ADQAPI_simple_example User's Guide

Table 1: Document history.

Revision	Date	Author	Description
A	2014-09-24	LL	Initial version

TABLE OF CONTENTS

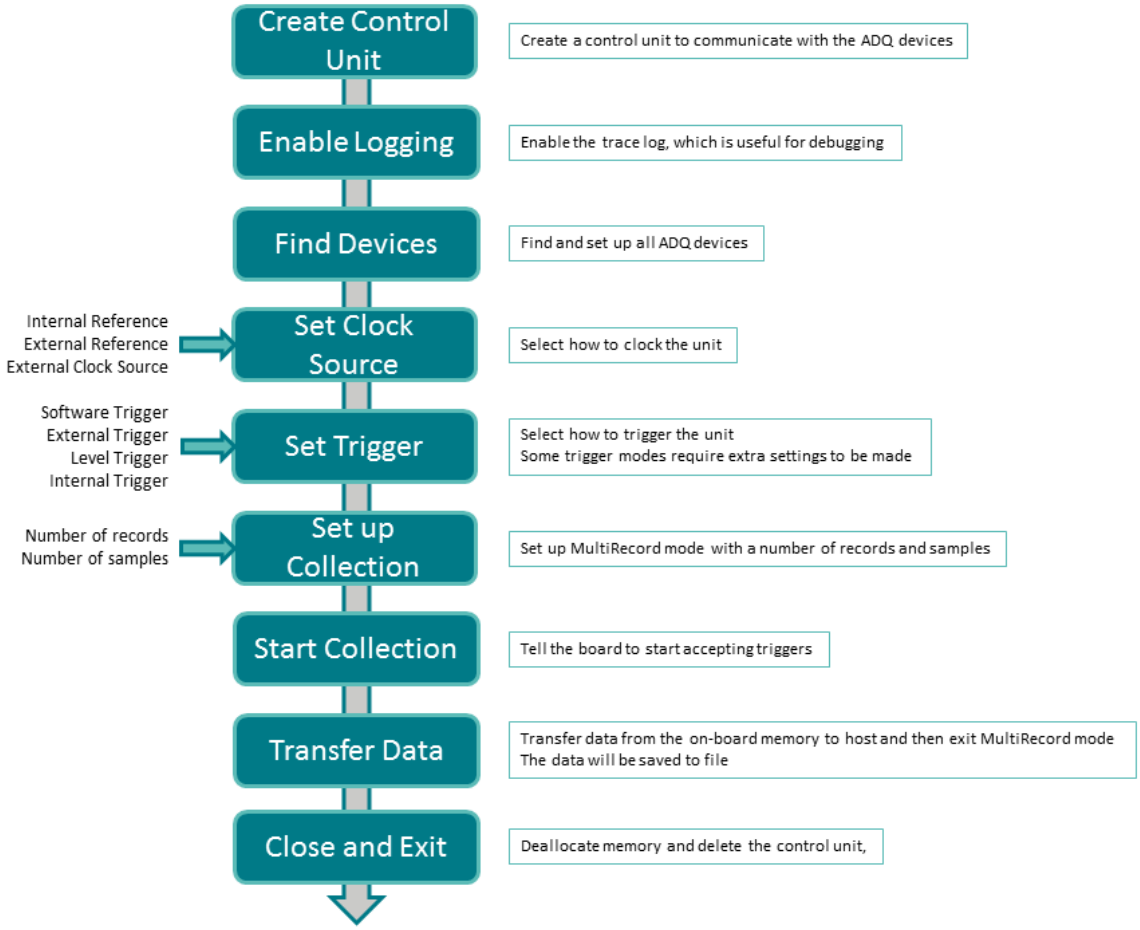
1 OVERVIEW

This document explains the functionality of our basic example, called ADQAPI_simple_example. The example sets up a very basic data collection without the need for the user to change any parameters. This is ideal as a starting point for building your first application.
For Windows, our SDK installer puts this example under the folder named *C_examples*. For the Linux package, it is located in the *examples* folder.

2 PROGRAM FLOW

2.1 Flow Chart

A simplified flow of the example can be seen in the figure below.



2.2 Key functions

All ADQAPI functions in the example are documented in the ADQAPI User's Guide. Here is a short summary of the most important functions, in the order that they are run in the example.

2.2.1 `CreateADQControlUnit`

Before any communication with the ADQ units can be done, an ADQ control unit must be created using this `CreateADQControlUnit`. The function returns a pointer to the control unit, and this pointer is then sent along to the functions of the API.

2.2.2 `ADQControlUnit_FindDevices`

This function searches for ADQ devices, and sets up all it finds. Each device receives its own device number, starting from 1. If several devices are connected to the system, starting them all may take some time. In this case, it is possible

to use the functions `ADQControlUnit_ListDevices`, `ADQControlUnit_OpenDeviceInterface` and `ADQControlUnit_SetupDevice` to only set up the boards that are desired. The ADQAPI user's guide contains more information on how to use these.

2.2.3 ADQ_SetClockSource

There are at least three clock modes for each ADQ device. These are internal clock reference, external clock reference and external clock. The example uses the internal clock reference by default, as this requires no external signal.

2.2.4 ADQ_SetTriggerMode

At least four different trigger modes are available for each ADQ device. These are software trigger (forced through API), external trigger (trigger input on the hardware), level trigger (triggers on signal level) and internal trigger (the ADQ generates triggers repeatedly). The example uses the software trigger by default, and calls the function `ADQ_SWTrig` to trigger the device. External trigger, level trigger and internal trigger are also supported by the example.

2.2.5 ADQ_MultiRecordSetup

Setup of the data collection is done using this function. The inputs decide the number of records (i.e. the number of times the device is trigged) and the number of samples for each record.

2.2.6 ADQ_ArmTrigger

Before calling this function, no triggers are accepted by the board. It is good practice to also call `ADQ_DisarmTrigger` before the trigger is armed, because this will properly shut down any old collection.

2.2.7 ADQ_GetData

This function transfers data from the ADQ to host. As input, it takes one buffer per channel and stores its data there. These buffers must first be allocated, e.g. by using `malloc` or a similar function. Remember to free the allocated memory before leaving the program. The user may also select a desired range of records and samples to store. For more information about each parameter, please refer to the ADQAPI user's guide.

2.2.8 DeleteADQControlUnit

This function destroys the ADQ control unit. This should always be done before exiting the program.

3 COMPILING AND RUNNING

3.1 Windows

The example folder includes project files for Visual Studio. With Visual Studio installed, follow this procedure:

- Unpack the example folder and put it in a suitable location
- Open the ADQAPI_simple_example.sln file with Visual Studio.
- From inside Visual Studio you can now view the source file ADQ_simple_example.cpp
- Change the platform setting to fit your system (Win32 for 32 bit or x64 for 64 bit systems)
- Press F5 to compile and run the program

Of course, other project environments and compilers may be used as well.

3.2 Linux

The example folder contains a make file. Compile it by entering the folder and running make. This will produce the executable file.

4 VIEWING THE DATA

The example saves each record to an .asc-file. We have prepared two convenient ways to plot these files.

4.1 Plotting with ADCaptureLab

With ADCaptureLab, each file can be plotted by either dropping it into the program, or by pressing the "Load Data" button.

4.2 Plotting with Matlab

The script plot_simple_example.m will load and plot each record