Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE, Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC32 logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, ETHERSYNCH, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and QUIET-WIRE are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestiC is a registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2016, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0979-3
Table of Contents

Chapter 1. Overview
1.1 Introduction ..................................................................................................... 5
1.2 LED Cube Features ........................................................................................ 6
1.3 LED Cube Kit Contents .................................................................................. 6
1.4 Design Files .................................................................................................... 6
1.5 MPLAB Harmony Features Used ................................................................. 6

Chapter 2. Assembly Instructions
2.1 Introduction ..................................................................................................... 7
2.2 File Locations ................................................................................................. 7
2.3 Necessary Tools ............................................................................................. 7
2.4 LED Lead Forming ......................................................................................... 8
2.5 LED Matrix Wire Forming and Cutting .......................................................... 14
2.6 LED Matrix Assembly ................................................................................... 16
2.7 Soldering the Planes to the PCB ................................................................... 27
2.8 Replacing an LED on the LED Cube ............................................................ 31

Chapter 3. Programming
3.1 Introduction ................................................................................................... 37
3.2 File Locations ............................................................................................... 37
3.3 Development and Firmware Download Tools .............................................. 37
3.4 Reprogramming the LED Cube .................................................................... 38
3.5 Reprogramming the LED Cube with an MPLAB X IDE Programmer ........... 40
3.6 Project Configuration .................................................................................... 41

Chapter 4. Custom Scene Creation
4.1 Introduction ................................................................................................... 43
4.2 Example Scene Overview ............................................................................ 43
4.3 Example Scene Detail .................................................................................. 44
4.4 Manipulating the Cube ................................................................................. 45

Appendix A. Frequently Asked Questions
A.1 Introduction .................................................................................................. 49
A.2 FAQ List ....................................................................................................... 49

Appendix B. Command Line Interpreter
B.1 Introduction .................................................................................................. 51
B.2 LED Cube PCD to PC USB Connection ...................................................... 52
B.3 Connecting to the Terminal Program ........................................................... 53
Appendix C. PCB Test

C.1 Introduction .................................................................................................. 55
C.2 Equipment and Software Requirements .................................................... 55
C.3 Hardware Connections ................................................................................ 56
C.4 Program PCB Device using MPLAB IPE .................................................. 57
C.5 Test Board using CoolTerm ....................................................................... 59
C.6 Error Conditions ......................................................................................... 62

Appendix D. Drawings & Schematics

D.1 Introduction ............................................................................................... 63
D.2 PCB Schematic and Drawings .................................................................... 63
D.3 LED Cube Schematic and Diagram ........................................................... 67

Index .................................................................................................................. 69
Worldwide Sales and Service ............................................................................ 70
1.1 INTRODUCTION

The 5x5x5 RGB LED Cube kit (AC100200) was inspired by an 8x8x8 LED cube that was built by Steven Bible (Senior Staff Technical Training Engineer at Microchip).

The LED cube firmware contains over 12 custom repeating scenes. You can also create your own scenes.

Firmware for the PIC32 microcontroller (MCU) on the printed circuit board (PCB) of the cube was developed using MPLAB® Harmony Configurator (MHC). See how the cube was programmed by accessing the project in MPLAB X Integrated Development Environment (IDE) through MHC. That is also where you can modify the firmware.

Begin by assembling the LED cube. Then, either enjoy the default operation of the cube or develop and program your own scenes!

The following topics are included in this overview:

• LED Cube Features
• LED Cube Kit Contents
• Design Files
• MPLAB Harmony Features Used

FIGURE 1-1: 5x5x5 RGB LED CUBE
1.2 LED CUBE FEATURES

The LED cube has the following features:

- 5x5x5 wire framework (12.7 cm x 12.7 cm x 12.7 cm) of RGB LEDs
  - 125 LEDs with 375 RGB connections
  - 5-levels with common drive
  - 75 12-bit pulse-width modulators (PWMs)
  - generates 68 billion colors
- Occupancy and motion detection with a passive-infrared (PIR) sensor that turns the cube on and off
- USB power, bootloader, and control
- Light sensor for automatic dimming
- Bluetooth® Serial Port Profile (SPP) control via an RN4677 module (for Android and PC, only)
- PIC32MX270F256B MCU with 256K of Flash memory for custom scene creation
- Simple scene-creation application program interfaces (APIs) for the development of custom scenes

1.3 LED CUBE KIT CONTENTS

The LED Cube kit (AC100200) includes the following:

- Assembled printed circuit board (PCB) with a PIC32MX270 MCU, an RN4677 module, an MCP1703 voltage regulator, a light sensor, and LED drivers
- 125 T5 RGB LEDs
- PIR sensor
- 40' (12.192 m) of 20-gauge (0.81 mm) tinned solid copper wire

Assembly instructions are presented in Chapter 2. “Assembly Instructions”.

1.4 DESIGN FILES

Design files can be downloaded from the Microchip website:

http://www.microchip.com/LEDCube

The following support documents are available on the website:

- Schematics
- PCB layout
- Source code
- Printable 3D models for cases and LED lead-forming fixtures
- Pictures of completed RGB LED cubes

1.5 MPLAB HARMONY FEATURES USED

The following MPLAB Harmony features were used to complete this kit:

- MHC configuration of the UART, USB, SPI, DMA, timer, output compare (OC) module and PWM
- USB Boot Loader
- DMA, Timer, SPI, and OC peripheral libraries (PLIBs)
- USB and UART drivers
Chapter 2. Assembly Instructions

2.1 INTRODUCTION

The following sections provide instructions to assemble the 5x5x5 RGB LED Cube kit (AC100200). Alternate methods of construction are also provided, so you can choose the best way for you to proceed.

- File Locations
- Necessary Tools
- LED Lead Forming
- LED Matrix Wire Forming and Cutting
- LED Matrix Assembly
- Soldering the Planes to the PCB
- Replacing an LED on the LED Cube

2.2 FILE LOCATIONS

Documentation and software for the assembly of the LED cube can be downloaded from the Microchip website:

http://www.microchip.com/LEDCube

These files include Eagle libraries, schematics, Gerber files, bill of materials, the presentation

2.3 NECESSARY TOOLS

You need the following tools to complete your LED cube. Additionally, you need a fixture to bend the LED leads and a second fixture to hold the LEDs with exact spacing as you solder each LED plane together.

- Soldering iron and solder
- Wire cutters
- Needle nose pliers
- Four clip leads to test LEDs planes before soldering to PCB
- LED Lead forming fixture (see Section 2.4 “LED Lead Forming”)
- LED matrix holding fixture (see Section 2.6 “LED Matrix Assembly”)
- 7 to 10 hours of assembly time
2.4 LED LEAD FORMING

The LEDs have (4) leads as shown in Figure 2-1:

- Red (cathode)
- Common Anode - longest lead
- Green (cathode)
- Blue (cathode)

FIGURE 2-1: RGB LED

The LEDs have to be formed (their leads have to be bent) into a shape that allows them to be soldered into a matrix and allows them to interface with the PCB. There are two methods for forming the LEDs. The first method requires fixtures that are made with a 3D printer. The second method provides an alternate way to form the leads without using 3D printed fixtures.

2.4.1 LED Lead Forming Using 3D Printed Fixture

You will find a design for a 2-piece lead forming tool at the link provided in Section 2.2 “File Locations”. The LED forming tool is shown in Figure 2-2. It is composed of two parts which slide together to form the leads.

Note: Use caution when forming the LEDs with the 3D printed fixture. Two LED cubes have been successfully assembled and operated without any issues using this method. However, a third cube assembled using this fixture and a different batch of LEDs had some latent failures in the LED leads which may have been due to mechanical stresses placed on the leads during the forming process.
The lower part has a LED cradle in which the LED is placed as shown in Figure 2-3. The slots for the leads are different lengths to match the length of the leads on the LEDs. Make sure that the LED is oriented according to the slots. The longest lead does not have a slot but sits on a shelf. This lead is not bent during the first bending operation.

**FIGURE 2-3: LED LEAD FORMING FIXTURE - LOWER HALF**

Align the top part with the guides on the bottom part as shown in Figure 2-4. There is a wedge which separates two of the leads. Make sure this goes between the two leads and does not sit on top of one of them.

**FIGURE 2-4: TOP AND BOTTOM PART INITIAL ALIGNMENT**

Carefully press the top half all the way down on the bottom half. Hold the LED in the cradle with your thumb while doing this to prevent it from flipping upward as the leads are bent. Three of the leads will be bent down into the cavity while the remaining lead will be left straight. Figure 2-5 shows a top view of the fixture after the two parts have been pressed together.
FIGURE 2-5: LED FORMING FIXTURE TOP VIEW AFTER PRESSING TWO PARTS TOGETHER

Remove the top part. The leads should appear as depicted in Figure 2-6. One lead is bent slightly outward so that the spacing between the three bent leads is approximately 0.1" (2.54 mm).

FIGURE 2-6: LED WITH THREE LEADS BENT IN FIXTURE

With needle-nosed pliers, bend the fourth lead sideways so that it is directly over the guide slot in the bottom section as shown in Figure 2-7. At this point, the LED is held in place rather tightly because the leads are bent underneath it at a greater than 90 degree angle. Remove the LED from the forming tool by poking from behind through the hole in the LED cradle or by prying it out with a knife.
The LED should now look as depicted in Figure 2-8. The leads form a Z-bend and point downward after this operation. The Z-bend using the 3D printed fixture is slightly different than the J-bend with the leads pointing up which is obtained using the alternative fixture (Section 2.4.2 “LED Lead Forming Using Alternative Fixture”). The leads can be trimmed to 1/8” at this time.

2.4.2 LED Lead Forming Using Alternative Fixture

If you don’t have a 3D printer, you can create a lead forming fixture with a 1/8” (3.175 mm) aluminum bar (which can be purchased at any home center) as shown in Figure 2-9. This bar can also be used as a measuring stick when cutting the matrix support wires.

Using a thin blade hacksaw, cut a slot ~3/16” (4.2765 mm) from the edge. The slot has to be at least 3/8” (9.525 mm) deep. This slot has to be wide enough to accommodate the LED lead thickness and long enough to accommodate all 4 leads. The slot width should be just slightly wider than the LED thickness. Test your LEDs to make sure you can insert them into the slot.
Insert your LED so that common anode (longest lead) is closest to the outside of the fixture as shown in Figure 2-10.

**FIGURE 2-10: LED INSERTION DIRECTION**

Bend the two leads adjacent to the common anode (longest lead) forward and the lead farthest from the edge parallel to the bar so they look as shown in Figure 2-11. DO NOT bend the common anode (longest lead) at this time.

**FIGURE 2-11: FIRST SET OF BENDS**

Using pliers, bend the left most lead forward so it is parallel to the two forward bent lead from the prior step as shown in Figure 2-12. These three leads need to have 0.1” (2.54 mm) spacing between them at this time.

**FIGURE 2-12: SECOND SET OF BENDS**
Now fold the three 0.1” (2.54 mm) spaced RGB cathode leads around the fixture so they are parallel to the LED body itself as shown in Figure 2-13. Using a pair of pliers, bend the common anode lead to the right. This bend should be ~0.1” (2.54 mm) above the fixture surface.

FIGURE 2-13: THIRD SET OF BENDS

Trim three 0.1” (2.54 mm) spaced RGB cathode LED leads so they are parallel to the fixture bar top. Also, trim the common anode lead to 1/8” (3.175 mm) in length as shown in Figure 2-14.

FIGURE 2-14: LEAD TRIMMING

Remove the LED from the fixture by sliding it out to the right. The formed LED will look as shown in Figure 2-15 when removed from the fixture. Be careful not to damage the leads.

FIGURE 2-15: FORMED LED
2.5 LED MATRIX WIRE FORMING AND CUTTING

Now that the LEDs have been formed, the wires used to connect them together into the matrix must be prepared.

Twenty-gauge copper tinned wire with 0.032" (0.81 mm) thickness is used. The wire will have natural curves and irregularities which can cause problems creating a perfectly straight matrix. It is important to straighten your wire prior to cutting each matrix segment piece.

The wire can be straightened by “stretching” it. You can use a piece of furniture (e.g., a pinball machine leg) to aid you.

1. Connect one end of the wire to the piece of furniture closest to the floor. Make sure this is a very solid connection by wrapping the wire around the piece and using a strong clamp.
2. Grab the other end of the wire and pull it. As you pull with more force, you will start to “feel” the wire stretch and become longer. Once you have stretched the wire, stop pulling. Continuing to pull or pulling too hard could cause the wire to break.
3. Disconnect it from the piece of furniture and verify that it is perfectly straight on the floor.
4. If it is not straight, repeat the pulling/stretching operation.
5. Once it is perfectly straight, use a length stick to cut each length to 4.1” (10.414 cm). This length is important as you will run out of wire if you cut these too long!
6. You will need to cut a total of 106 wires this length. Save the remaining uncut wire as you will need a remaining 5 wires to connect the common matrix to the PCB, but these will be custom cut to length as you insert them.

There is an alternate method if you don’t have enough space to pull the wire all at once:

1. Clamp one end of the wire in a bench vise or similar clamping tool.
2. Pull several feet at a time by wrapping the other end around a section of PVC pipe and using the pipe as a handle for pulling the wire (see Figure 2-16). The pipe prevents kinking but still allows you a way to pull tightly.
3. The wire rolls around the PVC pipe straighten easily when the next pull is made.
FIGURE 2-16: WIRE PULL WITH A CLAMP AND PIPE
2.6 LED MATRIX ASSEMBLY

Now that you have formed the leads of your LEDs, and stretched and cut your matrix wires, you need a fixture to hold the LEDs and wires in a consistently-spaced matrix. The LED-to-LED spacing is 0.85" (21.59 mm). The LED cube will look its best if this spacing is maintained in all 3 dimensions. Once again there is a 3D printable fixture for soldering the LEDs together and an alternative method if 3D printing is not available to you.

2.6.1 3D Printed Matrix Holder Soldering Fixture

The LEDs need to be assembled in sections. First five vertical stacks are soldered. Then, these five vertical stacks are connected using the LED common anodes to create a 5x5 LED plane. The vertical stacks and 5x5 plane are shown in Figure 2-17 and Figure 2-18. You will find a design for a plane-forming jig at the link provided in Section 2.2 "File Locations". For a schematic of a plane, see Figure D-5.

FIGURE 2-17: VERTICAL STACKS
The design of the plane forming jig uses 5.1mm diameter LED holes. Actual size may vary, depending on the 3D printer. The LEDs should fit into the holes with a slight interference fit so they are held in place during soldering. If the holes are too small, you may have to ream them out using an appropriately-sized drill bit or a small round file.
Figure 2-19 shows the printed jig with one LED inserted. The leads of the LED have not been trimmed yet as this more clearly illustrates orientation. However, you should trim the leads before inserting the LEDs into the jig.

**Note:** The length of the trimmed leads on the LEDs will eventually determine how high the LED matrix sits above the PCB. If you trim the length of the three cathode leads from about 3/16 to 1/4 inch (4 to 5mm) after the last bend, this should give you enough soldering length while still allowing the bottom plane of LEDs to sit fairly close to the PCB.

**FIGURE 2-19: PRINTED JIG WITH ONE LED INSERTED**
Insert the LEDs one vertical row at a time and solder the connecting wires, leaving enough at the bottom to be inserted into the holes in the circuit board. Figure 2-20 shows two completed vertical stacks and the third inserted and ready to solder. Again, the perpendicular-bent leads have not been trimmed yet to better illustrate their orientation. Complete all 5 stacks.

FIGURE 2-20: VERTICAL STACK ASSEMBLY
Use additional wires to solder the common anodes together in each row, as shown in Figure 2-21. This figure only highlights one of the wires used to connect the common anodes together, but there is an additional wire for each row.

This completes the assembly of the 5x5 plane, but don't remove it yet. It is best to test it before removing it from the fixture. See Section 2.6.3 “Testing a 5x5 LED Matrix”.
Figure 2-22 shows the connections for testing the first vertical stack. Once this vertical stack has been tested, move the RGB connections and test the remaining 4 vertical stacks using the same technique.

**FIGURE 2-22: VERTICAL STACK 1 TEST CONNECTIONS**

By testing before the LEDs are removed from the jig, you have the best chance to make any fixes with the least amount of work. Make sure all the LEDs work and all the solder joints look good, as it will be nearly impossible to fix a solder joint once the entire cube is assembled. At this point the array can be carefully removed from the jig, and the next array can be started. A completed 5x5 plane is shown in Figure 2-23. You will need to make five of these 5x5 planes.

**FIGURE 2-23: COMPLETED 5X5 PLANE**
2.6.2 Alternative Matrix Holder Soldering Fixture (No 3D Printer Required)

If you don't have a 3D Printer, you can make one from wood or cardboard. A wood fixture is shown in Figure 2-24.

**FIGURE 2-24: WOOD 5X5 PLANE FIXTURE**

It was created out of the following material:

- 4.5" x 4.5" x .75" (11.43 cm x 11.43 cm x 1.9 cm) block of scrap pine wood
- 4.5" x 4.5" x 4.5" (11.43 cm x 11.43 cm x 11.43 cm) scrap of 3/16" (4.763 mm) plywood

using the following procedure:

1. Use a table saw or router to cut five 3/16" x 3/16" (4.763 mm x 4.763 mm) slots each spaced 0.85" (2.159 cm) apart.
2. Cut five pieces to measure 1" x 4.5" (2.54 cm x 11.43 cm) from the 3/16" (4.763 mm) plywood
3. Drill five 5 mm holes into each strip spaced 0.85" (2.159 cm) apart and 3/16" (4.763 mm) from the edge. The edge spacing must match the spacing of your LED bending fixture so the lead bends line up to the top surface of the strip.
4. Glue each strip into the fixture so they are aligned vertically and horizontally.
Now it is time to start creating the matrix.

1. Insert five LEDs into a vertical stack as shown in the figure below, making sure they are fully seated, with their three RGB formed leads.

2. Place three of your wires so they line up with the three RGB LED formed leads. The excessive length of the wire needs to extend downwards.

3. Solder each LED onto the wire. Triple check the quality of your solder joint since it could be very difficult to repair a cold solder joint within an assembled LED matrix.

It is good practice to test each LED stack once it is assembled. See Section 2.6.3 “Testing a 5x5 LED Matrix”.

You will need to create a total of (25) of these five LED vertical stacks. Your LED vertical stacks should look as shown in Figure 2-25 through Figure 2-28.

FIGURE 2-25: VERTICAL STACK VIEW 1

FIGURE 2-26: VERTICAL STACK VIEW 2
Once you have all 25 LED stacks assembled, it is time to create your LED planes. Assemble five LED stacks into the fixture, all fully seated as shown in Figure 2-29.

Once you have all 25 LED stacks assembled, it is time to create your LED planes. Assemble five LED stacks into the fixture, all fully seated as shown in Figure 2-29.
Solder a common wire across the back side (orthogonal to the three RGB wires) to connect each plane of LEDs together. This will involve five common wires per plane. Figure 2-30 shows how the stacks are connected together by wiring the common anodes together to form a plane; the side containing the excess lead length is shown in the figure.

**FIGURE 2-30:** COMPLETED 5X5 PLANE USING WIRES CONNECTING COMMON ANODES TO FORM THE STACKS INTO A PLANE

Each plane can now be re-tested using the test procedure that is described in Section 2.6.3 "Testing a 5x5 LED Matrix".
2.6.3 Testing a 5x5 LED Matrix

To test a 5x5 LED matrix plane before removing it from a holder:

- Plug the PCB into the USB power supply
- Press "Mode" button to enter test mode
- Solder the four alligator clip wires onto the CA, R, G and B pads of the LED test board area:

![LED Test Board](image)

Test each of the 5 vertical stacks in the following manner.

1. Connect the RGB LED leads of your first vertical stack to the R, G and B alligator test leads.
2. Connect the CA alligator clip common connection sequentially to each of the 5 LED common anode connections and verify you see the RGB pattern at each location before moving on to the next.
3. The RGB test pattern will end 3 minutes after pressing the Mode button. Press Mode to restart the test pattern if you fail to see the RGB test pattern.
2.7  SOLDERING THE PLANES TO THE PCB

Now that you have assembled and tested the five planes, it is time to solder them to the PCB. By the time you have completed five planes, you have completed 500 solder joints! The good news is that you are nearly finished and have a very cool cube experience ahead.

The PCB was colored black and there are no components on the LED side of the PCB to create a very clean look. However, during PCB manufacturing and assembly, a label may be placed on this side of the board. If you want to maintain the clean look, you will want to remove this label before you begin soldering the LED planes. Otherwise, it will be very difficult to remove once the LEDs are all in place.

2.7.1  Inserting the 5x5 Planes

Insert the first plane using the following guidelines as depicted in Figure 2-31.

1. The LEDs must face the ICSP connector board side. This will take some careful alignment of the 15 RGB wires, but there should be adequate clearance. If your horizontal anode common wires point toward the front of the cube, instead of the back, in this orientation, please see Section 2.7.2 “Alternate Insertion of 5x5 Planes”.

2. Note the LEDs are on side 2 (opposite the PIC MCU and other components).

3. Make sure you have at least ~0.1” (2.54 mm) to ~0.2” (5.08 mm) of spacing between the common anode level wires and the PCB to prevent shorts.

4. Solder the first LED connection.

5. Make sure the LEDs are perfectly level with the PCB surface and the RGB wires are perfectly orthogonal to the PCB surface.

6. Solder remaining LED connections.

7. Test each plane by applying USB power, pressing the Mode button, and connecting the CA test lead to each of the 5 common anode level wires. Verify each LED sequences RGB.

8. Repeat this for the remaining (4) LED planes, making sure they are spaced 0.85” (2.159 cm) apart and level with each other.
2.7.2 Alternate Insertion of 5x5 Planes

If your horizontal anode common wires point towards the front of the cube, instead of the back, when the plane is oriented as described in Section 2.7.1 “Inserting the 5x5 Planes”, there are two ways to mount the planes:

1. Continue mounting the planes with the LEDs facing the ICSP connector side of the PCB. In this case, the common wires will be routed around the front of the LED array. The common anode level wires will then need to attach to the corresponding anode connection at the back of the array.

2. Mount the planes with the LEDs facing away from the ICSP connector side of the PCB. This will result in the LEDs being on the left side of the column wires instead of the right side. The anode wires can then be connected as described in the previous section. The plane connections on the PCB will now be underneath the LED column, which will take some creative wire routing to connect the PCB to the anode wires on the plane. A suggestion is illustrated in Figure 2-32. In this figure, the common wire comes from underneath the LED and is routed parallel to the cathode wires in this column, so the Red and Blue colors will be reversed in this orientation. As of this publication, a software revision is planned to account for that color reversal.
FIGURE 2-32: ALTERNATE PLANE INSERTION INTO PCB
2.7.3 Wiring Each Level

The matrix has five selection levels, labeled L1 through L5. This requires that the five planes have their common anode level wires connected to their assigned level. The following steps should be taken to connect the common anodes in a level together and then wire them to the board. Figure 2-33 highlights the wiring for Level 2. The remaining levels have a similar wiring procedure.

1. The excess common anode level wires should be protruding toward the rear connector area.
2. Bend the level wires 90 degrees so they create a connection point just in front of the first RGB vertical row.
3. Solder a horizontal wire onto the five common anode wires for the lowest level and make sure the LED spacing is 0.85" (2.159 cm). You may need to re-heat connections and move the matrix slightly to perfect the spacing.
4. Repeat for the remaining levels.
5. Add one more wire across the top level, in the opposite area, to mechanically stabilize the matrix.
6. Wire L1 to the lowest level, L2 to the next level and repeat connecting L5 to the highest level.
7. Plug the cube into USB power.

FIGURE 2-33: LEVEL 2 COMMON ANODE CONNECTION TOGETHER AND TO THE BOARD
2.8 REPLACING AN LED ON THE LED CUBE

Replacing an individual LED on the 5x5x5 cube can be difficult, but it is not impossible. It made immensely easier using the 3D-printed LED replacement alignment jig (Figure 2-34). This model is downloadable from the same location as this document.

FIGURE 2-34: LED ALIGNMENT JIG

Print the alignment jig and ream out the LED holes so that the LEDs fit snugly in the hole, without falling out. Ideally, you should be able to insert an LED in a hole so that it seats all the way to the bottom flange on the LED, and then shake the jig vigorously without the LED moving or falling out. Once complete, you are ready to begin repairs.

1. Remove the bad LED. Reach in with a small diagonal cutters and snip the leads of the LED (Figure 2-35).

FIGURE 2-35: REMOVE BAD LED
An offending LED is shown in Figure 2-36 after being cut out of the array. This LED failed with excessive reverse leakage current in the blue color. In the cube, this LED appeared to operate just fine, but the reverse leakage caused the other four LEDs in its vertical column to glow slightly blue when all the LEDs were supposed to be off. In this case, the bad LED was the one that had no glow.

**FIGURE 2-36: BAD LED**

2. Place two blank alligator clips (without insulation) on either side of the joint being touched by the soldering iron. These clips act like radiators to prevent heat from traveling too far down the wire. Using the clips will reduce the chance of disturbing adjacent joints.
3. Reach in with a hot soldering iron, reflow the joints, and remove the remaining sections of the LED leads (Figure 2-37). You may have to grasp the leads with a tweezers to prevent them from sticking to the wire frames. Be sure to apply as little heat as possible, for as short of time as possible, because the heat traveling down the wire frames can reflow adjacent connections and the internal stresses in the wires can cause these adjacent connections to pop apart as the solder melts.

FIGURE 2-37: REFLOW CUBE JOINTS
4. Form and trim the leads of a new LED. See Section 2.4 “LED Lead Forming” for details on how to do this. Then insert the new LED into the replacement alignment jig in the correct position (Figure 2-38). You will need to visually set the orientation of the leads so that they will line up with the wire frame when the jig is positioned in the LED cube.

**FIGURE 2-38: NEW LED IN JIG**

5. Insert the alignment jig into the LED cube array and position it over the four remaining LEDs in the row (Figure 2-39 and Figure 2-40). Carefully press the jig onto the LEDs so that they fit snugly. You may have to reach in with a screwdriver or similar rigid tool and hold the bottom of each LED as you press down from the top. The alignment jig should be seated on the flange of all four remaining LEDs plus the new fifth LED. You may have to work the jig down a little at a time on each LED to avoid putting too much stress on the wire frame.
FIGURE 2-39: POSITION JIG IN LED CUBE - FULL VIEW

FIGURE 2-40: POSITION JIG IN LED CUBE - CLOSEUP
6. Tweak the leads as necessary to get them to line up with the wire frame and then reach in with a soldering iron and fresh solder to solder each lead in place. Again, use as little heat, for as little time as possible, as the heat traveling down the wire frame could cause adjacent joints to pop apart.

7. Test the cube before removing the alignment jig. If you press the mode button on the PCB, a RGB test pattern will be displayed. As an example shown in Figure 2-41, an adjacent green LED stopped working because of a popped connection due to heat travel. I had to resolder this connection to restore proper operation.

**FIGURE 2-41: REPLACEMENT TEST**

When everything is working to satisfaction, remove the alignment jig. You may have to work each LED out a little at a time to prevent excess bending of the wire frame. The result should be a perfectly aligned LED.
Chapter 3. Programming

3.1 INTRODUCTION

While the LED cube PCB comes pre-programmed, you will need programming ability to update the firmware or develop your own custom scenes.

The following topics are discussed in this chapter:
- File Locations
- Development and Firmware Download Tools
- Reprogramming the LED Cube
- Reprogramming the LED Cube with an MPLAB X IDE Programmer
- Project Configuration

3.2 FILE LOCATIONS

Eagle libraries, schematics, layout, Gerbers, BOMs, pictures and source code can all be downloaded here:
http://www.microchip.com/LEDcube

3.3 DEVELOPMENT AND FIRMWARE DOWNLOAD TOOLS

The LED cube PCB is factory-programmed with a HID bootloader. While that bootloader does not allow real-time debugging of LED cube applications created in MHC, it is quite useful for updating to the latest firmware, or rapidly testing and developing new scenes. See Chapter 4. “Custom Scene Creation”) for information and instructions.

The standard MPLAB X IDE programmer which supports the PIC32MX270F256B MCU can also be used. An MPLAB X IDE programmer has the advantage of reflashing a complete image onto the PIC32 MCU. The programmer also provides real-time debugging support, including hardware breakpoints, single step, and the ability to examine and modify variables.

The following software is required:
- AN1388 Boot loader, PIC32UBL.EXE
  The file is located in the firmware directory. It can also be downloaded from:
- MPLAB X IDE
  The environment can be downloaded from
  http://www.microchip.com/mplab/mplab-x-ide
- MPLAB XC32 C Compiler
  The compiler can be downloaded from:
  http://www.microchip.com/mplab/compilers

The MPLAB Harmony drivers and PLIBs that are needed for this project are included in the firmware package. But, if you want to explore other features of Harmony or port this project to a different version of Harmony, go to:
http://www.microchip.com/mplab/mplab-harmony

You will need a Type A Mini USB cable to provide power and communicate with the PC for bootloader or CDC commands.
Optionally, you can use one of the following debug tools to debug the LED cube code:

- **MPLAB ICD3**
  An RJ-11 to ICSP® adapter (AC164110) is required.
- **MPLAB REAL ICE™ In-Circuit Emulator**
  An RJ-11 to ICSP adapter (AC164110) is required.
- **PICkit3**
  No adapter is needed.

You will need these tools to restore the entire LED cube image if the boot loader was corrupted.

### 3.4 REPROGRAMMING THE LED CUBE

The LED cube project integrates a bootloader which enables new firmware to be downloaded via USB. This eliminates the need for the hardware development tools. The bootloader is invoked by holding down the MODE button on the LED cube board and pressing the Reset button. The LED cube firmware scans the MODE button at power up, and if it is asserted, it enters Bootloader mode.

The USB bootloader implements the HID protocol, which eliminates the need to install any custom drivers. The LED cube bootloader uses the AN1388 bootloader application on your PC to download new firmware.

**FIGURE 3-1: BOOTLOADER APPLICATION**
1. Start the boot loader application and check “Enable” in “Communication Settings>USB”.

2. Click the Connect button to connect to the LED cube. You should see the boot loader version displayed.

3. Click the Load Hex File button to load the file "LED_Cube.X.production.unified.hex" located at firmware/LED_Cube.X/dist/RevB256K_Hardware/production

4. Click the Erase-Program-Verify button. In some cases you may need to press this twice. It should walk through the Flash Erase, Programming Completed and Verification successful phases.

5. Click Run Application and the LED cube will disconnect from the bootloader application and run normally.
3.5 REPROGRAMMING THE LED CUBE WITH AN MPLAB X IDE PROGRAMMER

For this programming configuration, you will need:

• a USB cable with a series A connector on one end (for connecting to a PC) and a mini series A connector on the other end (for connecting to the LED cube board). The cube is powered by USB, so you have to have USB connected any time you are programming or debugging the cube. The maximum current is 500 mA with default firmware and LED brightness settings.

• a hardware debug/programmer tool, such as PICkit 3, MPLAB ICD 3 or MPLAB REAL ICE in-circuit emulator. These tools are assumed to have their own USB cable (for connecting to the PC) and ISCP cable (for connecting to the LED cube board).

Before you begin, ensure you have installed the MPLAB XC32 C compiler and MPLAB X IDE (see Section 3.3 “Development and Firmware Download Tools”). Then open the project in MPLAB X IDE:

• Start MPLAB X IDE.
• Select File>Open Project, <Harmony Version>/apps/LEDCube/firmware/LED_Cube.X to open the LED_Cube project.
• Build project to verify compiler installation by pressing the Build Project button:

![MPLAB X IDE](image)

• The build should be successful. If not, see the FAQ.
• Program the cube via the Make and Program Device button.

• The device should be erased. If not, see the FAQ.

Device Erased...

Programming...

Programming/Verify complete

Please see Chapter 4. “Custom Scene Creation” for details of creating your own scenes.

3.6 PROJECT CONFIGURATION

The LED cube project was originally built with MPLAB X IDE using MPLAB Harmony v1.00 and subsequently migrated to both 1.07 (bootloader) and later 1.08 (v1.00 and v1.10 firmware releases). The LED Cube v1.10 and the later ZIP file contain both the LED cube source code and all MPLAB Harmony source files from v1.08 which were used to create the project. This allows you to unzip this anywhere on your hard drive and also enables you to create new build versions without the need to download and install MPLAB Harmony. It is advised that you use this project independent of MPLAB Harmony so that you do not encounter any potential MPLAB Harmony compatibility or relative path installation issues.

If you want to port the LED cube project to a newer version of MPLAB Harmony (maybe you want to expand functionality and want to use more MPLAB Harmony features), it needs to be unzipped into the /apps folder of the MPLAB Harmony distribution. To prevent the ZIP file from overwriting files in the MPLAB Harmony distribution, make sure you disallow/reject any file overwrites.

The current example is from MPLAB Harmony v1.08. It is critical that the /src directory land here since the project uses relative paths, and if the project does not have the same relative position, MPLAB X IDE will fail to locate MPLAB Harmony library files and subsequently fail to compile. Other versions of harmony will have a different version number (vX_XX), but the relative path remains the same.
The project contains core files for basic cube functions.

```plaintext
app
  app_bt.c
  app_clock.c
  app_command.c
  app_cube.c
  app_io.c
  app_nvm.c
  app_usb.c
  fonts.c
  main.c
```

It also contains the SPI Driver for the LED1642, initialization, task and interrupt functions.

```plaintext
SPI_LED1642.c
system_config
  Rev8256K_Hardware
    framework
      system_exceptions.c
      system_init.c
      system_interrupt.c
      system_tasks.c
```

It is unlikely you will need to modify any of these files unless you plan to alter and/or extend features (such as adding commands to the command line interpreter, adding BLE support, etc).

If you want to add custom scenes, follow the instructions in Chapter 4, “Custom Scene Creation” for details about how to create custom scenes.
Chapter 4. Custom Scene Creation

4.1 INTRODUCTION

The PIC32MX270F256B LED cube project provides a simple method for users to add their own custom scenes using the LED cube and MHC framework that is provided within the project. This chapter contains an overview of the process of creating a scene, a step-by-step example of how to create a scene, and details on some key helper functions that make it easier to perform more complex visual effects with the cube. Follow these steps and have some fun creating your own scenes!

- Example Scene Overview
- Example Scene Detail
- Manipulating the Cube

4.2 EXAMPLE SCENE OVERVIEW

To show how you can create your own scenes, examples of functions and how they execute are discussed in the following sections.

4.2.1 Functions to Create for Your Scene

To create your own scene, you will create three functions. Consider an example scene called `sceneMyNewScene`, with the following three functions for that scene:

1. Constructor Function: `bool sceneMyNewSceneConstruct(void);`
   a) This function is optional.
   b) It is normally called once for the purposes of initializing variables used in the scene and setting the initial color state of the cube for the start of the scene.
   c) It can also be called multiple times if the scene setup is completed through multiple steps. It will be called repeatedly until it returns `true`.

2. Cube Color Update Function: `bool sceneMyNewSceneRun(void);`
   a) This function is mandatory.
   b) This is where you will code the details of your scene.
   c) Each time it is executed you can update the colors displayed by the cube.
   d) It is executed repeatedly at a regular time interval (dt). Every dt period of time it will be called and will update the cube colors.
   e) This function will continue to be called until a maximum time (Tmax) for your scene is reached or until this function returns `true`. A Tmax value of -1 disables the max time which will allow the scene to run continually or until it returns `true`.

3. Destructor Function: `bool sceneMyNewSceneDestroy(void);`
   a) This function is optional.
   b) It can be called once after the scene has terminated to give you a chance to make final cube adjustments before the next scene.
   c) It can also be called multiple times to allow an orderly shutdown of a scene by allowing a scene to extinguish the illuminated LEDs in sequence rather than abruptly, all at once. It will be called repeatedly until it returns `true`. 
4.2.2 What Causes Your Scene to Execute?

To run your new scene on the LED cube, you have to add it to the list of scenes that are executed by the cube. In other words, registering your scene with the cube. The cube sequentially executes the scenes in the list and after you have added yours to the list, it will automatically execute in sequence. The scene list is a structure which contains pointers to the scene functions and the time parameters (dt, Tmax) necessary for properly executing the scene.

4.3 EXAMPLE SCENE DETAIL

The following steps give a detailed example of how to successfully create a scene.

1. Open the sceneDispatch.c file and add a function prototype of your new scene name. (You will see a list of the function prototypes for the other scenes in the file.) Your scene function must return a bool (which represents the ending of the scene) and its parameters are void, as shown in this example:

   ```c
   bool sceneMyNewSceneRun(void);
   ```

2. For simple scenes you only need the scene itself. However, the LED cube framework provides the option to also create "Constructor" scenes, which run first to create an entrance effect, and "Destructor" scenes, which runs once the main scene finishes. The destructor scene enables a controlled sequence to move to the next scene. The constructor and destructor scenes are optional; you can use NULL or existing functions like cubeClear().

3. Add your new scene to the sceneFunc[] table using the following format:
   a) pointer to the scene function itself
   b) pointer to the scene constructor (use &cubeClear for simple scenes)
   c) pointer to the scene destructor (use NULL for simple scenes)
   d) text name of the scene (this is displayed with the "LS" command)
   e) time slice (dt) in milliseconds (ms) for which you want the scene to run and update some pixels
   f) maximum time limit (Tmax) in ms for which the scene can run
   Also the scene can self-terminate by returning true.

   ```c
   {&sceneMyNewSceneRun, &cubeClear, NULL, "My New Scene", 5, 10000},
   ```

4. Create a source file which matches your scene name, sceneMyNewScene.c in this example.
   a) Include sceneHelper.h.
   b) Create your scene function with a default return of false.

   ```c
   # include "sceneHelper.h"
   bool sceneMyNewSceneRun(void){
       return false;
   }
   ```

Now you can start manipulating your cube through many cube APIs.
4.4 MANIPULATING THE CUBE

There are several methods to manipulate the display on the cube. The lowest-level method of changing a pixel is to directly manipulate the structure that contains the color data displayed on the cube. In order to make manipulating the cube display easier, a set of helper functions and APIs have been created.

4.4.1 Manually Setting a Single Pixel (Not Recommended, but Informative)

It is possible to directly access the LED cube structure and adjust the red, green or blue intensity of any given LED. When the cube updates the LEDs, it pulls directly from the data in this structure to set the colors of the cube. This gives you direct access to the cube colors; but can be difficult, since you have to write three separate values each time, as shown below. It is much easier to use the helper functions and APIs to update the cube display.

```c
LED.cube.z[LEDSIZE].y[LEDSIZE].xred[LEDSIZE] = LEDMAXVALUE;
LED.cube.z[LEDSIZE].y[LEDSIZE].xgreen[LEDSIZE] = LEDMAXVALUE;
LED.cube.z[LEDSIZE].y[LEDSIZE].xblue[LEDSIZE] = LEDMAXVALUE;
```

4.4.2 Helper Functions and APIs

The following helper functions and APIs are helpful in creating new custom scenes.

4.4.2.1 SETTING A PIXEL USING HELPER FUNCTIONS

Instead of manually setting a pixel in the LED cube structure, we suggest you use the `cubeSetColor(Red, Green, Blue)` API to set the pixel color, and then set or clear the pixel of interest using `cubeSetPixel(x,y,z)`.

```c
cubeSetColor(LEDMAXVALUE, LEDMAXVALUE, LEDMAXVALUE);
cubeSetPixel(LEDXSIZE, LEDYSIZE, LEDZSIZE);
```

4.4.2.2 COLOR SELECTION

There are several ways to select the current color: you can manually use R, G and B intensity values between 0 and `LEDMAXVALUE` or you can use any of these functions for saturated, random or a specific Hue (Hue between 0 and 359.9). Once you execute these functions to generate a color, that color is stored in `LED.red`, `LED.green`, and `LED.blue`. You can use other helper functions to set pixels to the color you generated.

```c
cubeSetColorRandom();
cubeSetColorRandomSaturated();
cubeSetHue(200);
```

Setting the color using the above functions does not affect anything that is already displayed; it only affects pixels that are displayed after the color has been set.

4.4.2.3 SETTING PIXELS

Pixels are manipulated by these functions with an X, Y and Z parameter between 0 and `LEDXSIZE`, `LEDYSIZE` and `LEDZSIZE`.

```c
cubeSetPixel(LEDXSIZE, LEDYSIZE, LEDZSIZE);
cubeClearPixel(LEDXSIZE, LEDYSIZE, LEDZSIZE);
cubeMovePixel(LEDXSIZE, LEDYSIZE, LEDZSIZE, LEDXSIZE, LEDYSIZE, LEDZSIZE);
cubeCopyPixel(LEDXSIZE, LEDYSIZE, LEDZSIZE, LEDXSIZE, LEDYSIZE, LEDZSIZE);
cubeClear();
```
4.4.2.4 DRAWING LINES, NUMBERS, AND LETTERS

You can draw lines, numbers and display fonts with these functions:

```c
cubeDrawLine(LEDXSIZE, LEDYSIZE, LEDZSIZE, LEDXSIZE, LEDYSIZE, LEDZSIZE);
cubeSetFont(0, 'H');
cubeSetNumber(2, 4);
```

4.4.2.5 CHECKING PIXEL STATUS

You can check the status of any pixel via the `cubePixelLit()` function:

```c
if (cubePixelLit(LEDXSIZE, LEDYSIZE, LEDZSIZE)) {
    // do something
}
```

4.4.2.6 ADVANCED APIS (MOVE, ROTATE, ETC.)

More advanced scenes can use the `cubeMove()` and `cubeRotate()` APIs.

```c
cubeMoveBackOnePixel(0);
cubeMoveDownOnePixel(0);
cubeMoveForwardOnePixel(0);
cubeMoveLeftOnePixel(0);
cubeMoveRightOnePixel(0);
cubeMoveUpOnePixel(0);

cubeRotateOuterRingAboutX();
cubeRotateOuterRingAboutY();
cubeRotateOuterRingAboutZ();
cubeRotateOuterRingAboutZ_Left();
cubeResetSpin();
```

4.4.3 Important Scene Properties and Suggestions

These are important items to consider when creating a scene:

- **Use the `scene.state` and `scene.count` variables to maintain any state transitions within your scene.** Both the state and count variables will be cleared the first time your scene is called. Your scene, if using these variables, should access and increment them as needed to enable scene progression.

- **Your scene function must be non-blocking;** that is, it will call the appropriate cube API helper functions, but cannot spin lock or wait for some external event. A blocking function can lock up and crash various features of the cube, including the Clock, USB, or Bluetooth® communications.

- **Your scene function will be called in ms intervals defined by the Time Slice parameter you specified in `sceneFunc[]`.** Your scene is essentially a “task” which is triggered/executed on every time slice.

- **Your scene needs to return `false` if it has not yet completed, `true` if it is complete, and the scene dispatcher should start displaying the next scene.** Returning `true` will override the Maximum Time limit specified in `sceneFunc[]`.

- **Refer to the source code of existing scenes to see how scenes are created.** `sceneClockRun.c` is a good example of using the font and number APIs along with count and state variables.
Example Code

```c
bool sceneClockRun(void)
{
    cubeClear();
    cubeSetColorRandomSaturated();
    switch(scene.state)
    {
    case 0:
    {
        if (clock.hour/10 > 0)
        {
            cubeSetNumber(0, clock.hour/10);
            break;
        }
    scene.state++;
    }
    case 1:
    {
        cubeSetNumber(1, clock.hour - (10*(clock.hour/10)));
        break;
    }
    case 2:
    {
        cubeSetFont(2, ':');
        break;
    }
    case 3:
    {
        cubeSetNumber(3, clock.min/10);
        break;
    }
    case 4:
    {
        cubeSetNumber(4, clock.min - (10*(clock.min/10)));
        break;
    }
    }
    if(++scene.state >= 5)
    {
        scene.state = 0;
        if(++scene.count >= 2)
        {
            scene.count = 0;
            return true;
        } else
            return false;
    }
    return false;
}
```
Appendix A. Frequently Asked Questions

A.1 INTRODUCTION

If you encounter issues when assembling or working with the 5x5x5 RGB LED Cube kit (AC100200), please see the list below for frequently-asked questions.

A.2 FAQ LIST

Q: Do I have to solder the LEDs?
A: Yes, the LEDs need to be lead formed, trimmed and soldered to the vertical (RGB) and horizontal (Level Selection) wires. The PCB is fully assembled with all surface mount components. You need to have a soldering iron, solder and the ability to solder 0.1" (2.54 mm) spaced wires to the LEDs to complete this cube.

Q: How many solder joints are involved in assembling the cube?
A: The needed solder joints are:
125 x 4 = 500 RGB LED to wire joints
5 x 7 + 5 levels = 40 wire to wire joints
25 x 3 + 5 levels = 80 wire to PCB joints
3 IR wire to PCB joints
Total solder joints = 623

Q: Are assembly fixtures provided or do I have to print or make them myself?
A: The LED Cube kit does not come with the fixtures which are needed to lead form the LEDs and properly space the LED matrix when soldering them together. If you have access to a 3D printer, you will find both lead forming and assembly/spacing fixture files which you can print and use. If you don't have a 3D printer, you can make a lead forming fixture from a 1/8" (3.175 mm) piece of flat bar aluminum stock and the assembly and/or spacing fixture from a block of wood. We have included assembly instructions for both methods in Chapter 2. “Assembly Instructions”.

Q: Does the Cube come with the base?
A: No, the cube includes only the base PCB, RGB LEDs, wire and PIR sensor. You can print a base with a 3D printer or make one from wood or other appropriate material.

Q: How long does it take to assemble the LED cube?
A: Using the fixture described in the documentation, it takes from 7 to 10 hours of soldering time for an experienced engineer or technician.
Q: Does the LED cube have a built-in bootloader?

A: The LED cube is pre-programmed with a HID bootloader that compiles with AN1388. The AN1388 GUI is Windows® only, and does not presently support Mac OS or Linux. You can re-flash the LED cube using MPLAB X IDE and a standard development tool like PICkit 3, MPLAB ICD3 or MPLAB REAL ICE in-circuit emulator using Linux or Mac operating systems.

Q: How is the USB interface used?

A: The USB interface is used to power the cube and enumerates as a CDC class virtual communication port. You can connect to it using a terminal program and control the cube via the command line interpreter. USB communications is optional, but USB power is required.

Q: How is Bluetooth used?

A: Current firmware configures the RN4677 as a SPP Bluetooth device. As such, it should appear as a virtual COM port on a PC or Android phone. Unfortunately, iOS does not have SPP support, so you cannot connect to the LED cube via iOS. The RN4677 can optionally support BLE (which is supported with iOS) but the firmware does not support it at this time. The Bluetooth connection is used to control the cube in the same way as USB. Bluetooth communication is optional, as the cube will operate without it.

Q: Do I need to know MPLAB Harmony in order to make this?

A: No, the cube is pre-programmed with both a bootloader and a complete MHC project with many LED Scenes. You can write your own scenes and add them using MPLAB X IDE, MPLAB XC32 C Compiler, and either the USB bootloader built into the Cube or a Microchip hardware debugger/programmer. You don't need to have any MHC knowledge to write new cube scenes as there are simple cubeFunc() APIs which are used to manipulate the cube. Please review Chapter 4, “Custom Scene Creation” for scene creation instructions, examples and APIs.

Q: What coding experience is necessary?

A: The LED cube is pre-programmed with many scenes, so it can be used “as-is” without any further programming. The LED cube contains a bootloader which can be used to update the firmware using the AN1388 bootloader executable. Updates to the LED cube firmware will be posted as they are available. You can also write your own scenes using MPLAB X IDE, MPLAB XC32 C Compiler, and either the bootloader or a Microchip hardware debugger/programmer. This is intended for embedded professionals.
Appendix B. Command Line Interpreter

B.1 INTRODUCTION

The LED cube implements a USB CDC and Bluetooth SPP command line interpreter which can be used to select scenes, set the clock, or manually manipulate pixels within the cube. You can use any terminal program such as CoolTerm to connect to the cube by selecting the LED cube's COM port.

- LED Cube PCD to PC USB Connection
- Connecting to the Terminal Program

For an example of use, see Appendix C. "PCB Test".

FIGURE B-1: INTERPRETER BLOCK DIAGRAM
B.2 LED CUBE PCD TO PC USB CONNECTION

Begin by connecting the LED cube PCB to the PC via the Type A Mini USB cable. When the LED cube is properly enumerated, it shows up as a USB serial port with a COMxx designator (Figure B-2).

**Note:** Some Windows operating systems will need a .INF file the first time you plug your cube into your PC. You will find a `mchpcdc.inf` file as part of the LED cube firmware and documentation download. Please point to that .INF file if Windows asks for a driver.

**FIGURE B-2:** DEVICE MANAGER COM PORT

![Device Manager COM Port](image-url)
B.3 CONNECTING TO THE TERMINAL PROGRAM

Connect to this COM port with your Terminal program (e.g., CoolTerm). Hold down the "H" key and a carriage return and you should see the help menu (Figure B-3).

FIGURE B-3: TERMINAL HELP MENU
You can list all active scenes via the LS command (Figure B-4).

**FIGURE B-4: TERMINAL LS COMMAND**

For example, you can set the clock via T1 10 23, which sets the clock to 10:23. If you have an Android phone, the same interface is available by installing the "Bluetooth Terminal" application from the Google Play store. This terminal program uses the "." Delimiter for commands, so you need to append a "." to any command before sending it.
Appendix C. PCB Test

C.1 INTRODUCTION

The LED Cube PCB Manufacturing Test Procedure, v1.2, is documented in the following sections.

• Equipment and Software Requirements
• Hardware Connections
• Program PCB Device using MPLAB IPE
• Test Board using CoolTerm
• Error Conditions

C.2 EQUIPMENT AND SOFTWARE REQUIREMENTS

Test hardware and software used:

• PC running Windows 7, 8.1 or 10 with a minimum of 2 GB RAM (4GB or more preferred)
• Mini USB - Type A cable
• PICkit 3 or MPLAB ICD 3 with a AC164110 RJ11-to-ICSP Adapter
• MPLAB X IDE (includes MPLAB IPE) and CoolTerm software installation:
  - http://www.microchip.com/mplab/mplab-x-ide
  - http://freeware.the-meiers.org
C.3 HARDWARE CONNECTIONS

Connect the PCB, PC and debug/programmer tools:
- Plug USB cable into PC and Mini USB connector on LED cube board.
- Plug PICkit 3 or MPLAB ICD 3 into PC and ICSP Connector into LED cube board.

FIGURE C-1: MPLAB ICD3 OR PICKIT 3 CONNECTIONS
C.4 PROGRAM PCB DEVICE USING MPLAB IPE

Follow the steps below to program the LED cube board (PCB) with firmware (Hex file) using MPLAB IPE.

1. Launch MPLAB IPE.

FIGURE C-2: MPLAB IPE ICON AND GUI

2. Set up MPLAB IPE and connect to the LED cube board:
   a) Family: Select 32-bit MCUs (PIC32).
   b) Device: Select PIC32MX270F256B.
   c) Tool: Select 'PICkit3' or 'ICD 3'.
   d) Click Apply.
   e) Click Connect. You should see the message below when it is connected.

```
2016-04-20T09:37:38-04:00: Completed loading IPE.

Connecting to MPLAB ICD 3...
Currently loaded firmware on ICD 3
Firmware Suite Version........01.41.06
Firmware type.................PIC32MX

Target voltage detected
Target device PIC32MX270F256B found.
Device ID Revision = A1
```
3. Load and program the Hex File:
   a) Click **Browse** and Load “LED_Cube.X.production.hex”

   ![Image of file browser with selected file](file-browser-image.png)

   Once loaded you will see this message in the output window:

   ```
   2015-05-20T14:23:56-0400: Loading hex file. Please wait...
   Loading code from
   C:\Microchip\harmony\v1_07\apps\LED_Cube\firmware\LED_Cube.Xdist\Rev6255K_Hardware\production\LED_Cube_V100_051316.hex...
   ```

   b) Click **Program**. When done, you should see:

   ```
   Programming...
   Programming/Verify complete
   2016-05-20T09:53:37-0400- Programming complete
   Pass Count: 1
   ```

   c) Unplug the hardware tool programming connection from LED cube board.

   **Note:** If programming multiple boards, you only need to click the **Connect** and **Program** buttons after hardware setup to program the board. It is not necessary to re-select the device or re-load the HEX file during each programming event.
C.5 TEST BOARD USING COOLTERM

Perform the steps below to test the LCD Cube board (PCB) using CoolTerm.

1. Launch CoolTerm.

2. Invoke “Device Manager” and locate the LED cube board. It will show up as a USB Serial Port and when you click on it, you will see detailed information.

Note: If testing multiple boards on the same PC, you only need to identify the COM port once since the COM port number will remain the same for all devices as long as you use the same PC and USB port.
3. In CoolTerm, click the **Options** button and select the COM port that corresponds to the LEDCube board.

4. Click the **Connect** button on CoolTerm.

5. Press the **<Mode>** button on LED cube board.

6. Observe Test Mode results.

```
See www.microchip.com/LEDCube for code updates...

Light   Passed, Level Min 15, Max 500, Reading = 100, LED Bias = -28
Bluetooth Passed, State Min 13, State = 13

POT = 729
Time = 01:08:05......047
```

Light and Bluetooth should both show “Passed”. If Light fails, check U6 (light sensor), R15 and U8 pin 26. If Bluetooth fails, check U9 (RN4677) and U8 pins 2,7,11,12,14,18.

7. Click **Disconnect** in CoolTerm.

8. Press **<Reset>** button on LED cube board.

9. Press **<Mode>** button on LED cube board.
10. Flip the board over so side 2 (bottom) is exposed. Push the contacts of an LED tester onto the left-most row, 2nd down from top LED position shown.

11. Insert a wire into each of the 5 level drive positions and verify that the LED sequences from red to green to blue. If you don’t see the RGB pattern, try pressing the Mode button and re-test. All 5 level positions need to be tested.
C.6 ERROR CONDITIONS

If the USB won't connect:
- check Crystal, USB connection, PIC32 MCU soldering and orientation
- check device manager to see if device enumerates as COM port

If BT fails:
- check RN4677 and PIC32 MCU soldering and orientation

If LIGHT fails:
- check Light sensor and R15

If LED tester fails:
- check U1, U2, U3, U4 and U5 soldering and orientation
- check T1 through T10 soldering and orientation
- check resistors associated with T1 through T10
Appendix D. Drawings & Schematics

D.1 INTRODUCTION

This chapter provides hardware details about the LED cube and associated PCB.

D.2 PCB SCHEMATIC AND DRAWINGS

FIGURE D-1: PCB ASSEMBLY DRAWING
FIGURE D-2: PIC32 MCU OPERATIONAL DRAWING

PIC32 Block Diagram

- PIC32MX270F256B
- USB
- DMA
- Timer
- DMA
- SPI
- Latch/OC
- Timer
- PWM/OC
- LED Matrix
- LED1642 16-Channel 12-bit PWM
- Bluetooth
- ADC
- UART
- GPIO
- GPIO
- Ambient Light
- POT
- Mode
- PIR
FIGURE D-3: PCB SCHEMATIC - PART 1
FIGURE D-4: PCB SCHEMATIC - PART 2
D.3 LED CUBE SCHEMATIC AND DIAGRAM

Figure D-5 is one slice of the matrix (it is replicated 5x in the actual cube). You will see five levels, but only one level is turned on at a time through a P-Channel FET (SiA445EDJ). The RGB for each vertical LED arrangement is PWMed individually (LED1642), so there are 15 RGB connections per slice, for a total of 75 PWM drivers. Since only one level is asserted at one time, each LED is individually dimmable.

FIGURE D-5: LED CUBE SCHEMATIC
Figure D-6 shows the LED cube planes and level selections which make up the 3-dimensional drive implementation. The SPI port operates as a 16-bit shift register where the Latch line latches the current shifted value into one 16 PWM duty cycle registers or one of several command registers within the LED1642 PWM driver. This PWM driver needs an external clock, so a PWM from the PIC MCU is used to generate one. The Level drive is handled through P-FETs. By combining the Level and PWM drive into the same LED1642, the multiplexing is automated and synchronized with the LED PWM period.
# Index

**A**
- AC100200 ............................................................. 5
- Advanced APIs (Move, Rotate, etc.) .................. 46
- Assembly File Locations .................................. 7
- Assembly Instructions ........................................ 7

**B**
- Bluetooth .......................................................... 51

**C**
- Checking Pixel Status ........................................ 46
- Color Selection .................................................. 45
- Command Line Interpreter ................................. 51
- CoolTerm .......................................................... 51, 55, 59
- Custom Scene Creation ...................................... 43

**D**
- Design Files ....................................................... 6
- Drawing Lines, Numbers, and Letters ................. 46

**E**
- Example Scene Detail ......................................... 44
- Example Scene Overview ................................... 43

**F**
- Functions to Create for Your Scene .................... 43

**H**
- Helper Functions and APIs .................................. 45

**I**
- Important Scene Properties and Suggestions ....... 46

**L**
- LED Cube Block Diagram .................................... 68
- LED Cube Features ............................................. 6
- LED Cube Kit Contents ...................................... 6
- LED Cube Schematic .......................................... 67
- LED Lead Forming Using Fixture
  - 3D Printed ...................................................... 8
  - Aluminum Bar ................................................ 11
- LED Matrix Wire Forming and Cutting ................. 14

**M**
- Manipulating the Cube ..................................... 45
- Manually Setting a Single Pixel
  - (Not Recommended but Informative) ................. 45
- Matrix Holder Soldering Fixture
  - 3D Printed .................................................... 16
  - Word or Cardboard ......................................... 22
- MPLAB Harmony
  - Features Used ............................................... 6
  - Project Configuration .................................... 41

**O**
- Overview .......................................................... 5

**P**
- PCB Assembly Drawing ..................................... 63, 64
- PCB Schematics ............................................... 65
- PCB Test .......................................................... 55
- Programming ................................................... 37
- Programming File Locations ............................... 37

**R**
- Reprogramming the LED Cube
  - with an MPLAB X IDE Programmer .................. 40
  - with the USB Bootloader ............................... 38

**S**
- Setting A Pixel Using Helper Functions .............. 45
- Setting Pixels .................................................. 45
- Soldering the Planes to the PCB ....................... 27

**W**
- What Causes Your Scene to Execute? ................. 44