

EloMcp2210Api.aar

Elo Status Light Kit Android Library

Elo Touch Solutions

2020-11-25

Michael Qi

1. Summary

The Elo Android library for MCP2210 Status Light Kit is contained in a single Android library “EloMcp2210Api.aar”. The library combines Android libraries provided by Microchip as shown in list below as of Sept 2020 but with Elo’s additions/modifications:

Constants

Mcp2210

Mcp2210Comm

Mcp2210Constants

MicrochipUsb

2. How To Use EloMcp2210Api.aar

2.1 The user needs to set EloMcp2210Api.aar as dependency of their own app.

2.2 To access the APIs within the library, the user needs to create an instance of the library by passing the calling activity and the broadcast receiver (processing USB permission/attach/detach events) to the library class’s constructor as below:

EloMcp2210Api EloApi = new EloMcp2210Api (Activity _activity, BroadcastReceiver _receiver);

Please refer to the demo code of EloMcp2210Config demo on details implementing the BroadcastReceiver which process device permission/attachment/detachment.

2.3 open all MCP2210 devices:

int mcp2210_count = EloApi.openAllMcp2210();

2.4 To get list of connected MCP2210:

Map <String, UsbDevice> DeviceMap = EloApi.getMcp2210Map();

Where “DeviceMap” is a Map (list of <Serial Number, UsbDevice> pairs) with MCP2210 device’s serial number as key.

Below is sample code to access each of the connected MCP2210 device (“sn” below stands for MCP2210’s serial number):

```

for (String sn : DeviceMap.keySet())
{
    // access APIs through m_EloApi based on device serial number...
}

```

For detailed usage of the APIs, please refer to the source code of the demo app.

3. List of APIs

3.1 public int getDeviceCount()

Get total number of MCP2210 devices.

3.2 Public Map getMcp2210Map()

Return a Map (list of <Serial Number, UsbDevice> pairs) containing all serial numbers and device.

The definition of the Map is:

Map <String serial_number, UsbDevice usb_device_object>

3.3 int setLight (String sn, int nPhase, LightColor color, boolean bLightOn)

This API is used to turn on/off a particular light.

sn: MCP2210 device's serial number.

nPhase: either **CURRENT_SETTINGS_ONLY** (0) or **PWRUP_DEFAULTS_ONLY** (1).

LightColor: enum type defining light colors: RED, GREEN and BLUE.

All further references to **sn** and **nPhase** parameters in all APIs have the same meaning as this API.

Below are example usage/calls of this **setLight()**:

3.3.1 To turn on the red light for device with serial number "sn":

```
setLight (sn, CURRENT_SETTINGS_ONLY, RED, true);
```

3.3.2 To turn off the red light for device with serial number "sn":

```
setLight (sn, CURRENT_SETTINGS_ONLY, RED, false);
```

3.3.3 To turns on the red light for device with serial number "sn" when the device is powered on:

```
setLight (sn, PWRUP_DEFAULTS_ONLY, RED, true);
```

3.3.4 To get a mixed color, for example purple, the user can turn on both the red light and blue light:

```
setLight (sn, CURRENT_SETTINGS_ONLY, RED, true);
```

```
setLight (sn, CURRENT_SETTINGS_ONLY, BLUE, true);
```

3.4 boolean getLight (String sn, int nPhase, LightColor color)

Return true if the particular colored light is currently on; or false if the light is off.

For example, to check if the red light is currently on:

```
Boolean red_light_on = getLight (sn, CURRENT_SETTINGS_ONLY, RED);
```

3.5 public int getSpiMode (String sn, int nPhase)

SPI mode: 0,1,2 or 3.

Return value is less than 0 when an error occurred.

3.6 public int getGpioPinDesignations (String sn, int nPhase, ByteBuffer bbDsgn)

bbDsgn: Array of 9 elements specifying each GP pin's designation:

GPIO = 0x00

Chip Selects = 0x01

Dedicated Function pin = 0x02

If the output ByteBuffer is NULL, then an error occurred.

3.7 public int getGpioPinDirection (String sn, int nPhase)

Get GPIO Pin direction: output(0), input(1)

Return value:

Lower 9 bits used (MSB to LSB GP8 to GP0).

Less than zero if error occurred.

3.8 public int getGpioPinOutput (String sn, int nPhase)

Get GPIO Pin's output value.

Lower 9 bits used (MSB to LSB GP8 to GP0) as output value (0 or 1).

Less than zero if error occurred.

3.9 public int getSpiCsActiveValue (String sn, int nPhase)

Return value of SPI chip select active value

Less than zero if error occurred.

3.10 public int getSpiCsIdleValue (String sn, int nPhase)

Return value of SPI chip select idle value

Less than zero if error occurred.

3.11 public int setSpiMode(String sn, int nPhase, int nSpiMode)

nSpiMode: Specify SPI mode 0, 1, 2, or 3

Return 0 for success, otherwise failure code code.

3.12 public int setSpiCsValues (String sn, int nPhase, int nIdleValue, int nActiveValue)

nIdleCsValSet – [IN] IDLE chip select value

nActiveCsValSet - [IN] ACTIVE chip select value

Return 0 for success, otherwise failure code.

3.13 public int setGpioConfig (String sn, int nPhase, ByteBuffer bbPinDsgn, int nPinOutputs, int nPinDirections)

bbPinDsgn – [IN] Array of 9 elements specifying each GPIO pin designation

(0=GPIO, 1=Chip-selects, 2=Dedicated pin function)

nPinOutput - [IN] Default GPIO output. Mapping(MSB to LSB - only lower 9 bits used):

GP8VAL GP7VAL GP6VAL GP5VAL AL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

nPinDirections - [IN] Default GPIO direction, where 0=output and 1=input Mapping

MSB to LSB - only lower 9 bits used:

GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

Returns:

Error code Indicates if the operation was successful or not. 0=successful; non-zero = failed.

--- END OF DOCUMENT ---