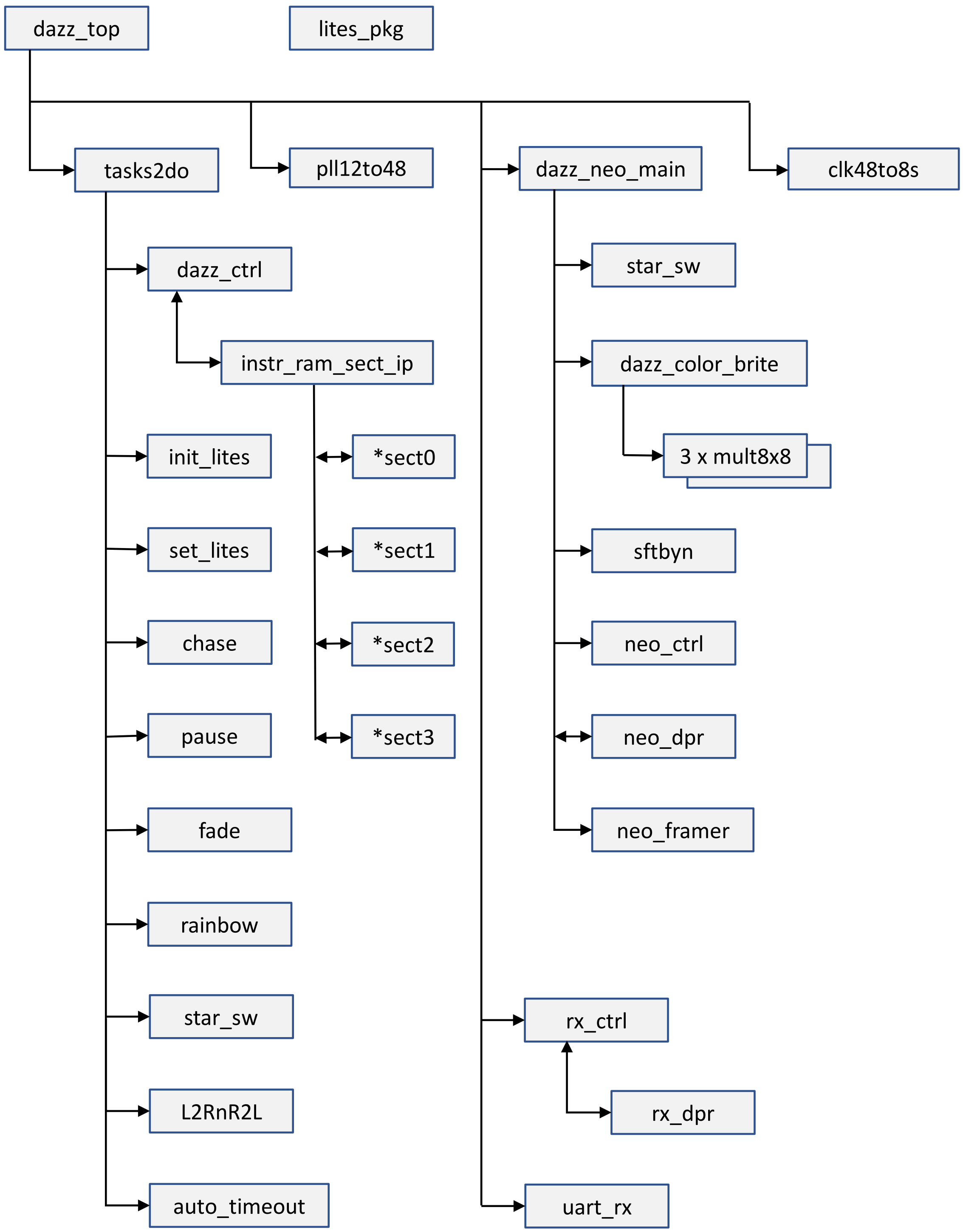
Code Base

The code base consists of mostly VHDL modules, some Verilog modules, instantiated and inferred IPs. There are 5 instantiated cores.

One instantiated core is a PLL (Phase Lock Loop). It's used for creating a 48 MHz clock from the 12 MHz oscillator on the daughter board. A separate module divides the 48 MHz clock down to the 8 MHz and 800 kHz clocks used in the design. The majority of the design runs at 800 kHz.

The 4 remain cores are 256 x 32 memories. Each core holds an instruction set for a display pattern.

Block Diagram of Dazzelite Code



UART

A UART can be used to temporarily download a new display pattern to Dazzelite. The pattern gets loaded into the first display's memory. It's remains in that memory until power is turned off.

The UART uses the following protocol.

- 9600 baud rate
- One start bit
- Eight hex data bits
- One stop bit

Each transfer consists of one bite. Four transfers are required to make one instruction. Multiple sets of four instructions, following each other, make up a Dazzelite instruction set.

The protocol has been used by an old Digi XBee wireless board. The newer boards "appear" to still support this protocol. This has not been tested with new boards.

By adding a wireless board to Dazzelite, one can use a remote wireless board to communicate with it. The UART pins are used to connect Dazzelite to the local wireless board. Digi's XCTU software can be used with Digi XBee boards to remotely send a new display pattern.

Some UART programs support sending hex data, though it's not common. These can be used to directly sent a new temporary display pattern to Dazzelite using a USB to serial (3.3V) cable.