

# Lab 3 - Si5351 Clock Generator

## Si5351 Clock Generator

This lab involves testing the Si5351 (which has an i2c interface).

### 1. Introduction

#### 1.1 I2C Interface to Si5351

The i2c interface is simple, ensure that the SDA and SCL lines of the Si5351 are connected to an appropriate port on the Beaglebone. In the example below, I used /dev/i2c-2.

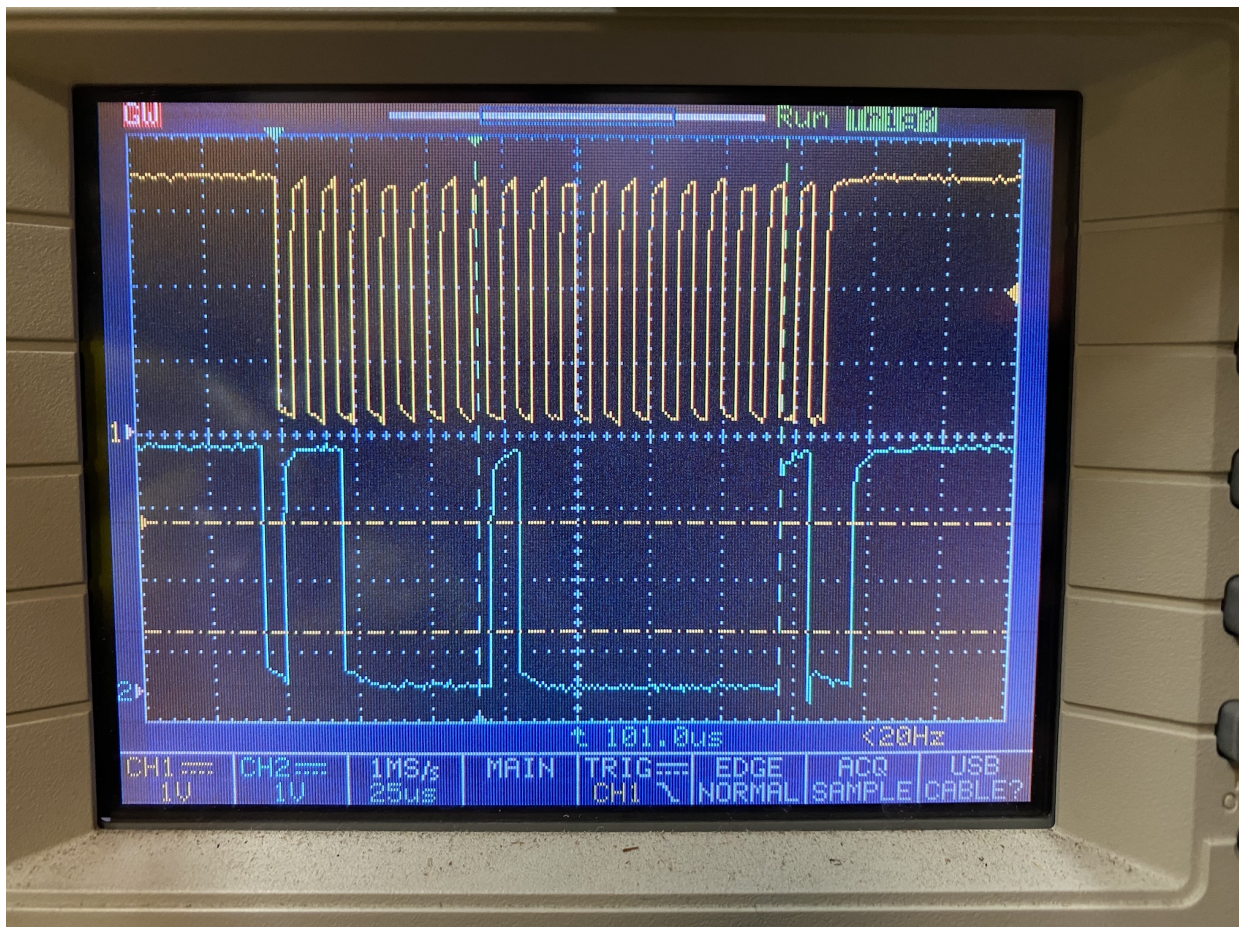
Then make sure all the software required is installed on the BBG.

```
sudo apt update
sudo apt-get install libi2c-dev
```

The commands below show all the i2c devices and that the Si5351 at address 0x60 is responding to the probe on I2C2 (i2c chip 2):

```
debian@beaglebone:~$ i2cdetect -l
i2c-1  i2c          OMAP I2C adapter          I2C adapter
i2c-2  i2c          OMAP I2C adapter          I2C adapter
i2c-0  i2c          OMAP I2C adapter          I2C adapter
debian@beaglebone:~$ i2cdetect -y -r 2
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60: 60  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Here is the i2c transaction for `i2cdetect -y -r 2 0x60 0x60`.



(<http://localhost:4000/assets/images/2021/03/si5351-i2c.jpg>)

## 1.2. Linux userspace driver

There are several ways that this interface can be made. We are going to create a userspace driver, which is the most straightforward. The following program, derived from the Linux Kernel [userspace driver documentation](https://www.kernel.org/doc/html/latest/i2c/dev-interface.html) (<https://www.kernel.org/doc/html/latest/i2c/dev-interface.html>), reads and prints register 0 of device /dev/i2c-2, address 0x60, i.e. register 0 of the Si5351. Note that the reason we need to specify address 0x60 is that there could be multiple devices connected to a single i2c chip.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <sys/ioctl.h>
5  #include <linux/i2c-dev.h>
6  #include <i2c/smbus.h>
7
8  #define I2C_FNAME      "/dev/i2c-2"
9  #define SI5351_ADDR    0x60
10
11 int    i2c_file;
12
13 void i2c_init()
14 {
15     i2c_file = open(I2C_FNAME, O_RDWR);
16     if (i2c_file < 0)
17         exit(1);

```

```
1  }
4
1  int i2c_read(unsigned char reg)
5  {
1      if (ioctl(i2c_file, I2C_SLAVE, SI5351_ADDR) < 0)
6          exit(1);
1
7      int res;
1
8      /* Using SMBus commands */
1      res = i2c_smbus_read_byte_data(i2c_file, reg);
9      if (res < 0)
2          exit(1);
0      else
2          printf("r dev(0x%x) reg(0x%x)=0x%x\n", SI5351_ADDR, reg, res
1      );
2      return res;
2  }
2
3  int
2  main()
4  {
2      i2c_init();
5      i2c_read(0);
2  }
```

6  
2  
7  
2  
8  
2  
9  
3  
0  
3  
1  
3  
2  
3  
3  
3  
4  
3  
5  
3  
6  
3  
7  
3  
8  
3  
9  
4  
0  
4  
1

It can be compiled and executed as follows:

```
debian@beaglebone:~$ gcc -o si5351 si5351.c -li2c
debian@beaglebone:~$ ./si5351
r dev(0x60) reg(0x0)=0x11
```

## 1.3. Programming the Clock Generator

The [data sheet](https://www.silabs.com/documents/public/data-sheets/Si5351-B.pdf) [\\_ \(https://www.silabs.com/documents/public/data-sheets/Si5351-B.pdf\)](https://www.silabs.com/documents/public/data-sheets/Si5351-B.pdf) for the Si5351 refers to the [Clock builder](https://www.silabs.com/developers/clockbuilder-pro-software) [\\_ \(https://www.silabs.com/developers/clockbuilder-pro-software\)](https://www.silabs.com/developers/clockbuilder-pro-software) software. This is only available for Windows but in embedded systems it is a fact of life that if you use operating system X, there will be a piece of software that is only supported on operating system Y that you need. Here is a header file that I generated for CLK0 and CLK1 outputs of 28.1544 MHz.

```
1  /*
2   * Si5351A Rev B Configuration Register Export Header File
3   *
4   * This file represents a series of Silicon Labs Si5351A Rev B
5   * register writes that can be performed to load a single configuration
6   * on a device. It was created by a Silicon Labs ClockBuilder Pro
7   * export tool.
8   *
9   * Part:                               Si5351A Rev B
10  * Design ID:
11  * Includes Pre/Post Download Control Register Writes: Yes
12  * Created By:                         ClockBuilder Pro v3.1
13  [2021-01-18]
14  * Timestamp:                         2021-03-12 15:00:15 G
15  MT-08:00
16  *
17  * A complete design report corresponding to this export is included at the
18  end
19  * of this header file.
20  *
21  */
22
23 #ifndef SI5351A_REVB_REG_CONFIG_HEADER
24 #define SI5351A_REVB_REG_CONFIG_HEADER
25
26 #define SI5351A_REVB_REG_CONFIG_NUM_REGS          52
27
28 typedef struct
29 {
30     unsigned int address; /* 16-bit register address */
31     unsigned char value; /* 8-bit register data */
32 } si5351a_revb_register_t;
33
34 si5351a_revb_register_t const si5351a_revb_registers[SI5351A_REVB_REG_CONFIG
35 _NUM_REGS] =
36 {
37     { 0x0002, 0x53 },
```

```
39     { 0x0003, 0x00 },
40     { 0x0004, 0x20 },
41     { 0x0007, 0x00 },
42     { 0x000F, 0x00 },
43     { 0x0010, 0x0F },
44     { 0x0011, 0x0F },
45     { 0x0012, 0x8C },
46     { 0x0013, 0x8C },
47     { 0x0014, 0x8C },
48     { 0x0015, 0x8C },
49     { 0x0016, 0x8C },
50     { 0x0017, 0x8C },
51     { 0x001A, 0xAF },
52     { 0x001B, 0xC8 },
53     { 0x001C, 0x00 },
54     { 0x001D, 0x0E },
55     { 0x001E, 0x8D },
56     { 0x001F, 0x00 },
57     { 0x0020, 0x85 },
58     { 0x0021, 0x58 },
59     { 0x002A, 0x00 },
60     { 0x002B, 0x04 },
61     { 0x002C, 0x00 },
62     { 0x002D, 0x0D },
63     { 0x002E, 0xE0 },
64     { 0x002F, 0x00 },
65     { 0x0030, 0x00 },
66     { 0x0031, 0x00 },
67     { 0x0032, 0x00 },
68     { 0x0033, 0x04 },
69     { 0x0034, 0x00 },
70     { 0x0035, 0x0D },
71     { 0x0036, 0xE0 },
72     { 0x0037, 0x00 },
73     { 0x0038, 0x00 },
74     { 0x0039, 0x00 },
75     { 0x005A, 0x00 },
76     { 0x005B, 0x00 },
77     { 0x0095, 0x00 },
78     { 0x0096, 0x00 },
79     { 0x0097, 0x00 },
80     { 0x0098, 0x00 },
81     { 0x0099, 0x00 },
82     { 0x009A, 0x00 },
83     { 0x009B, 0x00 },
84     { 0x00A2, 0x00 },
85     { 0x00A3, 0x00 },
86     { 0x00A4, 0x00 },
87     { 0x00A5, 0x00 },
88     { 0x00A6, 0x00 },
89     { 0x00B7, 0x92 },
90
91 };
92
93 /*
94  * Design Report
95  *
96  * Overview
```

```

97 * =====
98 * Part:                Si5351A
99 * Created By:          ClockBuilder Pro v3.1 [2021-01-18]
10 * Timestamp:           2021-03-12 15:00:15 GMT-08:00
0  *
10 * Design Notes
1  * =====
10 * Si5351A 10-pin MSOP
2  *
10 * Design Rule Check
3  * =====
10 * Errors:
4  * - No errors
10 *
5  * Warnings:
10 * - No warnings
6  *
10 * Design
7  * =====
10 * Inputs:
8  *     IN0: 27 MHz
10 *
9  * Outputs:
11 *     OUT0: 28.1544 MHz
0  *         Enabled LVCMOS 8 mA
11 *         Offset 0.000 s
1  *     OUT1: 28.1544 MHz
11 *         Enabled LVCMOS 8 mA
2  *         Offset 0.000 s
11 *     OUT2: Unused
3  *
11 * Frequency Plan
4  * =====
11 * PLL_A:
5  *     Enabled Features = None
11 *     Fvco              = 893.9022 MHz
6  *     M                = 33.107488888888888... [ 33 + 4837/45000 ]
11 *     Input0:
7  *         Source        = Crystal
11 *         Source Frequency = 27 MHz
8  *         Fpfd          = 27 MHz
11 *         Load Capacitance = Load_08pF
9  *     Output0:
12 *         Features      = None
0  *         Disabled State = StopLow
12 *         R              = 1 (2^0)
1  *         Fout          = 28.1544 MHz
12 *         N              = 31.75
2  *     Output1:
12 *         Features      = None
3  *         Disabled State = StopLow
12 *         R              = 1 (2^0)
4  *         Fout          = 28.1544 MHz
12 *         N              = 31.75
5  *
12 * Settings
6  * =====
12 *

```



7	* Location	Setting Name	Decimal Value	Hex Value
12	* -----	-----	-----	-----
8	* 0x0002[3]	X0_LOS_MASK	0	0x0
12	* 0x0002[4]	CLK_LOS_MASK	1	0x1
9	* 0x0002[5]	LOL_A_MASK	0	0x0
13	* 0x0002[6]	LOL_B_MASK	1	0x1
0	* 0x0002[7]	SYS_INIT_MASK	0	0x0
13	* 0x0003[7:0]	CLK_OEB	0	0x00
1	* 0x0004[4]	DIS_RESET_LOLA	0	0x0
13	* 0x0004[5]	DIS_RESET_LOLB	1	0x1
2	* 0x0007[7:4]	I2C_ADDR_CTRL	0	0x0
13	* 0x000F[2]	PLLA_SRC	0	0x0
3	* 0x000F[3]	PLLB_SRC	0	0x0
13	* 0x000F[4]	PLLA_INSELB	0	0x0
4	* 0x000F[5]	PLLB_INSELB	0	0x0
13	* 0x000F[7:6]	CLKIN_DIV	0	0x0
5	* 0x0010[1:0]	CLK0_IDRV	3	0x3
13	* 0x0010[3:2]	CLK0_SRC	3	0x3
6	* 0x0010[4]	CLK0_INV	0	0x0
13	* 0x0010[5]	MS0_SRC	0	0x0
7	* 0x0010[6]	MS0_INT	0	0x0
13	* 0x0010[7]	CLK0_PDN	0	0x0
8	* 0x0011[1:0]	CLK1_IDRV	3	0x3
13	* 0x0011[3:2]	CLK1_SRC	3	0x3
9	* 0x0011[4]	CLK1_INV	0	0x0
14	* 0x0011[5]	MS1_SRC	0	0x0
0	* 0x0011[6]	MS1_INT	0	0x0
14	* 0x0011[7]	CLK1_PDN	0	0x0
1	* 0x0012[1:0]	CLK2_IDRV	0	0x0
14	* 0x0012[3:2]	CLK2_SRC	3	0x3
2	* 0x0012[4]	CLK2_INV	0	0x0
14	* 0x0012[5]	MS2_SRC	0	0x0
3	* 0x0012[6]	MS2_INT	0	0x0
14	* 0x0012[7]	CLK2_PDN	1	0x1
4	* 0x0013[1:0]	CLK3_IDRV	0	0x0
14	* 0x0013[3:2]	CLK3_SRC	3	0x3
5	* 0x0013[4]	CLK3_INV	0	0x0
14	* 0x0013[5]	MS3_SRC	0	0x0
6	* 0x0013[6]	MS3_INT	0	0x0
14	* 0x0013[7]	CLK3_PDN	1	0x1
7	* 0x0014[1:0]	CLK4_IDRV	0	0x0
14	* 0x0014[3:2]	CLK4_SRC	3	0x3
8	* 0x0014[4]	CLK4_INV	0	0x0
14	* 0x0014[5]	MS4_SRC	0	0x0
9	* 0x0014[6]	MS4_INT	0	0x0
15	* 0x0014[7]	CLK4_PDN	1	0x1
0	* 0x0015[1:0]	CLK5_IDRV	0	0x0
15	* 0x0015[3:2]	CLK5_SRC	3	0x3
1	* 0x0015[4]	CLK5_INV	0	0x0
15	* 0x0015[5]	MS5_SRC	0	0x0
2	* 0x0015[6]	MS5_INT	0	0x0
15	* 0x0015[7]	CLK5_PDN	1	0x1
3	* 0x0016[1:0]	CLK6_IDRV	0	0x0
15	* 0x0016[3:2]	CLK6_SRC	3	0x3
4	* 0x0016[4]	CLK6_INV	0	0x0
15	* 0x0016[5]	MS6_SRC	0	0x0
5	* 0x0016[6]	FBA_INT	0	0x0
15	* 0x0016[7]	CLK6_PDN	1	0x1

```

6      * 0x0017[1:0] CLK7_IDRV      0      0x0
15     * 0x0017[3:2] CLK7_SRC       3      0x3
7      * 0x0017[4]   CLK7_INV       0      0x0
15     * 0x0017[5]   MS7_SRC        0      0x0
8      * 0x0017[6]   FBB_INT        0      0x0
15     * 0x0017[7]   CLK7_PDN       1      0x1
9      * 0x001C[17:0] MSNA_P1       3725   0x00E8D
16     * 0x001F[19:0] MSNA_P2       34136  0x08558
0      * 0x001F[23:4] MSNA_P3       45000  0x0AFC8
16     * 0x002C[17:0] MS0_P1        3552   0x00DE0
1      * 0x002F[19:0] MS0_P2        0      0x00000
16     * 0x002F[23:4] MS0_P4        4      0x00004
2      * 0x0034[17:0] MS1_P1       3552   0x00DE0
16     * 0x0037[19:0] MS1_P2        0      0x00000
3      * 0x0037[23:4] MS1_P4        4      0x00004
16     * 0x005A[7:0]  MS6_P2        0      0x00
4      * 0x005B[7:0]  MS7_P2        0      0x00
16     * 0x0095[14:0] SSDN_P2       0      0x0000
5      * 0x0095[7]    SSC_EN        0      0x0
16     * 0x0097[14:0] SSDN_P3       0      0x0000
6      * 0x0097[7]    SSC_MODE      0      0x0
16     * 0x0099[11:0] SSDN_P1       0      0x000
7      * 0x009A[15:4] SSUDP         0      0x000
16     * 0x00A2[21:0] VCXO_PARAM    0      0x000000
8      * 0x00A5[7:0]  CLK0_PHOFF    0      0x00
16     * 0x00A6[7:0]  CLK1_PHOFF    0      0x00
9      * 0x00B7[7:6]  XTAL_CL       2      0x2
17     *
0      *
17     */
1
17     #endif
2
17
3
17
4
17
5
17
6
17
7
17
8
17
9
18
0
18
1
18
2
18
3
18
4
18

```



5
18
6
18
7
18
8
18
9
19
0
19
1
19
2
19
3
19
4
19
5
19
6
19
7
19
8
19
9
20
0
20
1
20
2
20
3
20
4
20
5
20
6
20
7
20
8
20
9
21
0
21
1
21
2
21
3
21

4  
21  
5  
21  
6  
21  
7  
21  
8  
21  
9  
22  
0  
22  
1  
22  
2  
22  
3  
22  
4  
22  
5  
22  
6  
22  
7  
22  
8  
22  
9  
23  
0  
23  
1  
23  
2  
23  
3  
23  
4  
23  
5  
23  
6  
23  
7  
23  
8  
23  
9  
24  
0

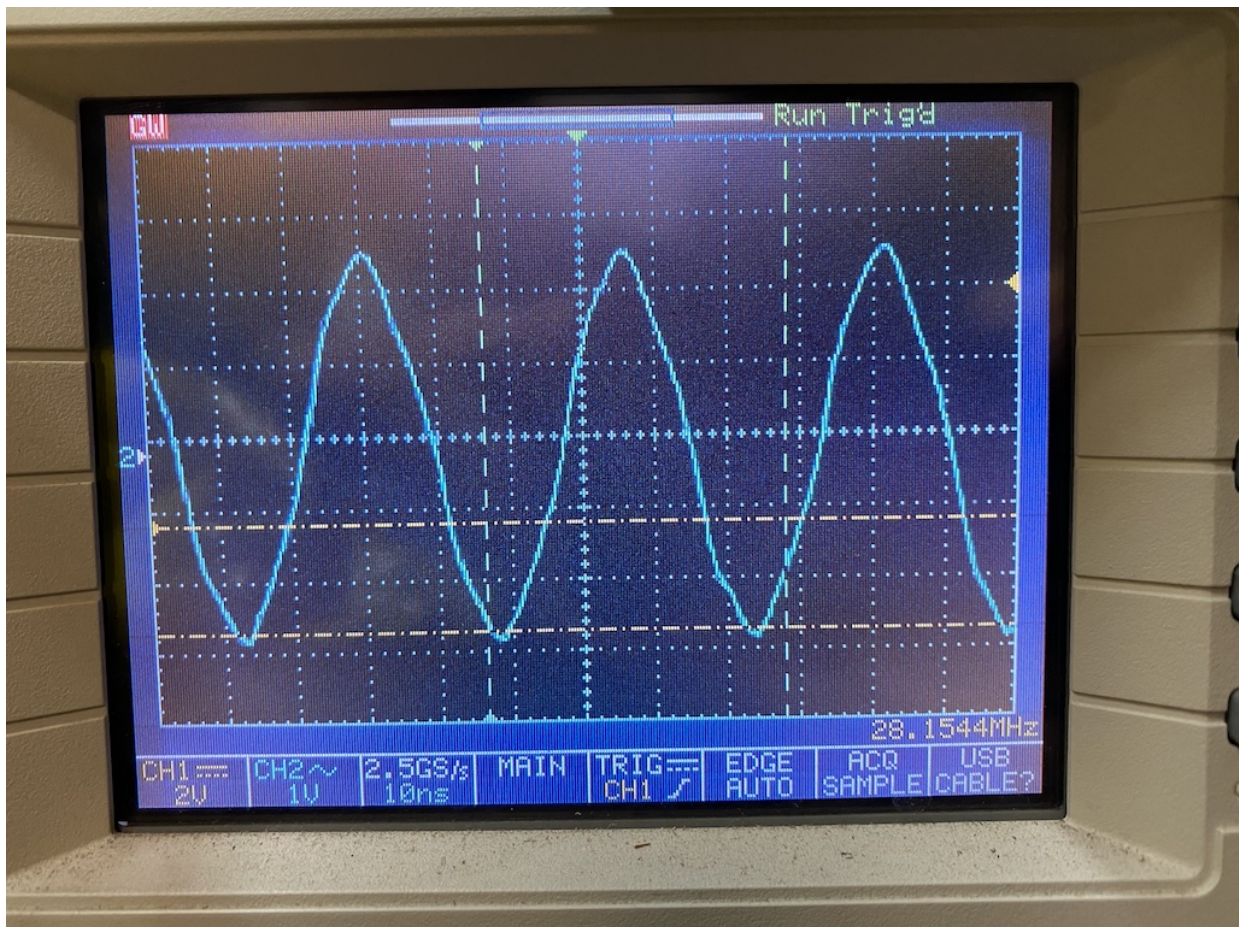
Our approach will be to use the header file above, [Application Note AN619 Manually Generating an Si5351 Register Map for 10-MSOP and 20-QFN Devices](#)

(<https://www.silabs.com/documents/public/application-notes/AN619.pdf>)\_ and a [Silicon Labs Knowledge Base entry](https://www.silabs.com/community/timing/knowledge-base.entry.html/2011/02/18/programming_the_si53-lrM8) ([https://www.silabs.com/community/timing/knowledge-base.entry.html/2011/02/18/programming\\_the\\_si53-lrM8](https://www.silabs.com/community/timing/knowledge-base.entry.html/2011/02/18/programming_the_si53-lrM8))\_ which says

ClockBuilder Desktop allows a user to generate RAM configuration files to program the Si5351 with custom frequency plans via I2C. Once the register map has been generated, use the procedure below to program the device.

1. Disable all outputs.  
    `reg3 = 0xFF`
2. Write `reg187 = 0xC0`
3. Power down all output drivers  
    `reg 16 = 0x80*`  
    `reg 17 = 0x80*`  
    `reg 18 = 0x80*`  
    `reg 19 = 0x80*`  
    `reg 20 = 0x80*`  
    `reg 21 = 0x80*`  
    `reg 22 = 0x80*`  
    `reg 23 = 0x80*`  
    \* If using the Si5351C with no crystal present on XA/XB, set `reg16-23 = 0x84`.
4. Set interrupt mask register (see Register 2 description in datasheet)
5. If needed, set crystal load capacitance, XTAL\_CL in `reg183[7:6]`. See datasheet for register description.
6. Write registers 15-92 and 149-170 using the contents of register map generated by ClockBuilder Desktop.
7. Apply PLL A and PLL B soft reset.  
    `reg177 = 0xAC`
8. Enable outputs with OEB control in register 3.

If these steps are followed, the output as specified in the include file should appear. It doesn't look much like a square wave on my 150 MHz bandwidth oscilloscope. Why?



(<http://localhost:4000/assets/images/2021/03/si5351-osc.jpg>)

Finally, we wish to have the inphase (I) clock (CLK0) lagging the quadrature clock (CLK1) by 90 degrees (or 1/4 cycle). We can do this by setting the CLK1\_PHOFF register to the appropriate value.

## 2. Laboratory Experiment

Update the lab files as below.

```

debian@beaglebone:~$ cd elec3607-labquestions/
debian@beaglebone:~/elec3607-labquestions$ git pull
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 14 (delta 2), reused 14 (delta 2), pack-reused 0
Unpacking objects: 100% (14/14), done.
From https://github.com/phwl/elec3607-labquestions
  6bf3a34..7c881d4  main      -> origin/main
Updating 6bf3a34..7c881d4
Fast-forward
 .DS_Store                               | Bin 6148 -> 6148 bytes
 labs/.DS_Store                         | Bin 0 -> 8196 bytes
 ... - GPIO_ ELEC3607 ELEC9607 Embedded Systems.pdf | Bin 0 -> 297484 bytes
 ...ruction_ ELEC3607 ELEC9607 Embedded Systems.pdf | Bin 0 -> 156733 bytes

```

```

...nerator_ ELEC3607 ELEC9607 Embedded Systems.pdf | Bin 0 -> 2310302 bytes
labs/lab3-i2c/Makefile | 7 +
labs/lab3-i2c/Si5351A-RevB-Registers.h | 240 ++++++
labs/lab3-i2c/si5351.c | 41 ++++
8 files changed, 288 insertions(+)
create mode 100644 labs/.DS_Store
create mode 100644 labs/lab1-gpio/Lab 1 - GPIO_ ELEC3607 ELEC9607 Embedded Systems.pdf
create mode 100644 labs/lab2-sdrconstruction/Lab 2 - SDR Construction_ ELEC3607 ELEC9607 Emb
edded Systems.pdf
create mode 100644 labs/lab3-i2c/Lab 3 - Si5351 Clock Generator_ ELEC3607 ELEC9607 Embedded
Systems.pdf
create mode 100644 labs/lab3-i2c/Makefile
create mode 100755 labs/lab3-i2c/Si5351A-RevB-Registers.h
create mode 100644 labs/lab3-i2c/si5351.c
debian@beaglebone:~/elec3607-labquestions$ cd labs/lab3-i2c/
debian@beaglebone:~/elec3607-labquestions/labs/lab3-i2c$

```

## Part 1 - I2C Interface (30%)

Connect up your BBG to the Si5351. Verify that you can obtain the following output.

```

debian@beaglebone:~$ i2cdetect -y -r 2
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60: 60  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --

```

## Part 2 - I2C Transaction (30%)

Use the userspace i2c driver (`si5351.c`) shown above to read register 0. Execute the program and capture the activity of the SCL and SDA pins on an oscilloscope. Make a printout of the oscilloscope display and annotate all parts of the i2c transaction (start, data, r/w, ack, etc). What is the period of the entire transaction?

## Part 3 - Si5351 Configuration (40%)

Modify the userspace driver for the Si5351 to generate a 7 MHz square wave output on CLK0 and CLK1, with CLK1 being delayed by 90 degrees from CLK0.

Copyright © The University of Sydney. Unless otherwise indicated, 3rd party material has been reproduced and communicated to you by or on behalf of the University of Sydney in accordance with section 113P of the Copyright Act 1968 (Act). The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice.

Live streamed classes in this unit may be recorded to enable students to review the content. If you have concerns about this, please visit our [student guide](https://canvas.sydney.edu.au/courses/4901/pages/zoom) (<https://canvas.sydney.edu.au/courses/4901/pages/zoom>) and contact the unit coordinator.