Boston Micromachines DM-SDK
Getting Started with C/C++ on Linux
Version 3.1.1

Introduction

The Boston Micromachines Deformable Mirror (**DM**) Software Development Kit (**SDK**) provides a common interface to all BMC products. It allows users to write one code base that can be used with any product. Command line and graphical applications are provided for demonstration of the DM-SDK and hardware functionality. A C API is provided for versatile development of high performance applications.

Minimum Software Requirements

- Linux kernel 3.x or later
- Qt 5.7 or later
- libusb-1

Recommended Software

- Ubuntu 17.04 or later
- GCC 4.8 or later
- Qt 5.8

Installation

The DM-SDK installer installs all the required software, documentation, and examples for DM software development. However, some additional configuration files specific to the hardware are required to operate the DM.

Prerequisites

A few packages that may not be included in the OS need to be installed before using the SDK. To install them on Ubuntu 17.04 run the following command:

```
sudo apt install libtbb2 liblog4cxx10v5 libqt5widgets5 libusb-1.0-0
```

For other distributions, please refer to the documentation for your distribution to install these packages.

Installation

Copy the installer to the machine, then run it from a terminal:

```
chmod +x DMSDK-setup-linux-x64-3.0.0.run
sudo ./DMSDK-setup-linux-x64-3.0.0.run
```

The installer installs device drivers and requires root privileges. Unprivileged user installation is not supported at this time.

Profiles

The SDK uses a configuration file matched to a DM and driver combination. These profiles contain information such as DM array size, driver interface, and other internal settings. The profile for your DM is identified by the serial number of your mirror and should be placed in /opt/Boston Micromachines/Profiles.

The profiles directory also contains some generic profiles, such as MultiUSB000.dm. Note that these generic profiles may not use the correct actuator mapping and limit driver output to half of its maximum to protect all mirrors from overvoltage conditions in case a mirror is connected to the incorrect driver.

Calibration Tables

A calibration table for open loop operation is provided with some hardware. It should be placed in:

> /opt/Boston Micromachines/Calibration

Do not rename the file when installing it.

Uninstallation

To uninstall the SDK and device drivers, run the following command, then select “Remove all components”:

> sudo /opt/Boston\ Micromachines/bmc_maintenancetool

Kernel Updates

When the OS kernel is updated, the driver for the PCIe card needs to be reinstalled. Run the following command, then select “Rebuild PCIe card driver”:

> sudo /opt/Boston\ Micromachines/bmc_maintenancetool

API Documentation

This document is intended as a primer. Comprehensive API documentation is available at:

> /opt/Boston Micromachines/Doc/html/index.html

Applications

The SDK contains several GUI and command line applications to control the mirror. The source code for command line applications is included to ease application development and demonstrate the basic features of the SDK.

DM Shapes

DM Shapes is a graphical user interface application that can load a shape from a file, set the entire DM to a particular value (piston), or set actuators individually (poke). See the user manual for complete details on the use of this application.

BMCExample

The BMCExample command line application performs two tests and is intended as a simple verification that the library installed correctly. It will first piston the DM between two values for 5 cycles then poke each actuator. Depending on input options, BMCExample can advance automatically with a specified delay or advance on user input. The help message displayed when the program is run explains the options and their format.

The source code for BMCExample is contained in /opt/Boston Micromachines/Examples.

```
Boston Micromachines DM tester.
Test 1: Piston mirror between two values.
Test 2: Poke each actuator to half amplitude.
Options are:
    -s xxxxxxx  Device serial number. If this
                option is not used, you will be
                prompted to enter the SN.
    -c x      Run test continuously with x
                millisecond delay between commands.
    -h         Display this message and exit.
```

BMCLoadShape

The BMCLoadShape command line application reads a shape from a text file and applies it to the mirror. The default shape is a repeating ramp. The shape file can be specified with the -f option. The help message displayed when the program is run explains the options and their format.

The source code for BMCLoadShape is contained in /opt/Boston Micromachines/Examples.

```
Boston Micromachines Load Shape Example.
Load shape from file and exit on user input.
Options are:
    -s xxxxxxx  Device serial number. If this
                option is not used, you will be
                prompted to enter the SN.
    -f x       Use file x as the shape. The file
                should be a list of numbers in [0,1].
    -h         Display this message and exit.
```

BMCExampleTiltSegments

The BMCExampleTiltSegments command line application performs two tests and is intended as a simple verification that the library installed correctly. It will first print out the available piston and tilt ranges from the DM's calibration table. Second, it will piston each segment one by one to half the maximum value and a small tilt. Third, it will piston and tilt all the segments simultaneously. Depending on input options, BMCExampleTiltSegments can advance automatically with a specified delay or advance on user input. The help message displayed when the program is run explains the options and their format.

The source code for BMCExampleTiltSegments is contained in /opt/Boston Micromachines/Examples.

```
Boston Micromachines DM tester.
Test 1: Piston and tilt each segment one by one.
Test 2: Piston and tilt all segments simultaneously.
Options are:
    -s xxxxxxx  Device serial number. If this
                option is not used, you will be
                prompted to enter the SN.
    -c x        Run test continuously with x
                millisecond delay between commands.
    -a          Exit when done, don't wait for input.

    -h          Display this message and exit.
```

Example Code Summary (C++)

```
// Include the BMC DM-SDK library functions
#include <BmcApi.h>
…
// Allocate data structures
DM hdm = {};
BMCRC rv = NO_ERR;
char serial_number[12] = "MultiUSB000";
…
// Open driver
rv = BMCOpen(&hdm, serial_number);
// Check error code
if (rv != NO_ERR) {
    std::cerr << "Error " << rv << " opening the driver type " << hdm.Driver_Type;
    std::cerr << BMCErrorString(rv) << std::endl << std::endl;
    …
}
…
// Allocate and initialize data arrays.
std::vector<unsigned int> map_lut(MAX_DM_SIZE, 0);
std::vector<double> test_array1(hdm.ActCount, 0.0);
// Open default actuator map from disk and read into a vector
```

```
rv = BMCLoadMap(&hdm, NULL, map_lut.data());
// Set all actuators to a vector of values
rv = BMCSetArray(&hdm, test_array1.data(), map_lut.data());
// Set a single actuator to a value
rv = BMCSetSingle(hdm, k, pokeValue);
// Load an open loop calibration table
rv = BMCLoadCalibrationFile(&hdm, "Lookup_Table.mat");
// Get full piston range with no tilt
double minPiston, maxPiston;
double xTilt = 0, yTilt = 0;
rv = BMCGetSegmentRange(&hdm, 0, DM_Piston, 0, xTilt, yTilt, &minPiston, &maxPiston);
// Piston and tilt a segment (open loop)
rv = BMCSetSegment(&hdm, k, pistonCmd, tiltCmd, tiltCmd, true);
// Set all actuators to 0
rv = BMCClearArray(&hdm);
// Close the driver
rv = BMCClose(&hdm);
```