Universal Library for Android

Android DAQ Programming Library

Features

- Software API communicates with select Measurement Computing DAQ devices over the Android™ 3.1 platform (API level 12) and later
- Provides similar high-level DAQ functions as the Universal Library for Windows®
- Ideal for developing apps for use on Android-based tablets and phones
- Supports Android project development on the following platforms:
  – Windows 10/8/7/Vista®/XP, 32- and 64-bit
  – Linux®*
  – Mac®
- Supports app deployment to devices running Android 3.1 and later
- Develop and deploy to Android devices using integrated development environments (IDEs) such as Eclipse or Android Studio
- Includes classes that communicate with analog I/O, digital I/O, counter I/O, and timer I/O device subsystems
- Includes example projects to help users get up and running quickly
- Some examples are also available as demo apps from Google Play™

Overview

The Universal Library (UL) for Android is a data acquisition API for developing applications that run on the Android operating system.

The UL for Android API installs on supported Windows platforms for users to develop and deploy applications to Android devices.

A download version is also available for Linux and Mac users.

Universal Library for Android is ideal for developing DAQ apps for Android-based devices such as tablets, phones, and mini-PCs.

Easy-to-Learn for Universal Library Users

The UL for Android provides similar high-level functions as the UL for Windows for the common DAQ device I/O operations – such as `aIn()`, `aInScan()`, `dIn()`, `cIn()`, and so on – so that experienced UL programmers can adapt quickly to this API.

InstaCal™ Not Required

One significant difference found in the UL for Android is that it does not require InstaCal to install a supported MCC device. UL for Android introduces the `DaqDeviceManager` class, which enables users to programmatically detect paired or attached DAQ devices and create a device object through which they can access the I/O subsystems available on the device.

Classes to Access and Control DAQ Devices and Subsystems

UL for Android includes classes that allow users to access supported DAQ devices and their functional subsystems.

The main device and subsystem classes are explained below.

- `DaqDeviceManager` – Use this class to detect and create DAQ devices.
- `DaqDevice` – Use this class to access a DAQ device. This class contains methods to access the device I/O subsystems, identifying information, and configurations.
- `AiDevice` – Use this class to access an analog input (AI) subsystem on a DAQ device. This class also contains analog input methods such as `aIn()` and `aInScan()`, along with methods to access AI subsystem information and configuration.

* Linux® is a registered trademark of Linus Torvalds in the United States and other countries.

- `AoDevice` – Use this class to access an analog output (AO) subsystem on a DAQ device. This class contains analog output methods such as `aOut()` and `aOutScan()`, along with methods to access AO subsystem information and configuration.
- `DioDevice` – Use this class to access a digital I/O (DIO) subsystem on a DAQ device. This class contains digital I/O methods such as `dIn()` and `dOut()`, along with methods to access DIO subsystem information and configuration.
- `CioDevice` – Use this class to access a counter I/O (CIO) subsystem on a DAQ Device. This class contains counter I/O methods such as `cIn()` and `cClear()`, along with methods to access CIO subsystem information and configuration.
- `TmrDevice` – Use this class to access a timer I/O subsystem on a DAQ Device. This class contains timer I/O methods such as `tmrOutStart()` and `tmrOutStop()`, along with methods to access timer subsystem information and configuration.

For example, the following code detects all of the DAQ devices currently paired with or attached to an Android system, using the `getDaqDeviceInventory()` method to return *device descriptors* (information about a device) for all detected devices. It then connects to the device, and reads data from AI channel 0.

```
DaqDeviceManager mDaqDeviceManager =
  new DaqDeviceManager(...);


// Find available DAQ devices

ArrayList<DaqDeviceDescriptor> daqDevInventory =

  mDaqDeviceManager.getDaqDeviceInventory();

if(daqDevInventory.size() > 0)

DaqDevice mDaqDevice = mDaqDeviceManager.
createDaqDevice(daqDevInventory[0]);


// Connect to DAQ device

mDaqDevice.connect();


// Read AI channel 0

AiDaqDevice mAiDevice = mDaqDevice.getAiDev();

mAiDevice.aIn(0, ChannelMode.SINGLE_ENDED, Range.
BIP5VOLTS, AiUnit.VOLTS)
```

Example Projects

The UL for Android also includes example projects that can be installed with the API. These examples can help users quickly get familiar with the library. Programmers can also use the source code of these examples as starting points for their own custom Android apps.

Following is a partial list of the example projects and the operations they perform. Those examples that are also available as demo apps for download from the Google Play™ store are indicated by .†

Project Name	Description
AIn	Reads an A/D input channel.
AInScan	Scans a range of A/D input channels and stores the sample data in an array.
AIn_Log	Scans a range of A/D input channels and saves the sample data in comma-separated values (.csv) file.
AInScan_Continuous	Scans a range of A/D input channels continuously in the background and stores the data in an array.
AInScan_Events	Scans D/A channels, displays the latest sample acquired every `EventSize` or more samples, and updates the latest sample when scan completes/ends.
AInScan_ExtClock	Scans a range of A/D input channels and stores the sample data in an array at a sample rate specified by an external clock.
AInScan_ExtTrigger	Selects the trigger source used to initiate the A/D conversion using `DaqDevice.aInScan()` with the `AiScanOption.EXTTRIGGER` option. Selects the trigger source, and displays the analog input on selected channels
AInScan_LoadQueue	Prepares a channel gain queue and loads it to the DAQ device. An analog input function demonstrates how the queue values work.
AInScan_Plot	Scans a range of A/D input channels continuously in the background, and plots the latest acquired samples.
AOut	Writes a value to a specified D/A output channel.
AOutScan	Writes values to a specified D/A output channel.
CIn	Resets and reads the event counter.
DBitIn	Reads the status of single digital input bit.
DBitOut	Sets the state of a single digital output bit.
DBitSetIn	Configures the selected port for input, if necessary, then reads and displays the value on the port.
DIn	Reads a digital input port.
DInScan	Scans a digital port and stores the sample data in an array.
DOut	Writes a value to a specified digital output port.
DOutScan	Writes values to a specified port.
TIn	Reads a temperature channel and displays the value.
TmrOut	Sends a frequency output to a specified timer.

† The Android robot is reproduced or modified from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.

Universal Library for Android

Ordering

When used with the BTH-1208LS, the UL for Android enables DAQ programmers to develop apps that communicate wirelessly with Android-based devices.

Ordering Information

The Universal Library for Android is included on the *MCC DAQ Software* installation CD.

For a list of supported devices or to download the software, visit www.mccdaq.com/ULforAndroid.