

Interfacing the Android Smart Phone with the Explorer 16 and PIC24FJ256GB110 Processor

Stephen Hilton

March 28, 2012

This application note has been created to explain how to interface the Android Smart Phone with the Explorer 16 using the PIC24FJ256GB110 processor module. It starts off with how to download and compile the appropriate files for both the PIC24F and the Android phone to get Microchip's Basic Android Accessory Demo working. It then gives a walkthrough of how the demo works and briefly gives an example of steps one may take to change the application.

1. Key Words

IDE, Integrated Development Environment, Android, Smart Phone, PIC24FJ256GB110, Explorer 16, SDK, Software Development Kit, Basic Demo, ICD3, USB PICTail, GUI, Graphical User Interface

2. Objectives

The purpose of this document is to walk you through downloading the correct Microchip Application library, compiling the source code for the Android application provided by Microchip, and downloading it to your phone. It will also show you how to build the given project onto the PIC24FJ256GB110 chip located on the Explorer 16 development board, give a quick walkthrough of the demo and give a brief example of how to modify the application to suit one's needs.

3. Assumptions

This application note assumes that *Eclipse IDE* has already been downloaded on your computer with the correct *Android SDK* components and the *ADT plugin* for a Windows operating system. It also assumes that you are using a smartphone with *Android v2.3.x*, though simple adjustments can be made if a higher version of the Android OS is being used.

4. Downloading the Microchip App Library

To get started, first download the microchip application libraries for Windows from <http://www.microchip.com/mal>. Download version 2012-02-15 or later. On the website you will find a selection of links as seen in **Figure 1**. If you are using Linux or Mac OS X, select the appropriate library to download.

Microchip Application Libraries	
Downloads	
Microchip Application Libraries v2012-02-15 Windows	
Microchip Application Libraries v2012-02-15 Linux	
Microchip Application Libraries v2012-02-15 Mac OS X	

Figure 1: Microchip Application Libraries

5. Loading the PIM on the Explorer 16

Once downloaded, find *Microchip Solutions v2012-02-15\Android Accessories\Basic Communication Demo - OpenAccessory Framework\Firmware* and choose the Basic Android Accessory Demo for the PIC24FJ256GB110 PIM using MPLAB IDE v8.83.

Make sure the correct PIM installed in your Explorer 16 and that the USB PICtail+ is attached.

To build and program the chip go to ***Project->Build All*** then ***Programmer->Program***. This will conclude the programming of the PIC24FJ256GB110 PIM.

6. Compiling the Android Phone Source Code

Now to compile the source code for the Android phone. Open up ***Eclipse IDE*** and create a new Android Project by right clicking in the ***Package Explorer*** sidebar go to ***New->Android Project***. This is shown below in **Figure 2**.

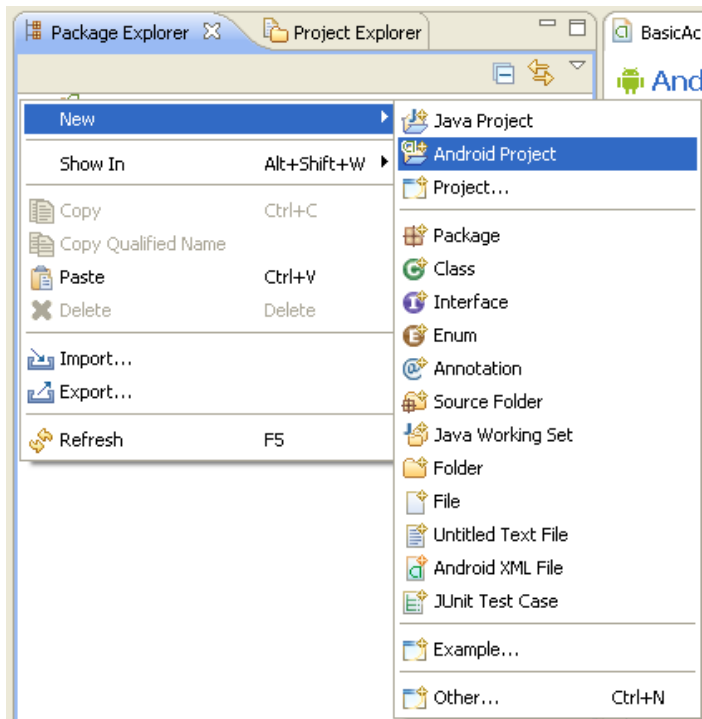


Figure 2: Creating a New Android Project

The window shown in **Figure 3** will pop up. Give the project a name and select to “**Create project from existing source**”. Browse to *Microchip Solutions v2012-02-15\Android Accessories\Basic Communication Demo - OpenAccessory Framework\App Source Code\v2.3.x*.

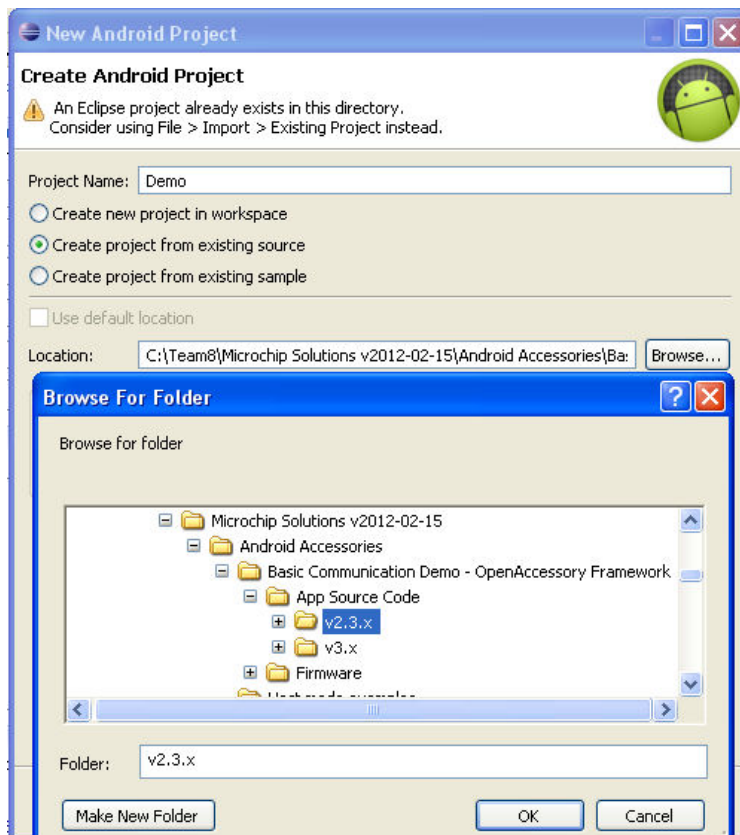


Figure 3: Creating Project from Downloaded Source Files

Once you have the correct file path, choose an SDK to target. In this case, choose the *Google APIs* located under the *Android 2.3.3* as shown in **Figure 4**. Click *Next*. The *Application Name* and *Package Name* are already filled out for you as “Basic Accessory Demo” and “com.microchip.android.BasicAccessoryDemo” respectively. Click *Finish*.

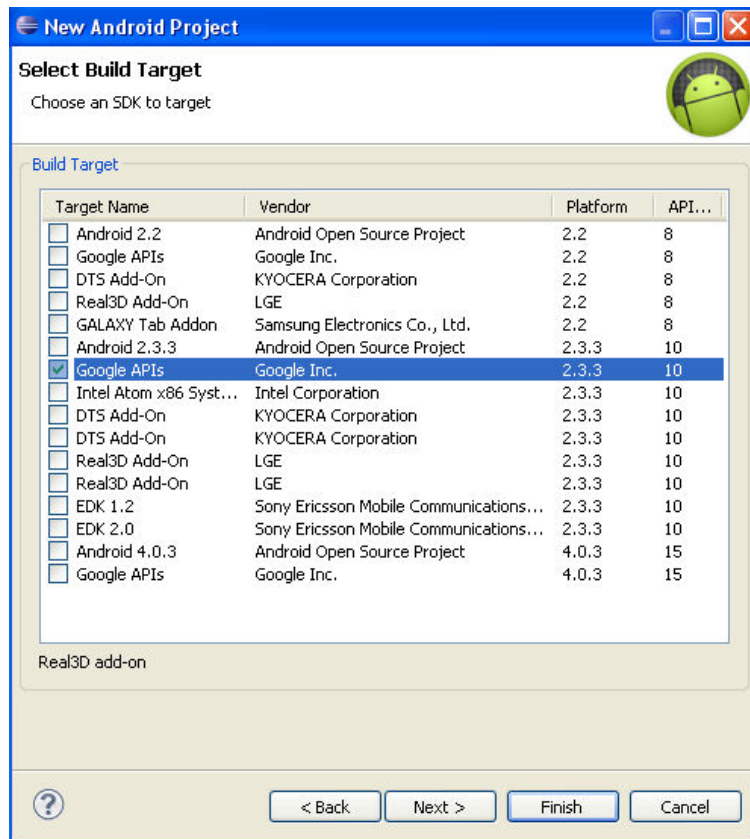


Figure 4: Selecting the Build Target

To download the application you've just compiled, *right click the Project -> Run As -> Android Application*. The *Android Device Chooser* will pop up as in **Figure 5**. Make sure your phone is on and connected to the computer. Select “*Choose a running Android Device*” and select your phone. Once downloaded on to your phone you can select your newly downloaded app and move it to the *Home Screen* to access easily.

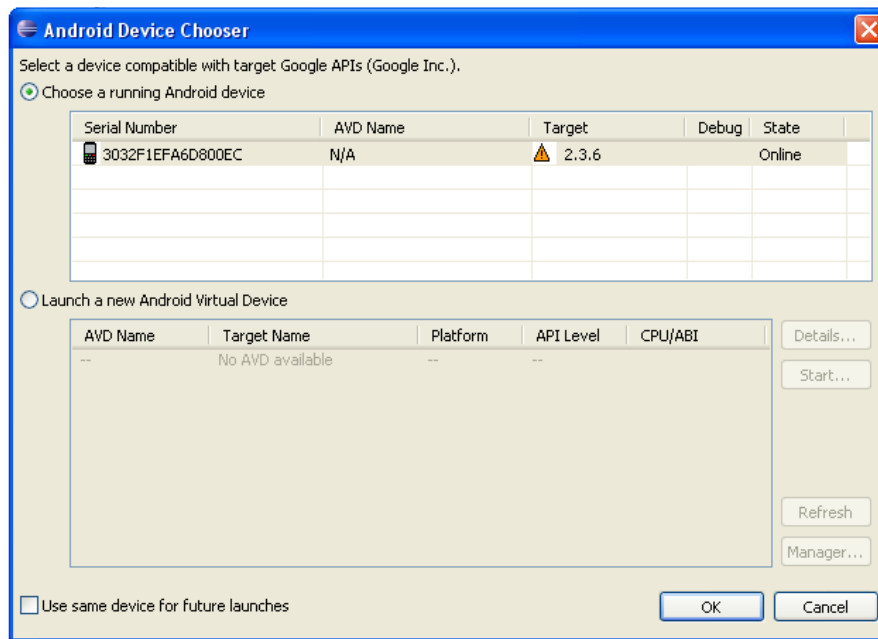


Figure 5: Android Device Chooser

7. Overview of the Application Demo

The application that has just been downloaded has several features. A picture of the demo can be seen below in **Figure 6**. The top green boxes are the LED controls. Each green button corresponds to the LED of the same relative location on the Explorer 16 Board. When the button is pressed, the button indicates that it has been pressed by lighting up and the corresponding LED on the Explorer 16 board turns on. Below the LED Controls on the smartphone are the statuses of each button on the Explorer 16. When each button is pressed the corresponding text on the phone indicates “Pressed.” and turns red. The text reverts to “Not Pressed.” when the button is let go. Finally, the potentiometer on the Explorer 16 board can be turned and the voltage level will be reflected on the smart phone.

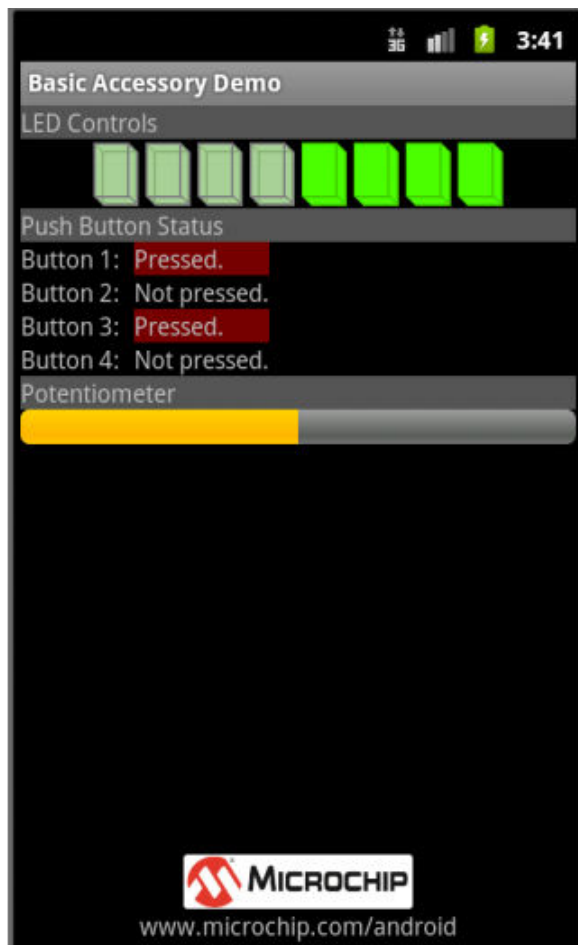


Figure 6: Android Basic Accessory Demo [1]

8. Understanding the Code

Now that you have set up a working example of an Android Application interfacing with the Explorer 16 board, the existing code can be modified to fit more specific applications. On the next page (**Figure 7**) is the main function explained in a pseudo-code/block diagram hybrid. As you can see from the block diagram, *main()* is constantly checking the LED buttons and updating on the board as well as checking the buttons and potentiometer on the board and updating the corresponding values on the Android app.

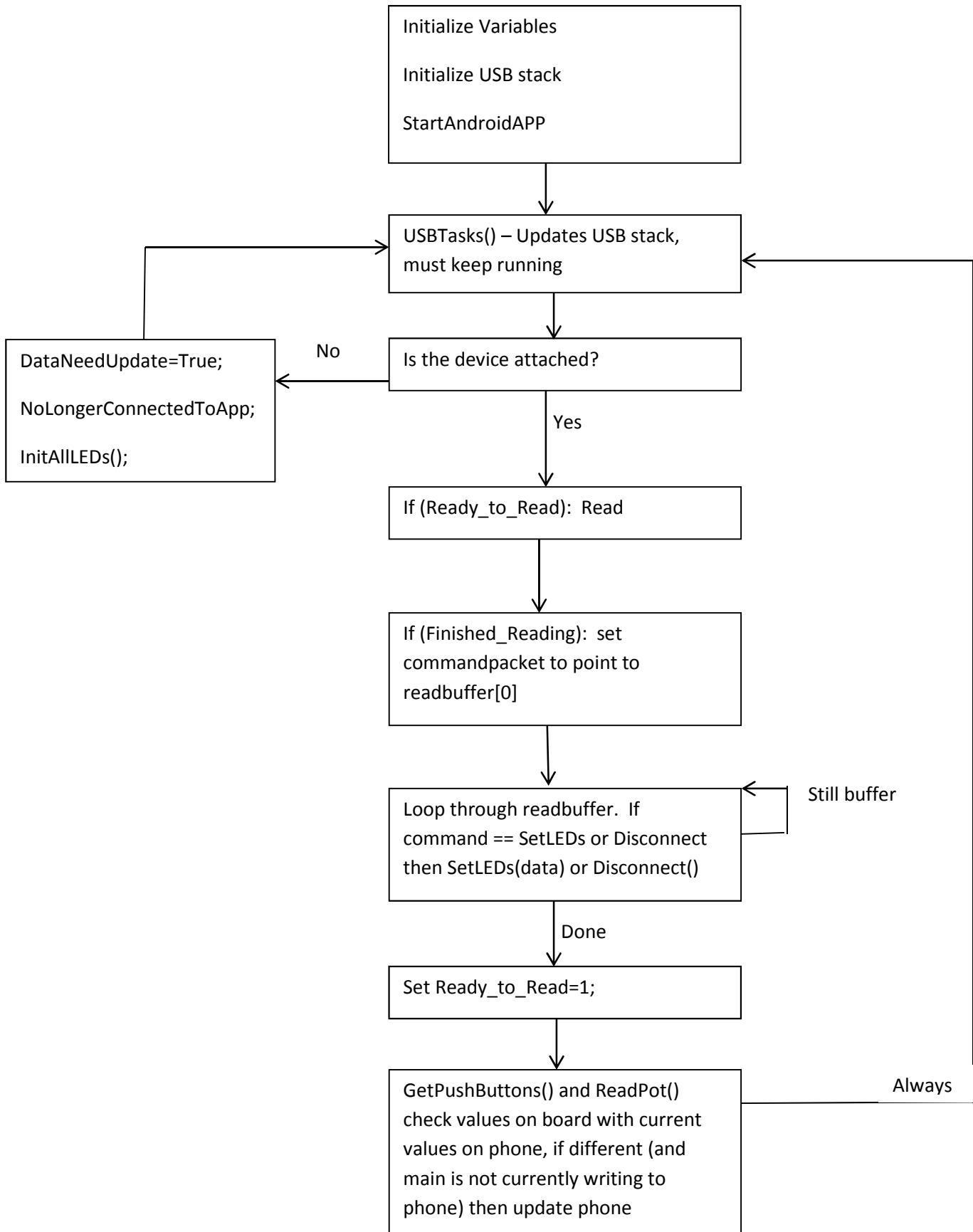


Figure 7: Main Program

9. Modifying the Code

This section will take a simple set of changes to the code and show how to execute them. This is to familiarize the reader with the process of taking the code, understand what is happening and make the code useful for their own purpose.

In this example we will turn on each LED one at a time when we go through the main while loop. Once all of the LEDs are on (i.e. the while loop has been executed 8 times) then all of the LEDs clear and the process repeats itself. In this example the LED buttons are not needed, the LEDs are instead controlled by iterations through the while loop.

The other change that we will make is to have two buttons. Pressing a button will display a specific message. Button1 displays "Harmful Chemical" and Button2 displays "Maintenance Needed".

9.1 The LEDs

For the LEDs, the phone no longer needs to be checked to see if a change has been made. This section of code can be eliminated from *main()* and the LED Buttons can be eliminated from the project folders in the *Eclipse IDE*. We will want to create an INT ledChecker initialized as zero at the top where all of the initializations are made (see **Figure 7**). This will increment at the beginning of the main while loop. At the end of the while loop will be a function call to SetLEDs(ledChecker), shown in **Figure 8**.

```
static void SetLEDs(INT ledChecker)
{
    Switch(ledChecker%8)
    {
        case 0:
            initAllLEDs(); //Special case - clear all
            LED0_On();
            break;
        case 1:
            LED1_On();
            break;
        case 2:
            LED2_On();
            break;
        //... Do up to case 7
    }
}
```

Figure 8: SetLEDs(ledChecker)

9.2 The Buttons

To make the necessary modifications take the following steps:

1. In *Eclipse IDE* locate the *Project Explorer*. Open the *layout folder* found in the *res* folder. Open *main.xml* and delete *Button3* and *Button4* on the *graphical interface*. As you did above in the **section 9.1** with the LEDs delete all references to *Button3* and *Button4* in the code.
2. Open the *values folder* found in the *res folder*. You will see a list of *Android Resources Elements*. Open *strings.xml*. If you haven't already done so, delete *Button3* and *Button4*.
3. Create the two strings *chemical_warning* and *maintenance*. The creation of *maintenance* is shown in **Figure 10**. Go to *Add->String* then enter a *Name* and *Value*.

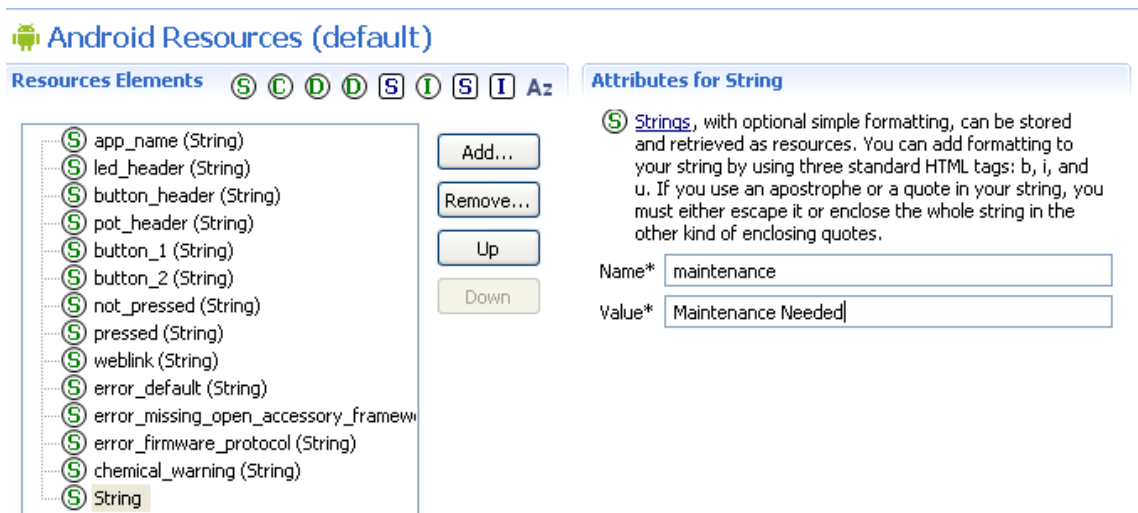


Figure 10: Creation of the String “maintenance”

4. After these are created be sure to refresh the Android Project Folder in the sidebar. This will create the needed declarations of *maintenance* and *chemical_warning* in the string class of *R.java*.

5. Open *BasicAccessoryDemo.java* located in the *src* folder. Change the *updateButton* function as shown in **Figure 11**.

```
private void updateButton(int id, boolean pressed) {
    TextView textViewToUpdate;
    LinearLayout layoutToUpdate;

    textViewToUpdate = (TextView)findViewById(id);
    layoutToUpdate = (LinearLayout)textViewToUpdate.getParent();

    if(pressed)
    {
        if(id==1)
        {
            textViewToUpdate.setText(R.string.chemical_warning);
            layoutToUpdate.setBackgroundResource(R.color.button_pressed);
        }
        if(id==2)
        {
            textViewToUpdate.setText(R.string.maintenance);
            layoutToUpdate.setBackgroundResource(R.color.button_pressed);
        }
    } else {
        textViewToUpdate.setText(R.string.not_pressed);
        layoutToUpdate.setBackgroundResource(0);
    }
}
```

Figure 11: Modifying updateButton in BasicAccessoryDemo.java

6. Compile and download onto the *Android Smartphone* by *right clicking Project -> Run As -> Android Application*.

The application now shows your message when you press the appropriate button. The *Explorer 16* also signifies to the user every time the while loop is iterated through in *main()*.

10. Results

In this application note, it has been explained how to download the appropriate Android source code as well as the PIC24F PIM code to work with the Explorer 16 from the Microchip Application Libraries. It goes through compiling the code on each, the PIC24F PIM as well as the Android smart phone. An overview of how the demo works is given as well as an explanation of the PIC24F PIM code, specifically *main()*. As an exercise to get more familiar with the code, there is then a small walkthrough of how to make changes to the code.

References

- [1] <http://ww1.microchip.com/downloads/en/DeviceDoc/Android%20Library%20Help.pdf>