CTA2045 UCM Library

License

Overview

This software implements the CTA2045 UCM protocol as defined by the Consumer Electronics Association in the R7.8 Modular Communication Interface for Energy Management subcommittee.

This project includes three sub projects:

- **cea2045**: Core CTA2045 Library
- **sample2**: Sample program demonstrating how to create a
- **test**: Unit testing program

The library is written in C++ and uses some features of C++11.

Dependencies

This project was developed and tested on Ubuntu 16.04, and is running in production on OpenWRT Linux for EPRI's UCM (MIPS) as well as Raspbien and OpenWRT on mulitple revisions of Raspberry pi (ARM).

This project does not rely on any 3rd party libraries, but requires the following build dependencies:

- cmake, make, g++

These dependencies can bin installed on Ubuntu with the following command:

```
sudo apt-get install g++ cmake make
```

Build instructions

Run the following commands depending on the type of build required. The commands assume starting from the root of the source tree.

Building the library will generate a file called `libcea2045.so`. Building the test program and sample2 projects will generate files called `testcea2045` and `sample2` respectively.

- **Debug**

```
mkdir -p build/debug
cd build/debug
cmake -DCMAKE_BUILD_TYPE=Debug ../../
make
```

- **Debug w/test and sample programs**

```
mkdir -p build/debug
cd build/debug
cmake -DCMAKE_BUILD_TYPE=Debug -DSAMPLE=1 -DTEST=1 ../../
make
```

The test program is a set of unit tests. Run it with:

```
./testcea2045
```

If the tests are successful, the following should be output:

```
===============================================================
All tests passed (164 assertions in 11 test cases)
```

- **Release** (for production builds)

```
mkdir -p build/release
cd build/release
cmake -DCMAKE_BUILD_TYPE=Release ../../
make
```

Notes on the sample program

The sample program can be used to communicate to an SGD device through a serial port. It can be tested with test cables and the EPRI CEA2045 simulator for example.

The sample program will attempt to open `/dev/ttyUSB0`. Adjust this file as necessary on line 27 of `main.cpp`.

On startup, the program will send a few commands to the SGD to determine what features are supported. Next, the program runs in a loop waiting for user input to send commands to an SGD.

Creating a UCM with the library

The best way to learn how to use the library is to look at `sample2`, but here are a few tips.

The library uses callbacks to notify user code when responses and unsolicited messages are received from the SGD. The programmer passes a `IUCM` implementation (which contains the callbacks) to the `DeviceFactory::createUCM()` function when creating a device.

Communicating with an SGD

The device won't start processing messages until the device's `start` function is called. To stop communication, call `shutDown` on the device. Calling the `start` creates a thread which processes incoming and outgoing messages from the serial port.

Functions called on the device are non-blocking: messages are queued and processed in order; each function call returns a future. To force your program to block until the command is complete, call `get()` on the returned future.

Each function call returns a `ResponseCode` object which indicates the success or failure of the function call.