Getting Started with DAQFlex for Android™

Software Framework for Developing Apps for Android Tablets

Introducing DAQFlex for Android

DAQFlex for Android from Measurement Computing (MCC) is a software framework for developing data acquisition applications, or *apps*, to run on Android-based tablets.

DAQFlex for Android greatly simplifies developing drivers and apps targeted for Android tablets. The DAQFlex for Android framework consists of the DAQFlex API, device driver and the DAQ device message engine. The message engine parses and converts the DAQFlex message-based command set, simplifying the transfer of the DAQFlex instructions that control the DAQ device and process data.

A DAQFlex for Android program sends DAQFlex methods to the driver. The driver sends the encapsulated messages to the DAQ device. The DAQ device interprets the message using the message engine, and sets its corresponding attributes using the DAQ engine. The DAQ device then returns the requested data to the DAQFlex driver, which returns the data to the program.

DAQFlex for Android includes example programs that give Android developers a head-start in creating mobile data acquisition apps that perform common data acquisition operations.

Computer and Tablet Hardware Requirements

- A computer running Windows® 8/7/Vista®/XP (SP 2 or later)
- A tablet running Android 3.1 or later and with USB Host mode support
  (visit www.mccdaq.com/support/android-apk.aspx for a list of MCC-tested tablets)
- A supported MCC DAQFlex device with the latest device firmware installed
  (visit www.mccdaq.com/solutions/DAQFlex-Solutions.aspx for a list of DAQFlex-supported devices)

  Visit www.mccdaq.com/firmware.aspx to see if you need to update the firmware on your MCC DAQ device. If no firmware is listed at that page, there are no firmware updates available for your DAQ device.

Computer Software Requirements

- Android SDK: Refer to developer.android.com/sdk/index.html to download and install the Android SDK on your computer.
- DAQFlex for Android API zip file (`daqflex_android.zip`): Visit www.mccdaq.com/android to download this file to your computer.
- Eclipse IDE: Visit www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2 to download and install this software on your computer.

Importing DAQFlex for Android into Eclipse

Complete the following steps to run in Eclipse on your computer, create a new Android project, and import the DAQFlex project into Eclipse:

1. On your computer, run Eclipse and select **Window»Perspective»Java**.
2. Right-click in the **Package Explorer pane** and select **Import**.
3. Expand the **General** folder, select **Existing Projects into Workspace** and click **Next**.
4. Choose the **Select archive file** option, browse for the DAQFlex for Android API zip file (`daqflex_android.zip`), and click **Finish**.

   The DAQFlex project is required for all applications that use the DAQFlex API. All of the other projects are usage examples.

   Each of the DAQFlex for Android projects is listed in the **Package Explorer** pane (refer to Figure 1). The function of each project is explained below.

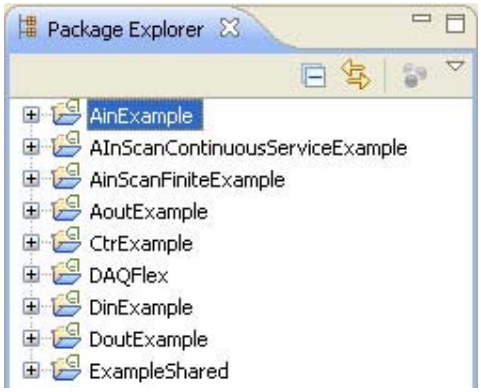

Figure 1. Package Explorer listing DAQFlex for Android example projects

AinExample	Enter a channel number and press the **Update Value** button to display the value acquired from the channel.
AInScanContinuousServiceExample	Enter a channel range (**Low Channel** and **High Channel**) and a sample rate, and press the **Run Scan** button. Application continuously displays acquired values from each channel. Scan runs until the user presses the **Stop Scan** button.
AinScanFiniteEx	Enter a channel range (**Low Channel** and **High Channel**), a sample rate, the number of samples to acquire, and press the **Run Scan** button. Application displays the samples acquired from each channel. Scan stops when the specified number of samples are displayed for each channel (or until the user presses the **Stop Scan** button).
AoutExample	Enter a channel number and a value and press the **Update Value** button. Specified analog channel is updated with the value.
CtrExample	Enter a channel number and press the **Start** button to start counting. User can press the **Update Value** or **Stop** button to display the current value. Users can also enter a number in the **Load Value** textbox and press the **Load** button to set the starting counter number.
DAQFlex	Core project that is required for all usage examples.
DinExample	Enter a channel number and press the **Update Value** button to display the value acquired from the digital channel.
DoutExample	Enter a channel number and a value and press the **Update Value** button. Specified digital channel is updated with the value.
ExampleShared	Project used by each example project that contains the UI control for the device selector drop-down and a permissions button.

Running a DAQFlex for Android Example Project

To compile a project on your computer and run the project on the Android tablet, select the project in the **Package Explorer** pane in Eclipse, and then select **Run»Run As»Android Application**.

The project automatically compiles and runs on the tablet. The DAQFlex project also compiles automatically with each example project.

Only one project can run on the Android tablet at a time

Each time you compile and run a DAQFlex for Android project, it replaces the existing project on the tablet.

To run another example project, select the project in the **Package Explorer** pane in Eclipse, and then select **Run»Run As»Android Application**.

Creating a DAQFlex for Android Project

To create your own DAQFlex for Android project on your computer using the **ExampleShared** project, complete the following steps.

1. Click to expand the **ExampleShared** project in the **Package Explorer** pane, and double-click on **AndroidManifest.xml** to open that file
2. Add the following `action` and `meta-data` section to the **AndroidManifest.xml** file (refer to Figure 2), and save the file:

```
<activity>
   <intent-filter>
      <action Android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
   </intent-filter>
   <meta-data
         android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
         android:resource="@xml/device_filter" />
</activity>
```

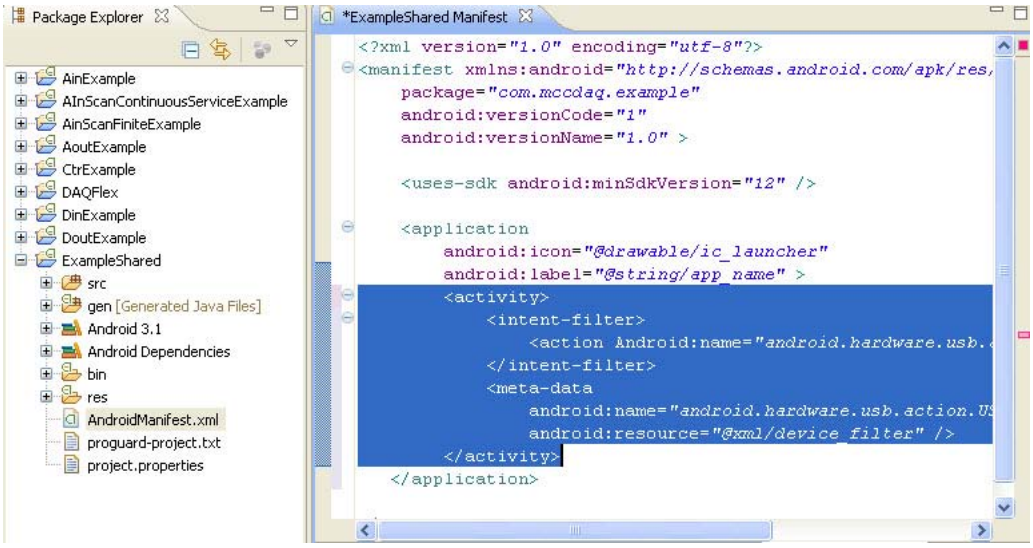

Figure 2. ExampleShared AndroidManifest.xml with added action and meta-data section

3. From one of the other example projects (except the DAQFlex project), right-click on the `res/xml/device-filter.xml` folder and select **Copy**.
4. Right-click on the `res` folder in the **ExampleShared** project and select **Paste** from the context menu.

This folder contains the file `device_filter.xml`, which the Android device uses to determine which USB devices can be used by the application. An example file provides support for the USB-7202 and USB-7204 is shown in Figure 3

Figure 3. `device_filter.xml` example projects

You can add any number of devices to this file to run with the software. The product IDs for each DAQFlex-supported device are listed below.

DAQFlex device	Product ID
USB-1208FS-Plus	232
USB-1408FS-Plus	233
USB-1608FS-Plus	234
USB-7202	242
USB-7204	240
USB-2001-TC	249
USB-2408	253
USB-2408-2AO	254
USB-1608G	272
USB-1608GX	273
USB-1608GX-2AO	274
USB-201	275
USB-204	276

Guidelines for developing a DAQFlex for Android project

Finding a DAQDevice

Call the `DeviceManager` `getDevices()` method to return a list of available `DaqDevices`. This list can then be directly passed to an `ArrayAdapter` object for use in a spinner. Unlike the desktop version of DAQFlex, the device serial number and ID is not available when running DAQFlex on an Android tablet. You must grant permission for the device and open the device before requesting this information from the hardware.

Acquiring Permission

The Android API requires permission for every USB device. To acquire permission for the USB device, call the `DaqDevice` `requestPermission()` method to open the Android USB **Permission** dialog box. The resulting broadcast is handled in the standard way for USB devices on Android.

Refer to the example in the `DeviceSelector` class of the **ExampleShared** project.

Managing the Lifecycle

After the device has permission, but before you send any commands to the `DaqDevice`, call the `open()` method to prepare the device to receive commands.

During the normal Android activity lifecycle, the activity may be destroyed any time after the `onPause()` callback. If you created the `DaqDevice` object inside of the activity, the `DaqDevice` object is also destroyed. Make sure to call the `close()` method during the activity `onPause()` method.

Managing the activity lifecycle is particularly important when running scan operations. Because the `DaqDevice` object may get destroyed by the activity, stop any active scan operations before calling the `DaqDevice close()` method.

If a scan must continue while the application is paused, start an Android Service to hold the `DaqDevice` object and to keep it open in the background. Failure to do so will result in a buffer overrun and data loss.

Sending a Message

After the device has permission and is opened, you can start sending messages to the device. Call the `DAQDevice sendMessage()` method. The `sendMessage()` method accepts a string command to send to the device.

For documentation on the DAQFlex command format, refer to the *DAQFlex Software Help* for the desktop API at www.mccdaq.com/pdfs/manuals/DAQFlex Software.pdf and the *Supported Messages* section in this document.

Reading Analog Input Scan Data

Tablet CPU limitations affect the sampling rate

Analog input scans do not run at full speed for all devices due to tablet CPU limitations. Using minimal display code on a low-end tablet, a maximum of 150 kS/s aggregate was achieved using this API with USB-1608G Series devices.

After sending an `AISCAN:START` message, scan data is read with the `readScanData()` method. Data is returned as a `double[][]`data type, with the channel number as the first dimension and the sample number as the second. This method is overloaded so that if no arguments are passed to the method, the method returns all available data. Passing the number of samples and timeout only returns that number of samples.

Supported Messages

DAQFlex for Android supports the following messages. Refer to the *DAQFlex Software Help* for the desktop API at www.mccdaq.com/pdfs/manuals/DAQFlex Software.pdf for more detailed information on these messages.

If a message from subsequent DAQFlex releases is not in the following table, or if a supported message returns an unknown status, that message may or may not function correctly.

Component	Properties
DEV	FLASHLED, FPGACFG, FPGAV, FWV, ID, MFGCAL, MFGSER, RESET, STATUS/ISO, TEMP
AI	CAL, CHMODE, CJC, DATARATE, OFFSET, RANGE, RES, SCALE, SENSOR, SLOPE, STATUS, VALUE*
AISCAN*	BURSTMODE, CAL, COUNT, HIGHCHAN, LOWCHAN, RANGE, RATE, RESET, SAMPLES, SCALE, START, STATUS, STOP, XFRMODE
AO	OFFSET, RANGE, REG, RES, SCALE, SLOPE, UPDATE, VALUE
CTR	START, STOP, VALUE
DIO	DIR, LATCH, VALUE
TMR	DELAY, DUTYCYCLE, IDLESTATE, PERIOD, PULSECOUNT, START, STOP

*Temperature channels are not currently supported during an `AISCAN`. Temperature channels can be read with the single-point `AI:VALUE` message.

Where to Find More Information

Additional information about DAQFlex software is available on our website at www.mccdaq.com. You can also contact Measurement Computing Corporation by phone, fax, or email with specific questions.

- Knowledgebase: kb.mccdaq.com
- Phone: 508-946-5100 and follow the instructions for reaching Tech Support.
- Fax: 508-946-9500 to the attention of Tech Support
- Email: techsupport@mccdaq.com