iCE40 Ultra™ Barcode Emulation

User's Guide

June 2014
UG73_1.0

Introduction

This guide describes how to use the iCE40 Ultra™ Mobile Development Platform for demonstrating Barcode Emulation design for user application. This guide familiarizes you with the process of setting up your Barcode Emulation Design Environment. It guides you through the hardware and software required to successfully run your Barcode Emulation demonstration.

The document discusses complete demonstration steps and the associated designs.

After you complete the procedures in this guide, you will be able to:

- Set up the iCE40 Ultra Mobile Development Platform properly and become familiar with its main features.
- Work and become familiar with the software required for Barcode Emulation demonstrations.
- Utilize the additional hardware required to run the demonstrations.
- Understand the design details of the Barcode Emulation demo implemented on iCE40 Ultra.
- Run the demo along with the Intrinsyc® DragonBoard™.
- Use other Lattice documentation in conjunction with this guide.

This document assumes that you have already installed the Lattice iCEcube2 and the Lattice Diamond® Programmer software and are familiar with basic tasks. If you need more information on these software, please refer to the iCEcube2 and Diamond Programmer help.

For details on specific board features and other information, refer to:

- EB90, iCE40 Ultra Mobile Development Platform User's Guide
- DS1048, iCE40 Ultra Family Data Sheet
- MoBeam ASIC documentation

This document is divided into two sections. The first section describes the Barcode Emulation demonstration in detail and the second section describes the Barcode Emulation design. The Barcode Emulation demonstration is performed using I2C or SPI interface with the Processor.

Barcode Emulation Demonstration

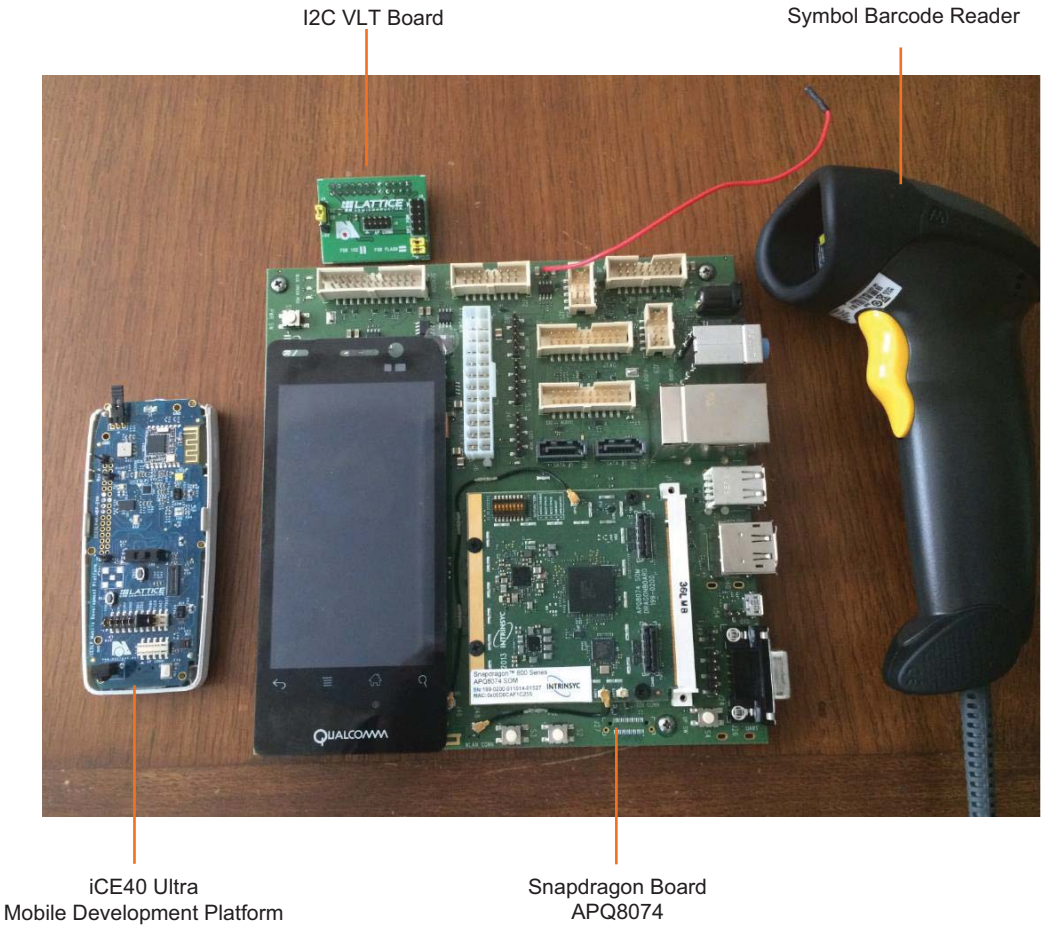
This section describes the Barcode Emulation demonstration in detail.

Barcode Emulation Demonstration Setup Using I2C Interface

The Barcode Emulation demonstration setup using I2C interface consists of the following components, as shown in Figure 1.

- APQ8074 Intrinsyc DragonBoard Gen 2 with +5V adaptor and USB debugger cable
- iCE40 Ultra Mobile Development Platform with programming USB cable
- I2C VLT board
- Symbol Barcode Reader
- Three flexible connecting cables

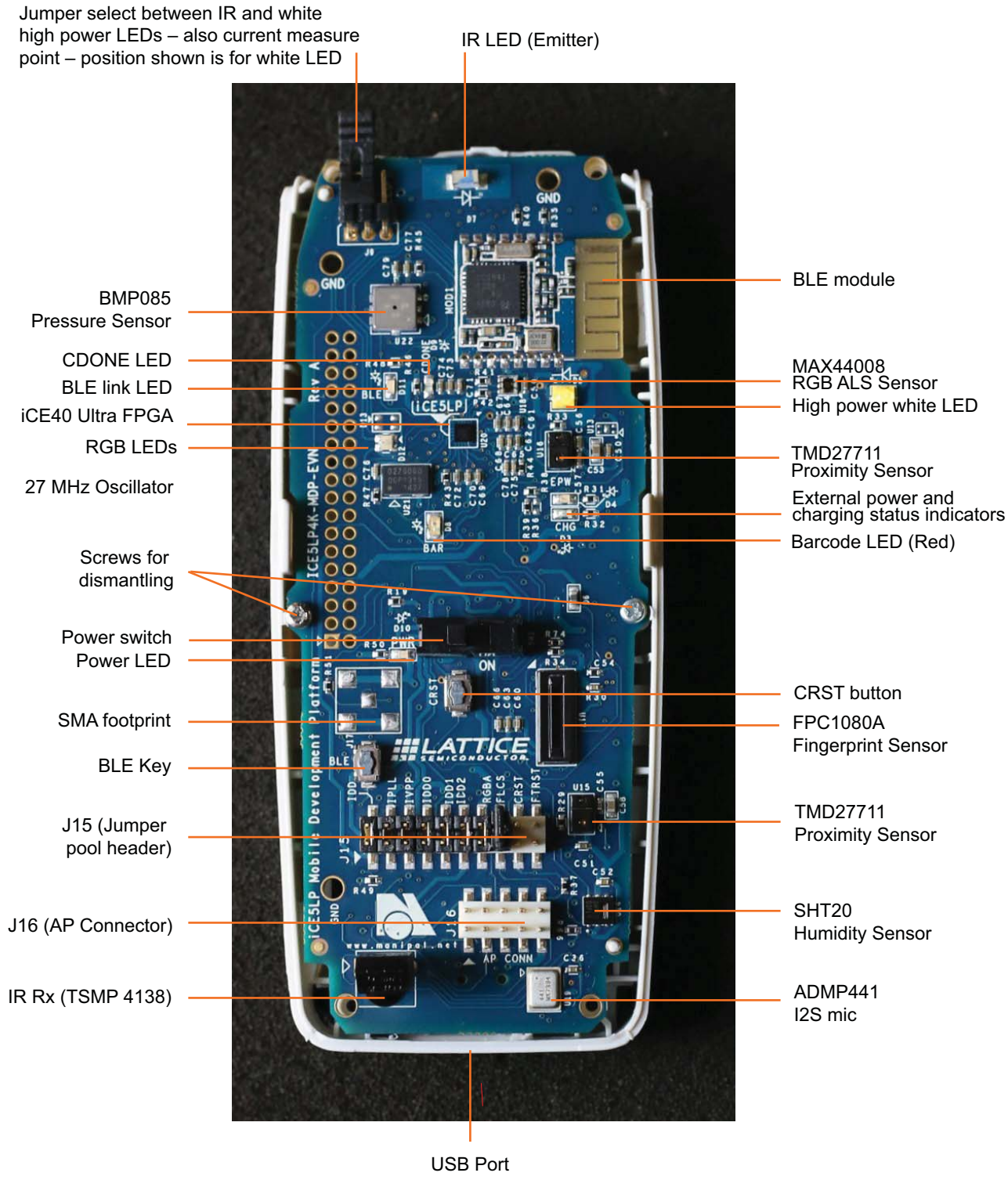
Figure 1. Barcode Emulation Controller I2C Demo Setup



iCE40 Ultra Mobile Development Platform Default Jumper Settings

The details of the iCE40 Ultra Mobile Development Platform default jumper settings are shown in Figure 2

Figure 2. Default Jumper Settings



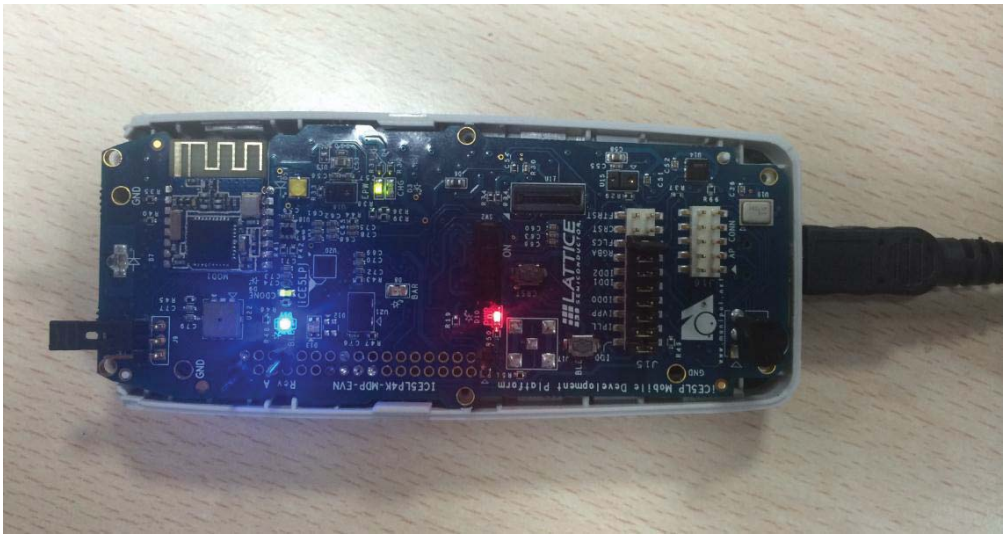
Note: In the J15 jumper set the FLCS pins.

Programming the iCE40 Ultra Mobile Development Platform

To program the iCE40 Mobile Development Platform:

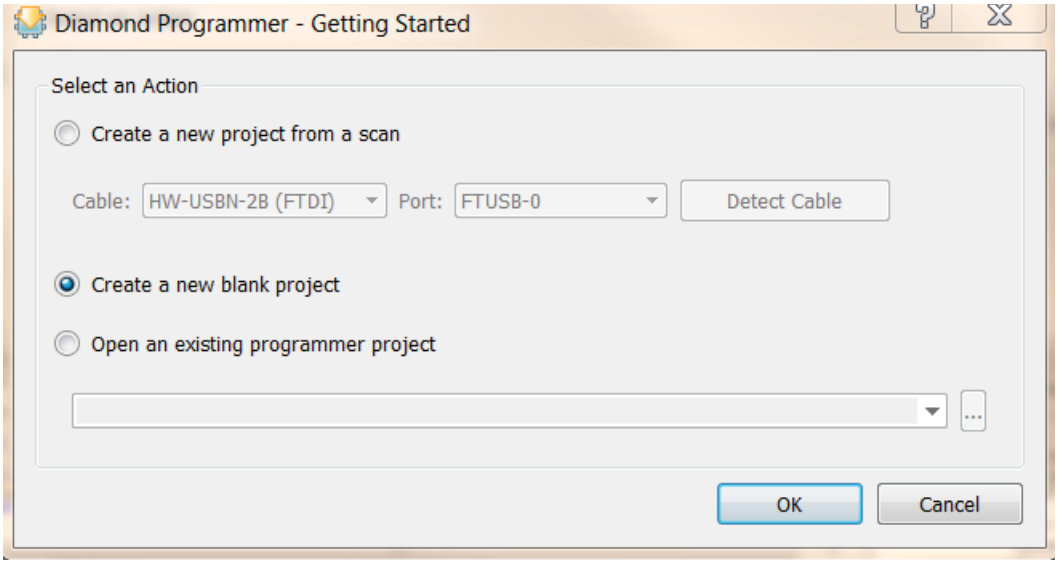
1. Connect the iCE40 Mobile Development Platform to the USB port of the PC.

Figure 3. Connecting Board to PC



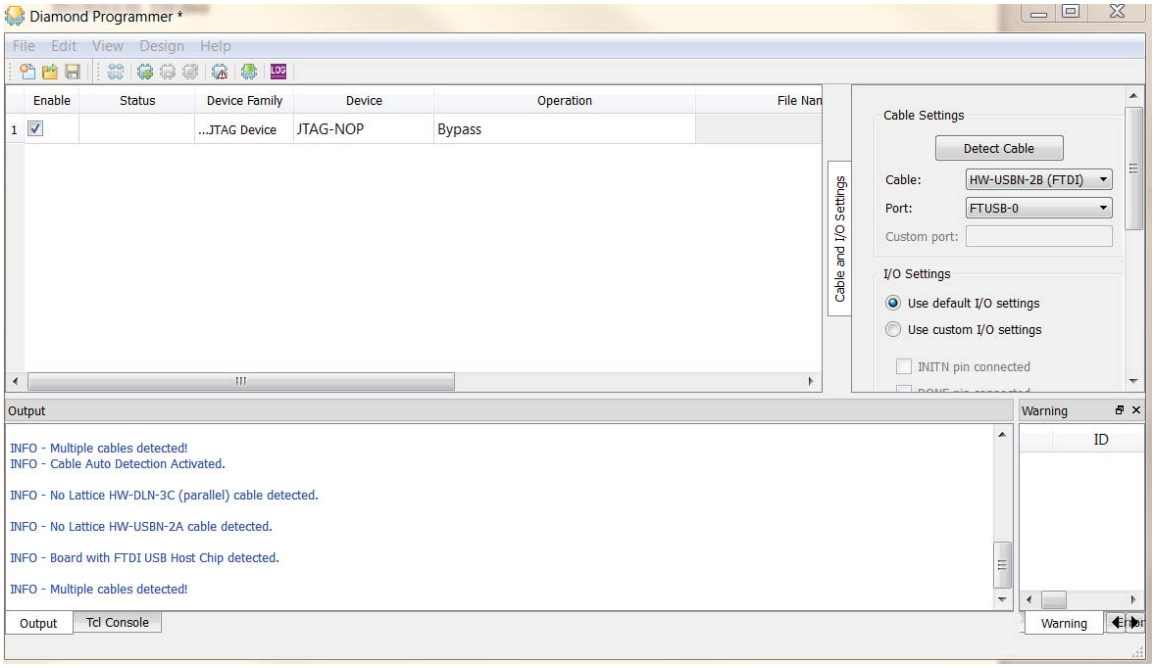
2. Open the Diamond® Programmer version 3.2 and above.
3. In the Getting Started dialog box, select **Create a new blank project** and click **OK**.

Figure 4. Creating New Project



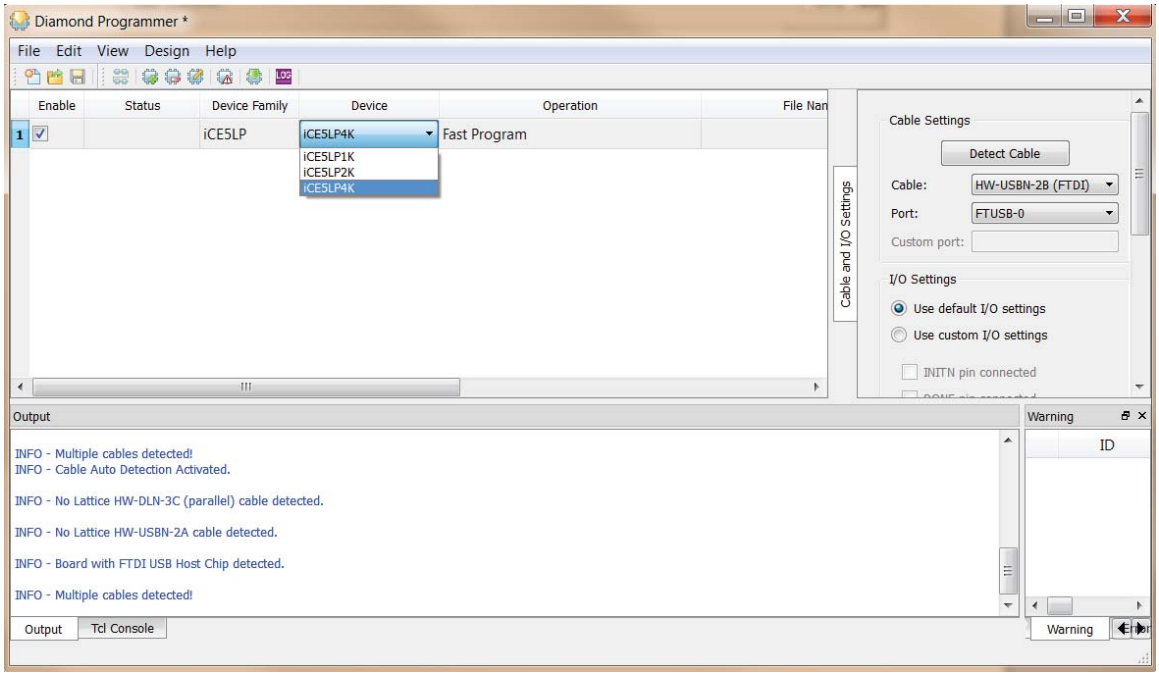
This opens the Diamond Programmer main interface.

Figure 5. Diamond Programmer Interface



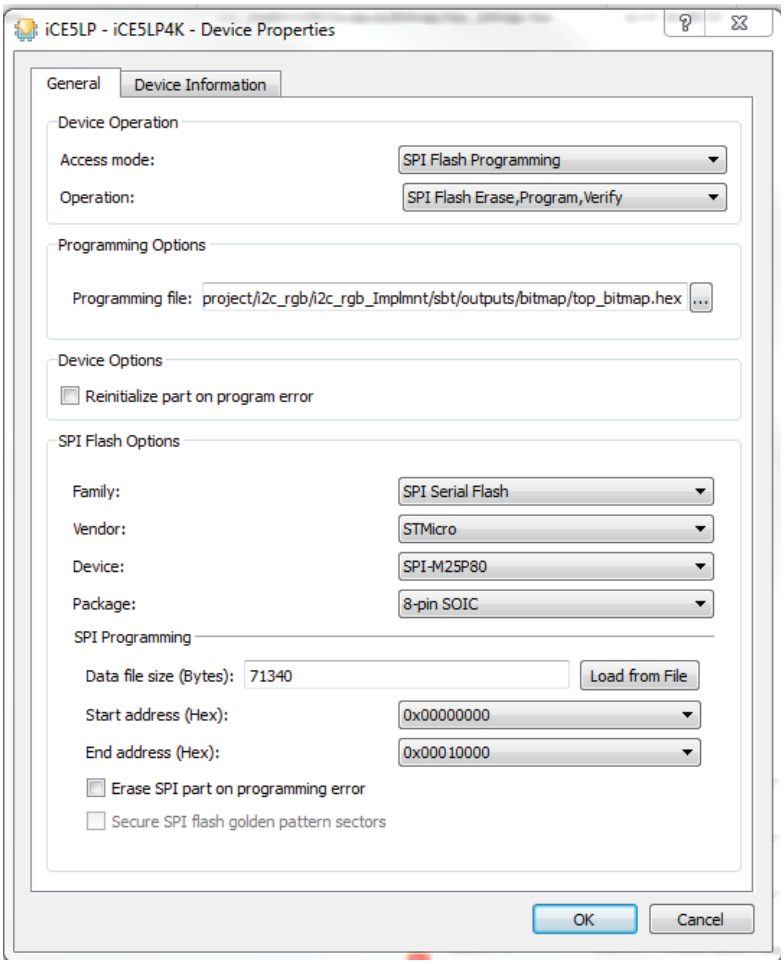
4. Select device under Device Family and then select **iCE5LP4K** under Device.

Figure 6. Selecting the Device



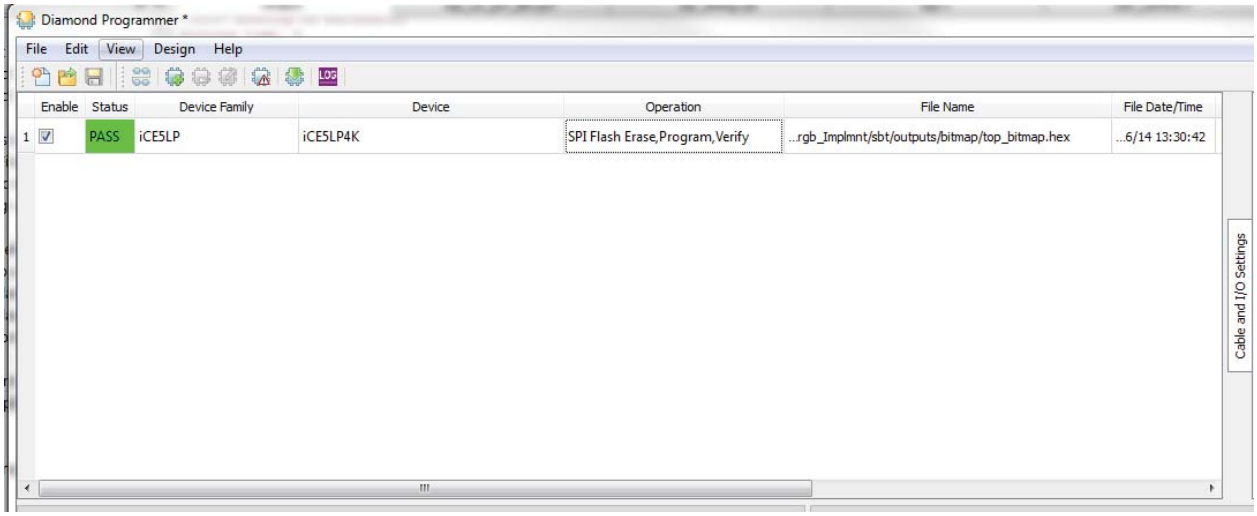
5. Double-click on Fast Program and the .hex file provided in the demonstration /i2c folder.

Figure 7. Selecting Programming File



6. Select the program to use in programming the device.
7. Verify that the operation has successfully completed.

Figure 8. Verifying Operation



Connecting iCE40 Ultra Mobile Development Platform to SnapDragon Board APQ8074

To connect iCE40 Ultra Mobile Development Platform to SnapDragon Board APQ8074:

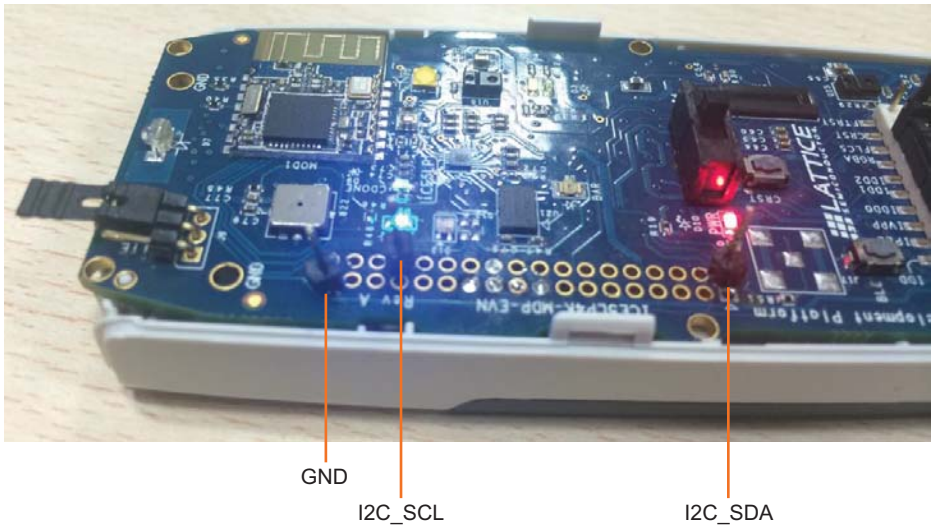
1. Connect the I2C lines between I2C VLT board and iCE40 Ultra Mobile Development Platform as shown in Table 1.

Table 1. I2C Line Connections

Sr. No.	I2C VLT Board [J6 Jumper] Pin Numbers	Signal	iCE40 Ultra Mobile Development Platform [J10 Jumper] Pin Numbers
1	4	I2C_SDA	2
2	3	I2C_SCL	30
3	9	Ground	18 / 36

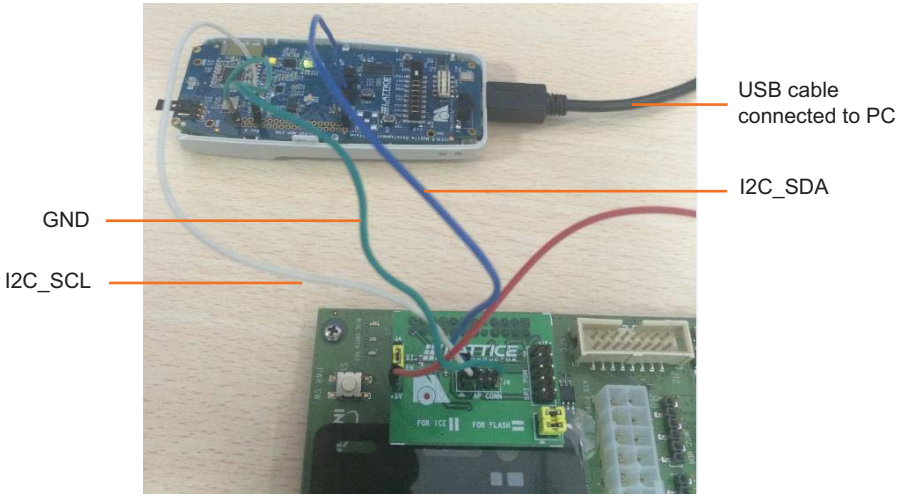
2. Solder three jumper pins as shown in the Figure 9.

Figure 9. Soldering Jumper Pins



3. Connect the I2C lines as shown in Figure 10.

Figure 10. Connecting I2C Lines



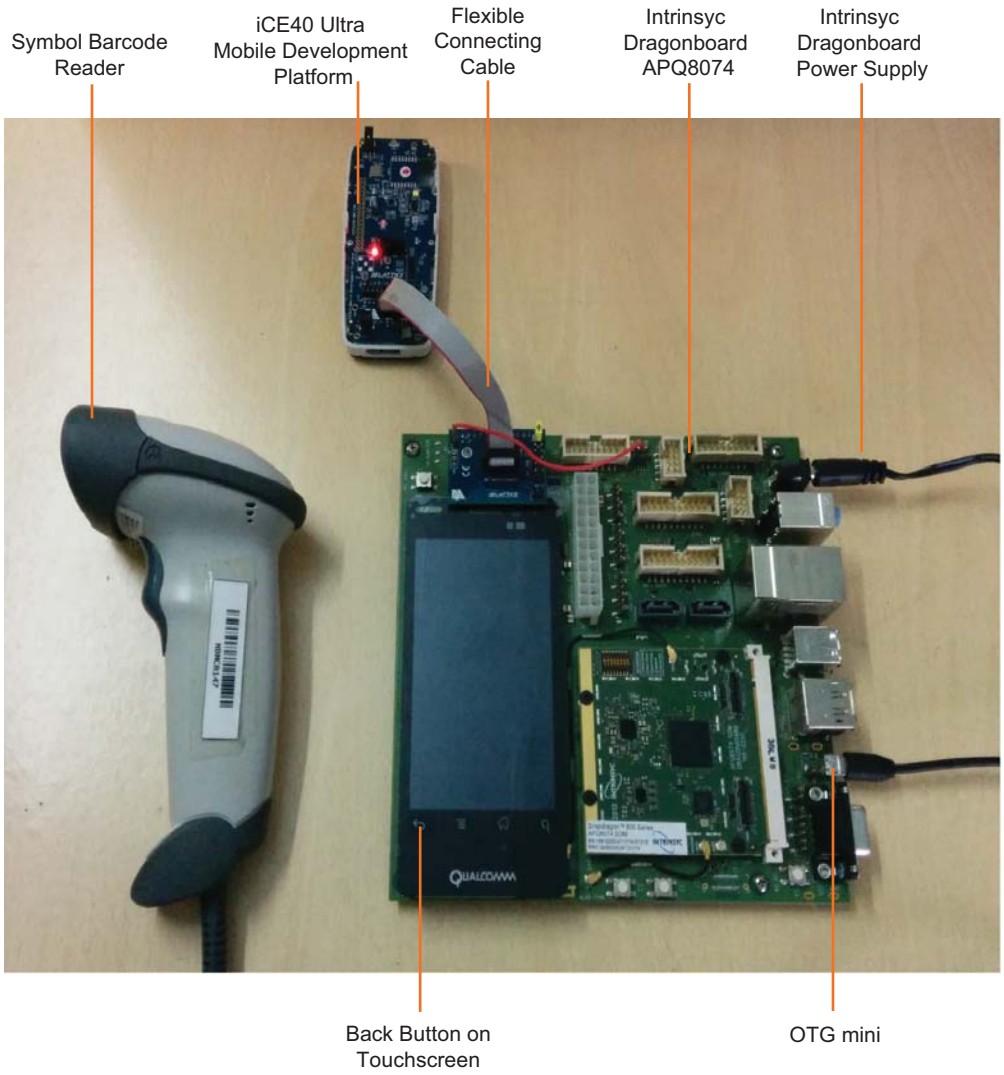
Barcode Emulation Demonstration Setup Using SPI Interface

The Barcode Emulation demonstration setup using SPI interface consists of the following components:

- APQ8074 Intrinsyc DragonBoard Gen 2 with +5V adaptor and USB debugger cable
- iCE40 Ultra Mobile Development Platform with programming USB cable
- Symbol Barcode Reader
- Flexible connecting cables

Connect the iCE40 Ultra Mobile Development Platform to the adapter board mounted on the Intrinsyc APQ8074 DragonBoard as shown in Figure 11. The sensor header is directly above the display.

Figure 11. Connecting the iCE40 Ultra Mobile Development Platform



iCE40 Ultra Mobile Development Platform Default Jumper Settings

Set Key A to select Bar LED. In the jumper pool J15, set all jumpers except FLCS and CRST.

Note: For processor configuration demo, set FTRST. Do not set FLCS.

Connecting the iCE40 Ultra Mobile Development Platform to the Intrinsyc DragonBoard

To connect the iCE40 Ultra Mobile Development Platform to the Intrinsyc DragonBoard:

1. Power-off the Intrinsyc DragonBoard.
2. Connect one end of the flexible connecting cable to the adapter board mounted on the Intrinsyc DragonBoard. The red wire of the cable connector should be connected to Pin 1 of the connector on the adapter board. Pin 1 is located near the white triangle.
3. Connect the other end of the cable to the iCE40 Ultra Mobile Development Platform. The red wire of the cable connector should be connected to Pin 1 of the J16 connector. Pin 1 is located near the white triangle.
4. Power-on the Intrinsyc DragonBoard.

The connections are shown in

Figure 12. Cable Connection to Adapter Board

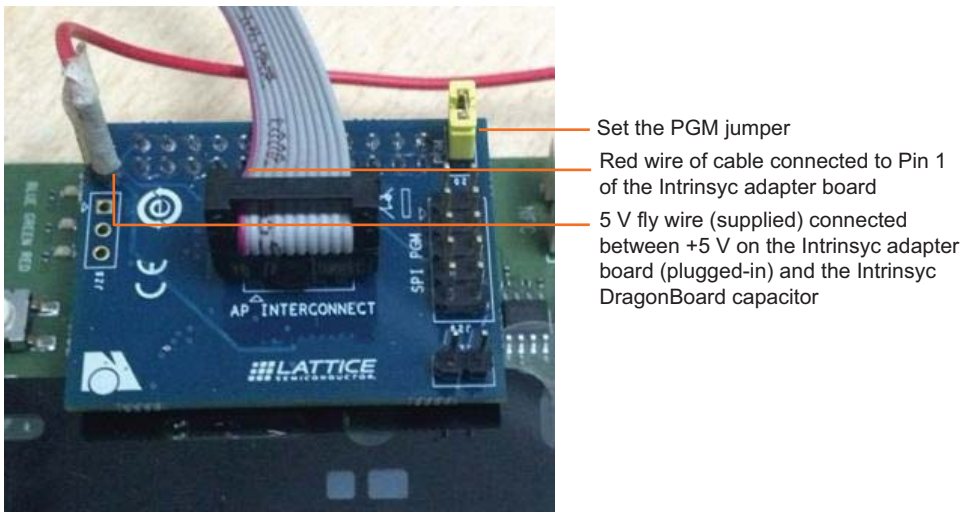
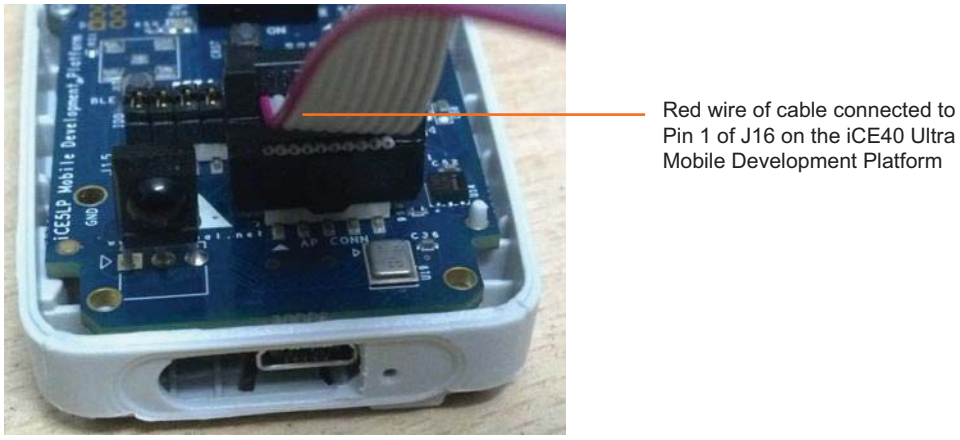


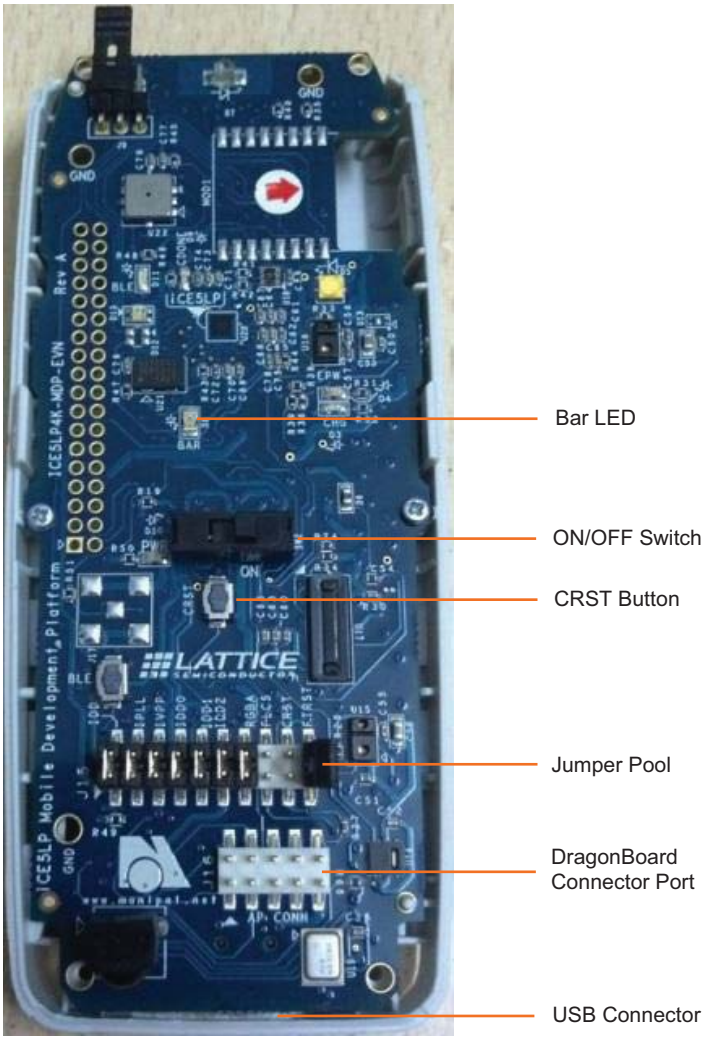
Figure 13. Cable Connection to iCE40 Ultra Mobile Development Platform



iCE40 Ultra Mobile Development Platform Details

The details of the iCE40 Ultra Mobile Development Platform are shown in Figure 14.

Figure 14. iCE40 Ultra Mobile Development Platform Details



Barcode Emulation Software Setup

This section provides the procedures in downloading Flash system and boot images to Intrinsyc DragonBoard.

Note: This procedure is not required if the DragonBoard is already flashed with the system and boot images.

To flash system image and boot image to Intrinsyc DragonBoard:

1. Make OTG (mini-USB) connection from Intrinsyc DragonBoard mini USB port of your Host system.
2. Download and install android-sdk. Android-sdk is available for Linux and Windows environments from http://developer.android.com/sdk/index.html.

   Note: Add the installation location /android_sdk/platform-tools/ in your system PATH variable.
3. Run the command below in the command prompt.

```
#sudo -s
```

Note: This command is only applicable for Linux machines. In Windows, administrative permission is required.

4. Reboot the Intrinsyc DragonBoard in FASTBOOT mode.

   Keep the S2 button pressed on the Intrinsyc DragonBoard during power on. If the board is already powered ON and is in adb mode, run the command below for FASTBOOT mode.

   ```
   #adb reboot bootloader
   ```

5. When the Intrinsyc DragonBoard is in FASTBOOT mode, a white screen is displayed with only the Intrinsyc name.

   Run the command below in the command prompt. The FASTBOOT device number and its name is listed.

   ```
   #fastboot devices
   ```

   When the procedure is completed, the board is ready to be flashed with the system and boot images.

6. Download APQ8074_JB_BootImage.zip and extract the contents below.

   For I2C: */APQ8074_JB_BootImage/I2C_img*
   For SPI: */APQ8074_JB_BootImage /SPI_img*

7. To flash the system and boot images, run the script file *flashall.sh*.

   For I2C: From the /APQ8074_JB_BootImage/I2C_img directory, run the script file *flashall.sh*.
   For SPI: From the / APQ8074_JB_BootImage /SPI_img directory, run the script file *flashall.sh*.

   Alternatively, you can run the commands below to flash the system image.

   For I2C:

   ```
   #cd APQ8074_JB_BootImage /I2C_img
   #fastboot flash system system.img
   ```

   For SPI:

   ```
   #cd APQ8074_JB_BootImage /SPI_img
   #fastboot flash system system.img
   ```

   If flashing is successful, OKAY and Finished are displayed on the terminal.

   Run the command below to flash the boot image.

   ```
   #fastboot flash boot boot.img
   ```

   If flashing is successful, OKAY and Finished are displayed on the terminal.

8. Reboot the board to verify the current board images using the command below.

   ```
   #fastboot reboot
   ```

9. After reboot is completed, go to System settings > About phone. Scroll down and tap on **Build number** seven times to enable Developer options.
10. Go to Developer options and select the **Stay awake** check box to keep the DragonBoard awake at all times.

Installing BarcodeTest.apk to Android

To install BarcodeTest.apk to Android:

1. Make OTG (mini-USB) connection from Intrinsyc DragonBoard mini USB port of your Host system.
2. Run the command below on the command prompt.

```
#sudo -s
```

3. To establish and verify adb connection, run the commands below.

```
#adb kill-server
#adb start-server
#adb devices
```

   If the connection is successful, the device ID is displayed on the terminal.
4. To install *BarcodeTest.apk* to Intrinsyc Dragonboard run the command below.

```
#cd Mobeam/Mobeam_Demo/demonstration/Demo_Apk
#adb root
#adb remount
#adb install BarcodeTest.apk
```

5. The application searches for the FPGA bitmap in the */etc/firmware/* directory. Manually push the FPGA bitmap to this location after installing the application. To do this, run the command below.

   For SPI:

```
#cd Mobeam/Mobeam_Demo/demonstration/bitstream/spi
#adb push mobeam_bitmap.bin/etc/firmware
```

   For I2C:

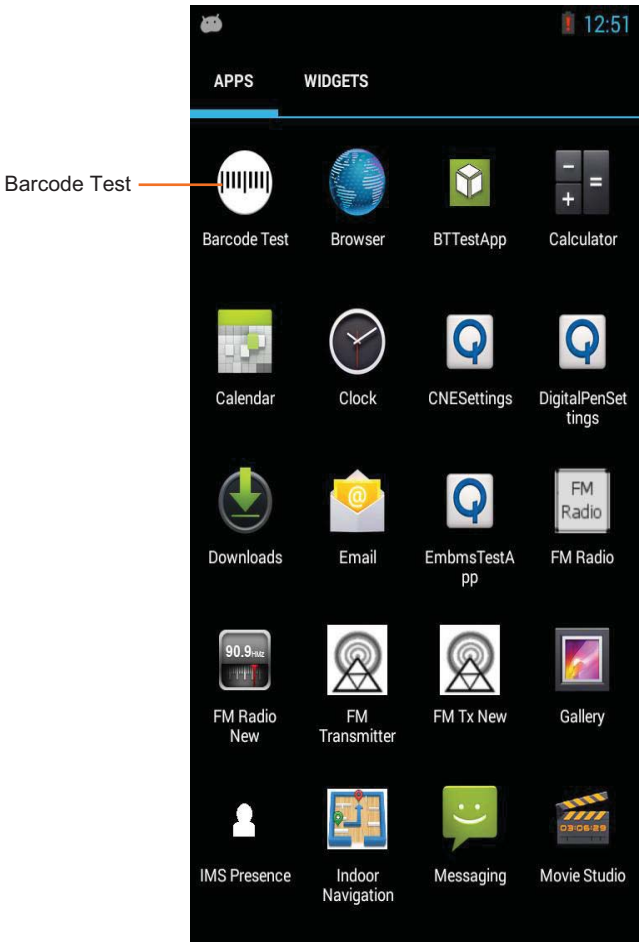
```
#cd Mobeam/Mobeam_Demo/demonstration/bitstream/i2c/mobeam_bitmap.bin
```

Demo Procedure

To run the demo:

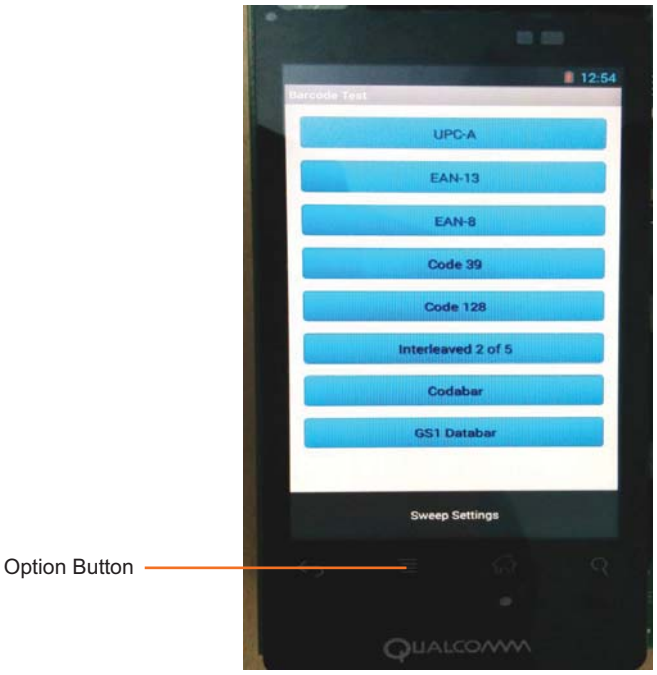
1. Connect the iCE40 Ultra Mobile Development Platform to DragonBoard using the flexible connecting cable.
2. Power ON Dragonboard and wait for the boot sequence to complete and the Home screen to appear.
3. Connect the Barcode reader to the USB port of your PC.
4. Unlock the screen. Go to the Android application menu and click the **Barcode Test** application as shown in Figure 15.

Figure 15. Barcode Test



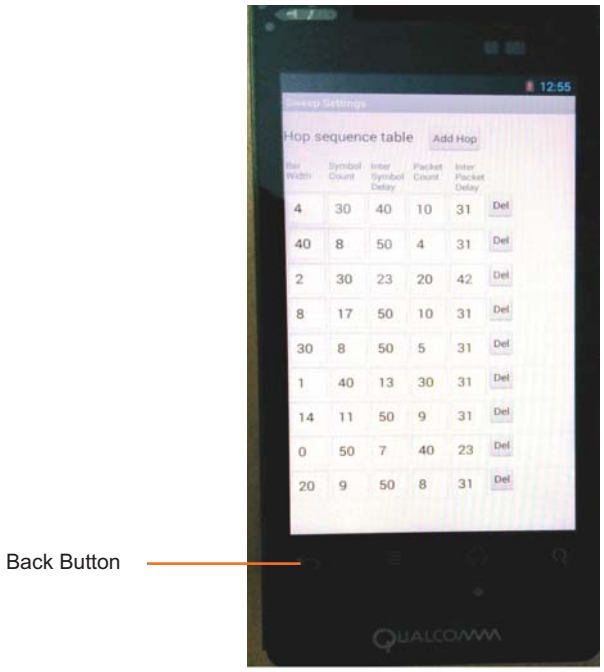
5. Wait for the Processor Configuration to be completed. This is indicated by the glowing of the CDONE LED on the iCE40 Ultra Mobile Development Platform.
6. Select the **Option** button to view the Hops settings as shown in Figure 16.

Figure 16. Option Button



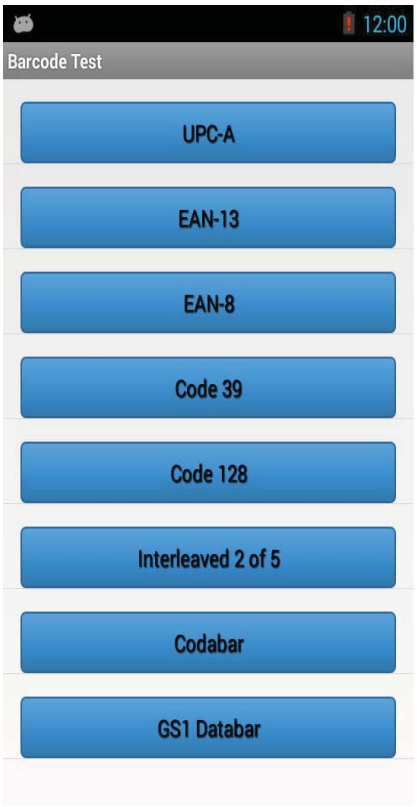
7. You have the option to change the Hops settings by selecting the Option button. You can add new HOPs or delete existing Hops settings. You can also change the BSR data values such as Bar Width, Symbol Counts, Inter Symbol Delays, Packet Counts and Inter Packet Delays. When you have completed the changes, select the **Back** button to go back to main menu.

Figure 17. Back Button



8. To begin, select one of the listed options, such as UPC-A, as shown in Figure 18.

Figure 18. Barcode Standards



9. The menu with the default code is displayed as shown in Figure 19.

Figure 19. Barcode Text Box

Select the **Start** button. This beams the UPC-A data and turns to on the BAR LED D8 on the iCE40 Ultra Board. The application is shown in Figure 20.

Figure 20. Beaming of Data Started

10. Open a text editor in the PC that is connected to the Symbol barcode reader. Point the Symbol barcode reader on D8 BAR LED of the iCE40 Ultra Mobile Development Platform. Note that the text on the text editor matches the text in the Text Box UPC-A code in the application. This confirms that the MoBeam code on the board is functioning.
11. Select the **Stop** button to stop beaming the data. The BAR LED is turned off. You can perform the same procedure for other Barcode standards by selecting them from the list of barcodes.

Figure 21. Beaming of Data Stopped

Figure 22. LED Indicator when Beaming Data

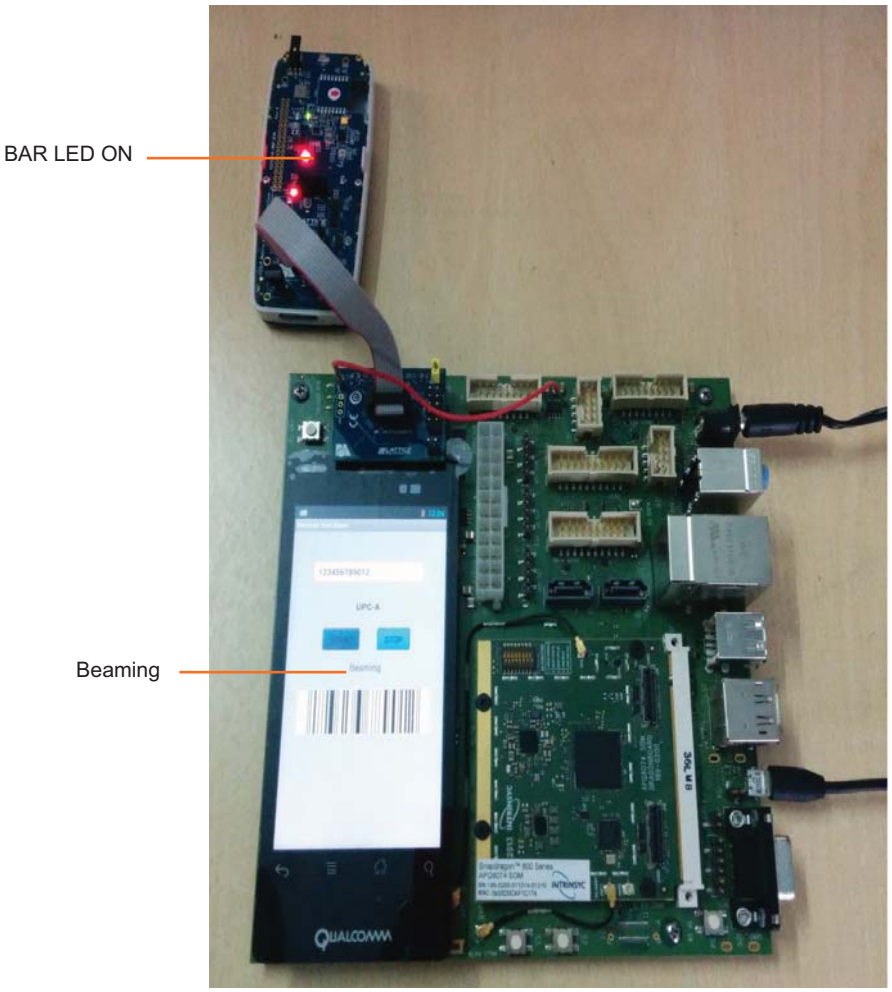
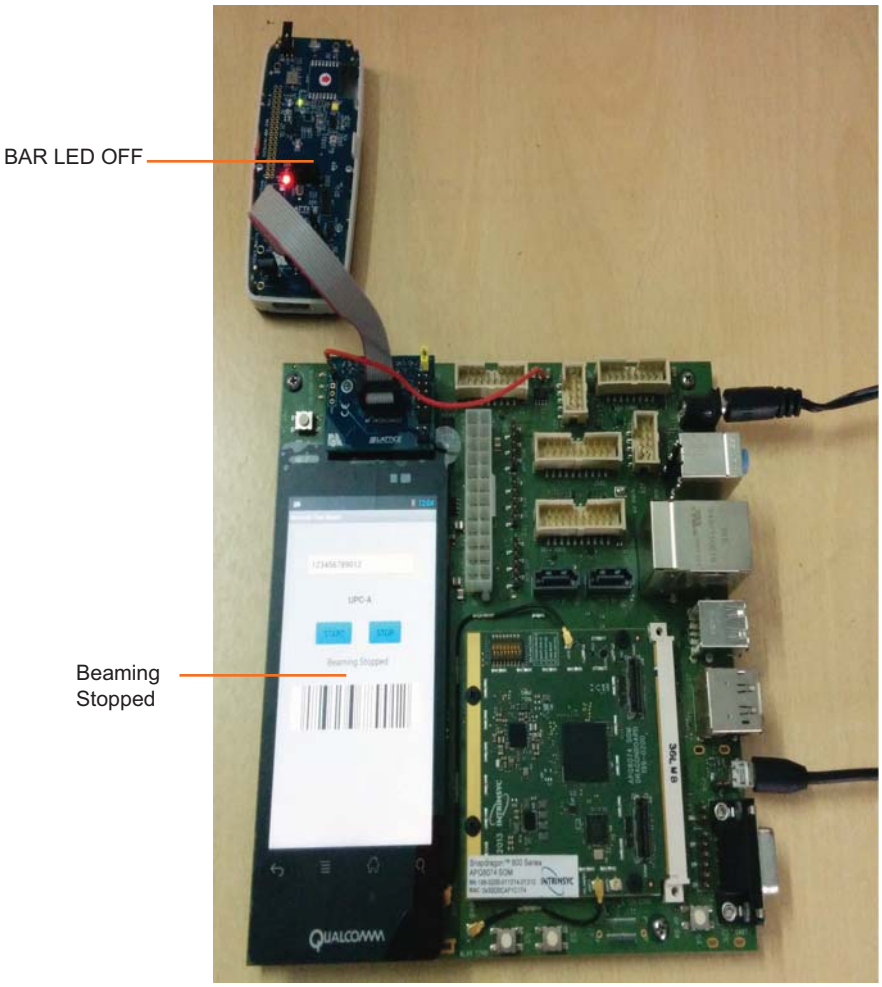


Figure 23. LED Indicator when Beaming Data is Stopped

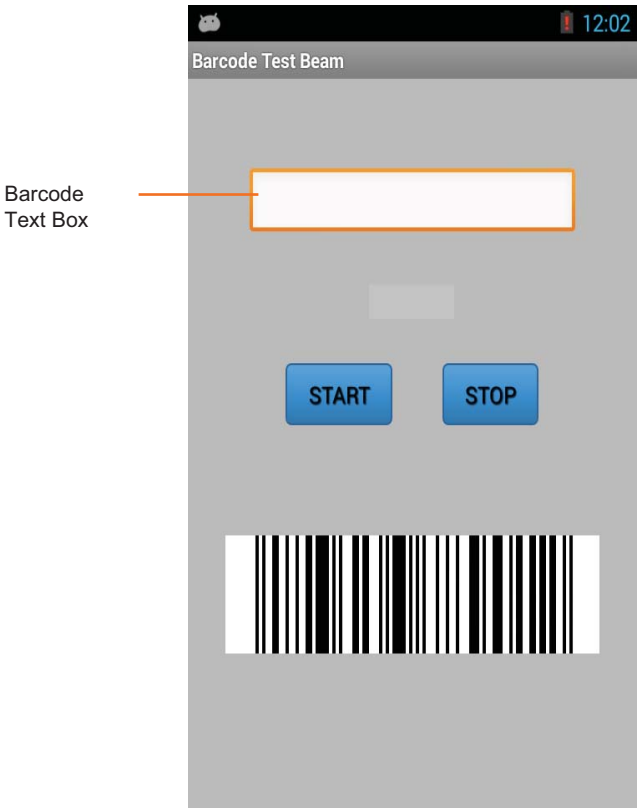


Barcode Emulation Demo Application Features

The following are the features of the Barcode Emulation Demo application:

- The Barcode Emulation application supports eight Barcode standards which are used worldwide such as UPC-A, EAN, Code-39, Code-128, I 2 of 5, Codebar and GS1 Databar.
- A text box is provided where you can enter a valid barcode text for testing, as shown in Figure 24.

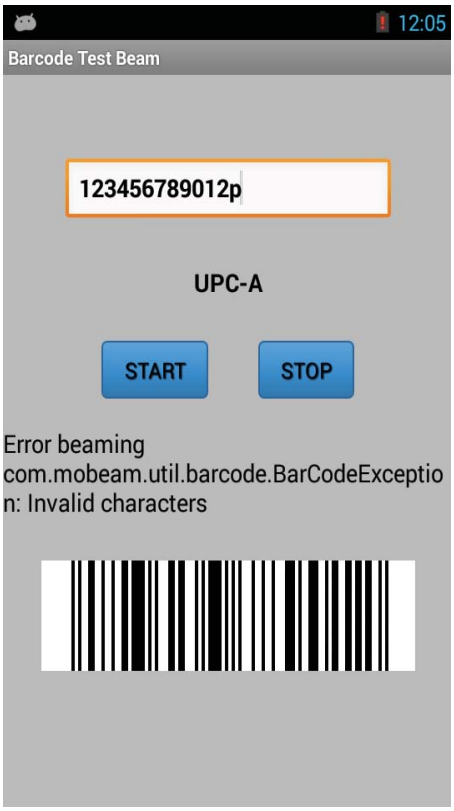
Figure 24. Barcode Text Box



- The barcode beaming can be started and stopped at any point.
- Every Barcode standard has a minimum and maximum number of digits and valid characters. You can check on the internet whether the barcode used is valid or not. The application displays appropriate error messages if you enter invalid barcode text.

  For example, for the UPC-A standard, entering the character “P”, as shown in Figure 25, results in an error for invalid characters as this barcode only allow numeric characters.

Figure 25. Invalid Barcode Character



Troubleshooting

If the Android application does not respond, perform the following procedure:

1. Close the Barcode Emulator process running in background.
2. In the Android menu, select System settings > Applications > Manage Applications > Barcode Test > Force Stop
3. Restart Intrinsyc DragonBoard.
4. Open the Barcode Test Demo application from the Android menu. Beam the barcode data on the iCE40 Ultra Mobile Development Platform.

Barcode Emulation Design

The following section describes the internal details of the Barcode Emulation design.

Overview

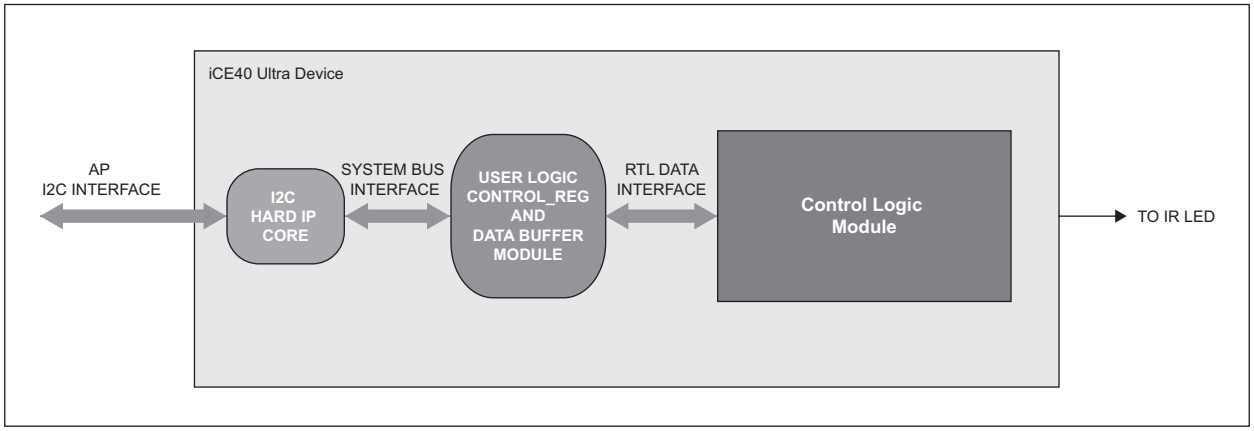
This user's guide contains an example barcode emulation technique wherein the barcode pattern is beamed using LED. The LED ON represents no bar and LED OFF represents bar, making it barcode standard agnostic. This example barcode emulation technique is used in smart phones and handheld devices without change on the existing hardware.

The example barcode emulation technique uses I2C protocol between Application Processor and a barcode emulation ASIC. We are using the iCE40 Ultra device for ASIC implementation. There are defined I2C register sets of ASIC to which the AP writes data.

The functional registers are beamable sequence register, beamable data register, number of HOPs, number of Packets and number of Symbols.

The Control Logic Module accesses this data and configures the timings of ON/OFF time and sends the beamable pattern to LED, thus toggling the LED as barcode lines.

Figure 26. IP Core RTL Design Block Diagram



Note: i_clk will from external oscillator

Design Features

- I2C Compliant Design
  - 7-bit addressing
  - Supports operations at 100 KHz and 400 KHz
- Example Barcode Emulation Design
  - Standard I2C interface for read and write operations
  - Beaming Start and Stop commands
  - Revision code read back
  - LED polarity is active high or low
  - Reset command

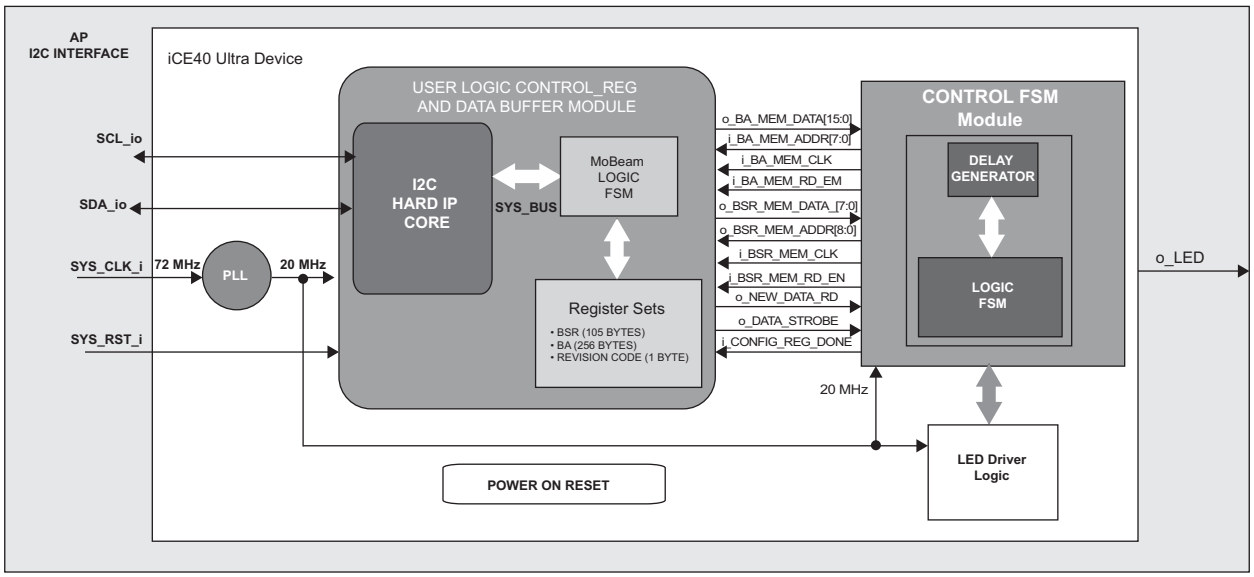
Functional Overview

The design consists of the following components.

There are three main blocks for the RTL design:

- USER_LOGIC_CONTROL_ REG_DATA_BUF MODULE
- CONTROL FSM MODULE
- LED DRIVER LOGIC MODULE

Figure 27. I2C Slave, User Control Logic and Data Register, and Logic Modules Interface Logic Diagram



The modules are described below:

1. **USER_LOGIC_CONTROL_ REG_DATA_BUF MODULE** – consists of the Hard I2C block, I2C slave data FSM and register sets.
    a. HARD I2C block – this is a built in I2C hard block feature of the iCE40 Ultra device. Input to this module are connected I2C lines and output from this module are wishbone system bus signals for user logic.
    b. I2C slave data FSM – this module gets the I2C data from the hard IP block. Input to this module are system bus signals and output from this module consists of valid I2C data address and data.
    c. Register sets – this module defines the storage of Register sets.There are two Embedded Block RAMs in this module. One for BSR (Beam Sequence Register) memory and another for BA (Beamable Array) memory data. This module also checks for Start and Stop commands, reset command, LED polarity and Revision Code.

Figure 28. RTL Code Signal Description for "user_logic_control_reg_data_buf.v"

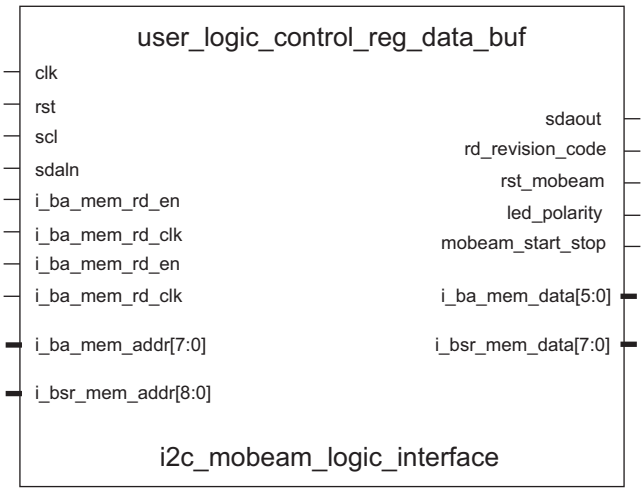


Table 2.

PORT NAME	DIRECTION	DESCRIPTION
clk	Input	Input clock.
rst	Input	Active High Reset.
scl	Bi-Dir	I2C clock line
sdaIn	Input	I2C data line
i_ba_mem_rd_en	Input	Read enable for BA Memory (HIGH=READ ENABLE)
i_ba_mem_rd_clk	Input	Read clock to BA memory
i_bsr_mem_rd_en	Input	Read enable for BA Memory (HIGH=READ ENABLE)
i_bsr_mem_rd_clk	Input	Read clock to BSR memory
i_ba_mem_addr[7:0]	Input	Input address from mobeam_control_fsm to read BA data from BA memory.
i_bsr_mem_addr[8:0]	Input	Input address from mobeam_control_fsm to read BA data from BA memory.
sdaout	Output	I2C data line
rd_revision_code	Output	Indicates revision code is placed on i2c_slave_to_master_data_o[7:0] bus for master to read. This signal is asserted /deasserted based on the command received from I2C master and decoded by mobeam_i2c_reg_interface.
rst_mobeam	Output	When HIGH, resets the mobeam_control_fsm. This signal is asserted/deasserted based on the command received from I2C master and decoded by mobeam_i2c_reg_interface.
led_polarity	Output	When LOW, indicates the led_driver to invert LED polarity. This signal is asserted/deasserted based on the command received from I2C master and decoded by mobeam_i2c_reg_interface.
o_ba_mem_data[15:0]	Output	BA memory data read by mobeam_control_fsm on asserting i_ba_mem_rd_en.
o_bsr_mem_data[7:0]	Output	BSR memory data read by mobeam_control_fsm on asserting i_bsr_mem_rd_en.

2. **CONTROL FSM MODULE** – consists of the delay generator and the Logic FSM.
    a. Delay generator – this module mainly generates the inter-symbol delay (ISD) and packet delay (IPD) as defined by the application. Input to this module are clock, ISD, and IPD values from the Register set modules and outputs from this module are the delay c.
    b. Logic FSM – this module is the control FSM for the example barcode emulation technique. Inputs to this module are the accessing of the registers stored in BSR and BA memories, start and stop, and reset commands. Outputs from this module provide the BEAMABLE data and timing signals to the LED Driver module.

Figure 29. RTL Signal Description of Mobeam Control FSM

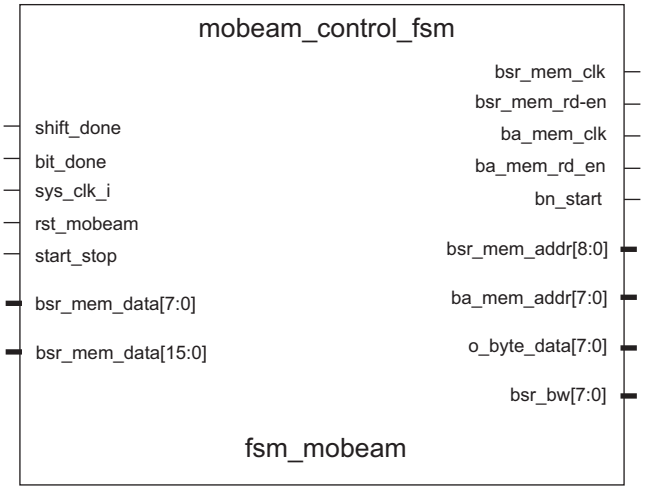


Table 3.

PORT NAME	DIRECTION	DESCRIPTION
sys_clk_i	Input	Input System clock.
rst_mobeam	Input	Active high Reset from mobeam_i2c_reg_interface.
shift_done	Input	Signal from led_driver to indicate a byte of data is successfully beamed by output led.
bit_done	Input	Signal from led_driver to indicate one bit is successfully beamed by output led.
start_stop	Input	Issued by mobeam_i2c_reg_interface. When HIGH, indicates mobeam_control_fsm to start fetching the BSR, BA data and start beaming the data.
bsr_mem_data[7:0]	Input	Input BSR data from BSR memory.
ba_mem_data[15:0]	Input	Input BA data from BA memory.
bsr_mem_clk	Output	Read clock to BSR memory
bsr_mem_rd_en	Output	Read enable for BSR Memory (HIGH=READ ENABLE)
ba_mem_clk	Output	Read clock to BA memory
ba_mem_rd_en	Output	Read enable for BA Memory (HIGH=READ ENABLE)
txn_start	Output	When HIGH, indicates valid data on data bus to led_driver logic. Time for which this signal remains LOW can be used to measure ISD or IPD delays.
ba_mem_addr[7:0]	Output	Address to BA memory to read BA data.
bsr_mem_addr[8:0]	Output	Address to BSR memory to read BSR data.
o_byte_data[7:0]	Output	Beamable data to led_driver logic.
bsr_bw[7:0]	Output	Bar Width data to led_driver logic.

3. LED Driver Module – this module consists of the timing signals and beamable data inputs, which drives the oled signal. Below gives the signal description of the module.

Figure 30. LED Driver Module

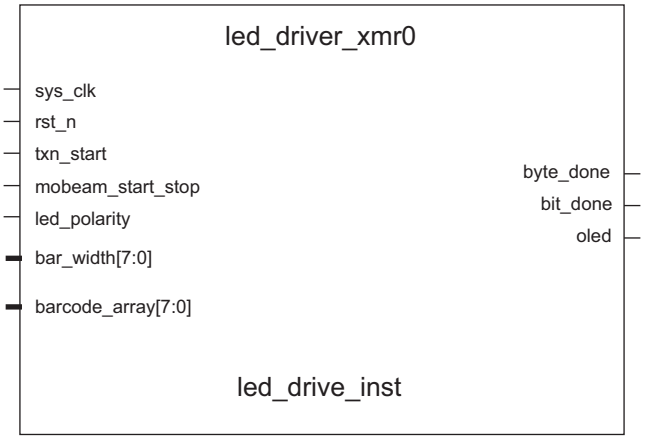


Table 4.

PORT NAME	DIRECTION	DESCRIPTION
sys_clk_i	Input	Input System clock.
rst_n	Input	Active low Reset.
txn_start	Input	When HIGH, indicates valid data on data bus to led_driver logic. Time for which this signal remains LOW can be used to measure ISD or IPD delays.
mobeam_start_stop	Input	Issued by mobeam_i2c_reg_interface. When HIGH, indicates to start beaming the data received from mobeam_control_fsm.When LOW indicates mobeam_control_fsm to stop beaming the data.
led_polarity	Input	Issued by mobeam_i2c_reg_interface. When LOW, indicates the led_driver to invert LED polarity. When HIGH, polarity remains the same.
bar_width[7:0]	Input	Bar Width data from mobeam_control_fsm.
barcode_array[7:0]	Input	Beamable data received from mobeam_control_fsm.
byte_done	Output	Signal to indicate a byte of data is successfully beamed by output led.
bit_done	Output	Signal to indicate one bit is successfully beamed by output led.
oled	Output	Serial led output.

Pin Table

Table 5 provides the pins on the iCE40 Ultra Mobile Development Platform.

Table 5. Pinout for Target System

Pin Name	Direction	Pin Description
i_clk	Input	System clock
io_scl	Inout	I2C Clock
io_sda	Inout	I2C Data
o_led	Output	BAR LED
drive_on	Output	Test point

I2C Slave

The configuration data, beam sequence register and beamable data are communicated by the application processor to the FPGA using the I2C lines. Most of the communication is write operations by master. The write transaction is as follows.

I2C Write

Configuration data – the master sends the slave address followed by the data address and the data as shown in Figure 31.

Figure 31.

NAME	DATA ADDRESS	DATA SIZE	FUNCTION
BEAM START STOP (BSS)	0XF0	1 BYTE	COMMAND DATA

Table 6.

NAME	DATA ADDRESS	DATA SIZE	FUNCTION
LED BRIGHTNESS (LB)	0xEA	1 BYTE	CONFIG DATA
LED POLARITY (LP)	0XEB	1 BYTE	CONFIG DATA
BEAM LENGTH (BL)	0XEC	1 BYTE	CONFIG DATA
NO. HOPS (NH)	0XED	1 BYTE	CONFIG DATA

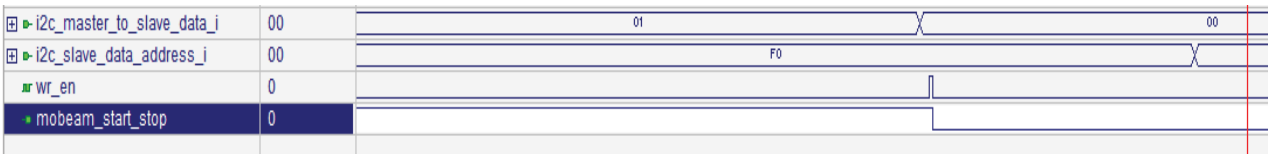
The simulation waveforms below show the I2C master to slave data.

Mobeam_start_stop signal going High to start beaming, master to slave data is 0x01

Figure 32. Simulation Waveform

Mobeam_start_stop signal going Low to stop beaming, master to slave data is 0x00

Figure 33. Simulation Waveform

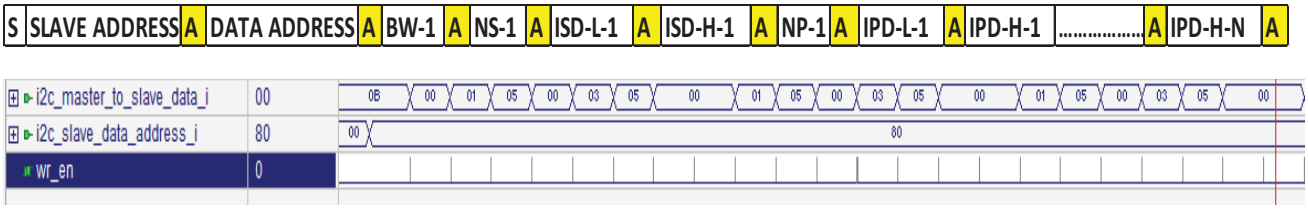


BSR Data Write

After config data, BSR data I2C master write. The BSR data is written in burst mode to speed the write operation, the data address for BSR is from 0x80 to 0xE8. The I2C write starts from 0x80 followed by BSR register data. In this we write HOP data (BW) Bar Width - 1 byte, (NS) Number of symbols - 1 byte, (ISD) inter-symbol delay - 2 bytes, (NP) Number of packets - 1 byte, (IPD) inter-packet delay -2 bytes. In the MoBeam I2C register logic, you auto increment address.

In Figure 34, the simulation waveform shows the sequence of the BSR data.

Figure 34. Simulation Waveform

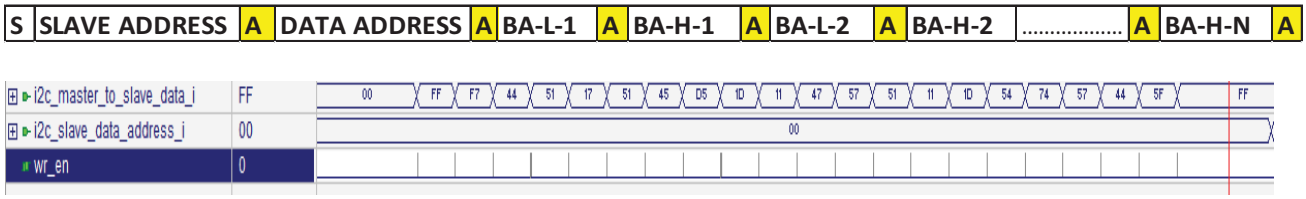


BA Data Write

After BSR data, BA data I2C master write. The BA data is written in burst mode by giving the base address followed by data. The data address for BA is from 0x00 to 0x7F. In the MoBeam I2C register logic we auto increment the memory address.

In Figure 35, the simulation waveform shows the sequence of the BA data.

Figure 35. Simulation Waveform



Register Set

A set of ten registers available to the application processor. Each register has a width of 8 bits. The register mapping is shown in Table 7.

Table 7. Register Address

SEQUENCE NO	REGISTER SETS	DATA ADDRESS	DATA SIZE	REMARKS
1	LED BRIGHTNESS (LB)	0xEA	1 BYTE	
2	LED POLARITY (LP)	0XEB	1 BYTE	
3	BEAM LENGTH (BL)	0XEC	1 BYTE	
4	NO. HOPS (NH)	0XED	1 BYTE	

SEQUENCE NO	REGISTER SETS	DATA ADDRESS	DATA SIZE	REMARKS
5	NO. REPEATS (NR)	0XEE	1 BYTE	
6	BEAMABLE SWEEP REGISTER (BSR)	[0X80 - 0XE8]	<=105 BYTES	MoBeam DEFINED
7	BEAMABLE ARRAY (BA)	[0X00 - 0X7F]	<=256 BYTES	MoBeam DEFINED
8	BEAM START STOP (BSS)	0XF0	1 BYTE	
9	RESET MOBEAM (RM)	0XF1	1 BYTE	
10	REVISION CODE (RC)	0XF2	1 BYTE	

LED Driver Output

The example code, after writing the Configuration register, BSR data and BA data, issues a command to Start Beaming on BEAM START STOP register. This asserts the signal mobeam_start_stop and “oled” output drives the LED as shown in Figure 36. oled_check is parallel data of oled shown below is the same as the BA data received.

Figure 36.

Once beaming is completed, the 0x00 on BEAM START STOP register, which issues the command to stop beaming, and the mobeam_start_stop signal go low as shown in Figure 37 “oled” goes low, this an asynchronous command issued by the AP.

Figure 37.

Resource Utilization

Table 8. Resource Utilization

LUTs	PLBs	BRAMs	I/Os	I2C	SPI
889	199	2	5	1	0
886	207	2	7	0	1

Board Information

For more information on procuring the iCE40 Ultra Mobile Development Platform, please contact your local Lattice Sales Representatives.

For more information on Snapdragon Board APQ8074, please go to www.intrinsyc.com.

Demo Associated Files

- RTL Code (Verilog) with testbench
- Apk files and source code

Documentation

- iCE40 Ultra Mobile Development Platform

References

- DS1048, iCE40 Ultra Family Data Sheet
- www.wikipedia.org

Technical Support Assistance

e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2014	1.0	Initial release.