SLCD Application Note AN-111

Sample GPIO QML Example
Reach Technology, Inc.
11/10/2015

Reach Technology Inc.

www.reachtech.com

Sales: 510-770-1417 x112
sales@reachtech.com

Technical Support: 503-675-6464
techsupport@reachtech.com

1 Overview

This application note describes a sample QML application that demonstrates how to execute Linux commands from within the QML application, and then using these commands, how to access the I2C bus on a G2 Display Module.

The basic function of this application provides 8 switches in the GUI, with each switch controlling a single GPIO output on the display module. With LEDs attached to the GPIO pins on the G2's GPIO connector, as shown in the Schematics section, the switches can turn the LEDs on and off. Alternatively, you can connect the 11-pin flying lead cable (P/N 23-0144-10) to the GPIO connector and measure the voltage on each pin as it is turned on or off.

The Schematics section shows a simple circuit with an LED and associated current-limiting resistor on each GPIO pin. The GPIO signals are set high when the associated switch is turned on.

This application note assumes that the reader has installed G2Link from http://linuxtouchscreen.com/support/download-center/g2link.

Note: Reach provides a set of GPIO functions for use within QML applications to access the GPIO pins on the display module. The GPIO functions are the preferred method for accessing the GPIO pins. This application note is merely using the GPIO pins as a handy platform to demonstrate accessing the I2C bus from a QML application.

1.1 G2H2 7" Display Module

The user interface is setup for the 7" G2H2 Display Module. Other G2H2 modules will work as well, but the canvas size and the switch placement would need to be changed.

This application note assumes that the reader has a copy of the G2H2 Hardware Manual, from http://linuxtouchscreen.com/products/production-modules/7-modules/, in the Documents section.

The code for this application note is written with Qt Creator 3.2.1, as installed from http://linuxtouchscreen.com/support/download-center/qt-creator/.

The GPIO connector is J22 on the G2H2 Display Module.

1.2 G2C1 4.3" Display Module

The user interface is setup for the 4.3" G2C1 Display Module.

This application note assumes that the reader has a copy of the G2C1 Hardware Manual, from http://linuxtouchscreen.com/products/production-modules/4-3-modules, in the Documents section of the specific model you have.

The code for this application note is written with Qt Creator 2.8.0, as installed from http://linuxtouchscreen.com/support/download-center/qt-creator/.

The GPIO connector is J8 on the G2C1 Display Module.

2 Program Structure

The program consists of a single QML file, mainview.qml, which is derived from the template found in the Empty Project folder from the QML Samples on http://linuxtouchscreen.com/support/download-center/demos-and-examples.

There are eight `AnimatedSwitch` components for the GUI switches, eight `Text` components for the switch labels, a *`set_pin()`* function that is used to set a single output pin to a specified state, and a `Component.onCompleted` handler to initialize the GPIO port expander on the G2H2 controller.

Each switch uses an `onOnChanged` handler to call the *`set_pin()`* function when the switch state is changed.

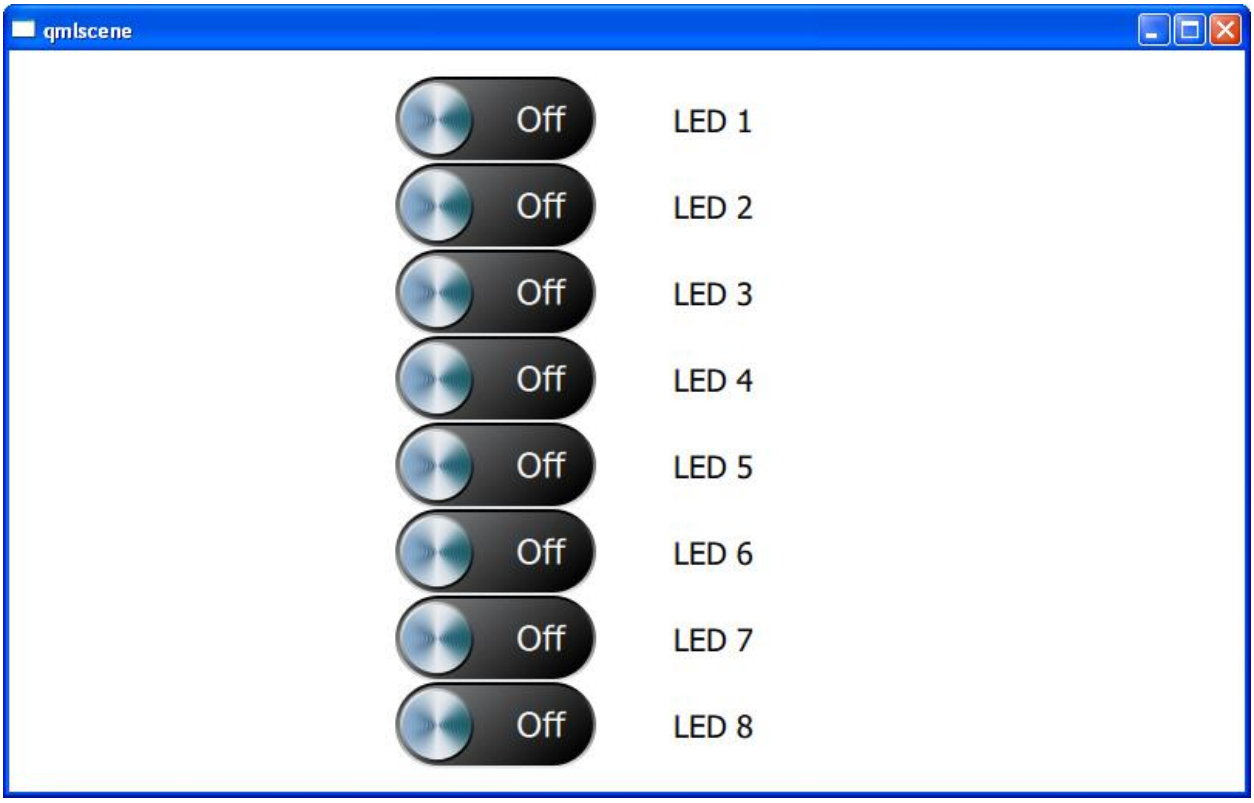
The *`set_pin()`* function gets the current GPIO bit settings value, clears the current setting for the specified pin, sets the new state of the specified pin, then writes the new GPIO bit settings value.

The code in *`set_pin()`* and in the `Component.onCompleted` handler are specific to the GPIO port expander, a TI PCA9534 Remote 8-Bit I2C I/O Expander.

Note that the code shown in the App Note is set specifically for the G2H2 7" Display Module. There is another source code folder for the G2C1 4.3" Display Module.

3 Application Screen

The GUI screen contains a set of eight switches with labels:



4 Linux Commands Used

This application uses two standard Linux commands to access the I2C bus: `i2cget` and `i2cset`. The command details are shown below.

/usr/sbin/i2cget:

```
Usage: i2cget [-f] [-y] I2CBUS CHIP-ADDRESS [DATA-ADDRESS [MODE]]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77)
  MODE is one of:
    b (read byte data, default)
    w (read word data)
    c (write byte/read byte)
    Append p for SMBus PEC
```

/usr/sbin/i2cset:

```
Usage: i2cset [-f] [-y] [-m MASK] [-r] I2CBUS CHIP-ADDRESS DATA-ADDRESS
[VALUE] ... [MODE]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77)
  MODE is one of:
    c (byte, no value)
    b (byte data, default)
    w (word data)
    i (I2C block data)
    s (SMBus block data)
    Append p for SMBus PEC
```

5 QML Source Code

```
import QtQuick 2.0
import com.reachtech.systemplugin 1.0
import "components"

/*
 * GPIO_I2C_Demo:
 *
 *  Sample application showing how to use the I2C bus from a QML application.
 *  Uses the 'system' plugin to 'execute' standard command-line binaries to
 *  read/write the G2H's GPIO Expander on the I2C bus.
 *
 *  Setup for use on a G2H2-7R Display Module.
 *
 *  The code in set_pin() and Component.onCompleted is specific to the
 *  TI PCA9534 Remote 8-Bit I2C I/O Expander.
 */

Rectangle {
    id: page
    width: 800
    height: 480
    property int sa: 0x3E               // GPIO Expander slave address
    property int inp: 0                 // port expander input port
    property int outp: 1                // port expander output port
    property int polarity_inversion: 2
    property int config: 3              // port expander config register


    System {
        id: system
    }

    AnimatedSwitch {
        id: animatedSwitch1
        x: 250
        y: 17
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
```

```
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(0, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch2
        x: 250
        y: 73
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(1, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch3
        x: 250
        y: 129
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
```

```
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(2, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch4
        x: 250
        y: 185
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(3, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch5
        x: 250
        y: 241
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
```

```
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(4, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch6
        x: 250
        y: 297
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(5, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch7
        x: 250
        y: 353
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
```

```
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(6, on)
        }
    }

    AnimatedSwitch {
        id: animatedSwitch8
        x: 250
        y: 409
        width: 130
        height: 56
        font.pixelSize: 22
        textOn: "On"
        sliderImageY: 2
        textOnX: 18
        font.bold: false
        textColor: "#ffffff"
        sliderImageOffX: 78
        imageOn: ""
        textOff: "Off"
        textOffX: 78
        imageOff: "images/background.svg"
        imageOffHeight: 56
        sliderImage: "images/knob.svg"
        font.family: "DejaVu Sans"
        imageOffWidth: 130
        sliderImageOnX: 1
        on: false

        onOnChanged: {
            set_pin(7, on)
        }
    }

    Text {
        id: text1
        x: 430
        y: 33
        text: qsTr("LED 1")
        font.pixelSize: 20
    }

    Text {
        id: text2
        x: 430
        y: 89
        text: "LED 2"
        font.pixelSize: 20
    }
```

```
Text {
    id: text3
    x: 430
    y: 145
    text: "LED 3"
    font.pixelSize: 20
}

Text {
    id: text4
    x: 430
    y: 201
    text: "LED 4"
    font.pixelSize: 20
}

Text {
    id: text5
    x: 430
    y: 257
    text: "LED 5"
    font.pixelSize: 20
}

Text {
    id: text6
    x: 430
    y: 313
    text: "LED 6"
    font.pixelSize: 20
}

Text {
    id: text7
    x: 430
    y: 369
    text: "LED 7"
    font.pixelSize: 20
}

Text {
    id: text8
    x: 430
    y: 425
    text: qsTr("LED 8")
    font.pixelSize: 20
}
```

```
    /*
     * set_pin is an example of using javascript in a QML file
     */
    function set_pin(pin, state)
    {
        // convert pin number into bit mask
        var bit = (1 << pin);
        // create new bit setting
        var val = (state === true) ? (1 << pin) : 0;
        // get current output register value
        var ret = system.execute
                ("/usr/sbin/i2cget -y 0 %1 %2 b".arg(sa).arg(outp))
        // clear desired bit, OR in new setting
        var ret_val = (parseInt(ret) & ~bit) | val;
        // set new output register value
        system.execute
         ("/usr/sbin/i2cset -y 0 %1 %2 %3 b".arg(sa).arg(outp).arg(ret_val))
    }

    Component.onCompleted: {
        // setup GPIO port pins (all) as output
        system.execute
                ("/usr/sbin/i2cset -y 0 %1 %2 0x00 b".arg(sa).arg(config))
        // set pins to low (LEDs off)
        system.execute
                ("/usr/sbin/i2cset -y 0 %1 %2 0x00 b".arg(sa).arg(outp))
    }

}
```

6 Schematics

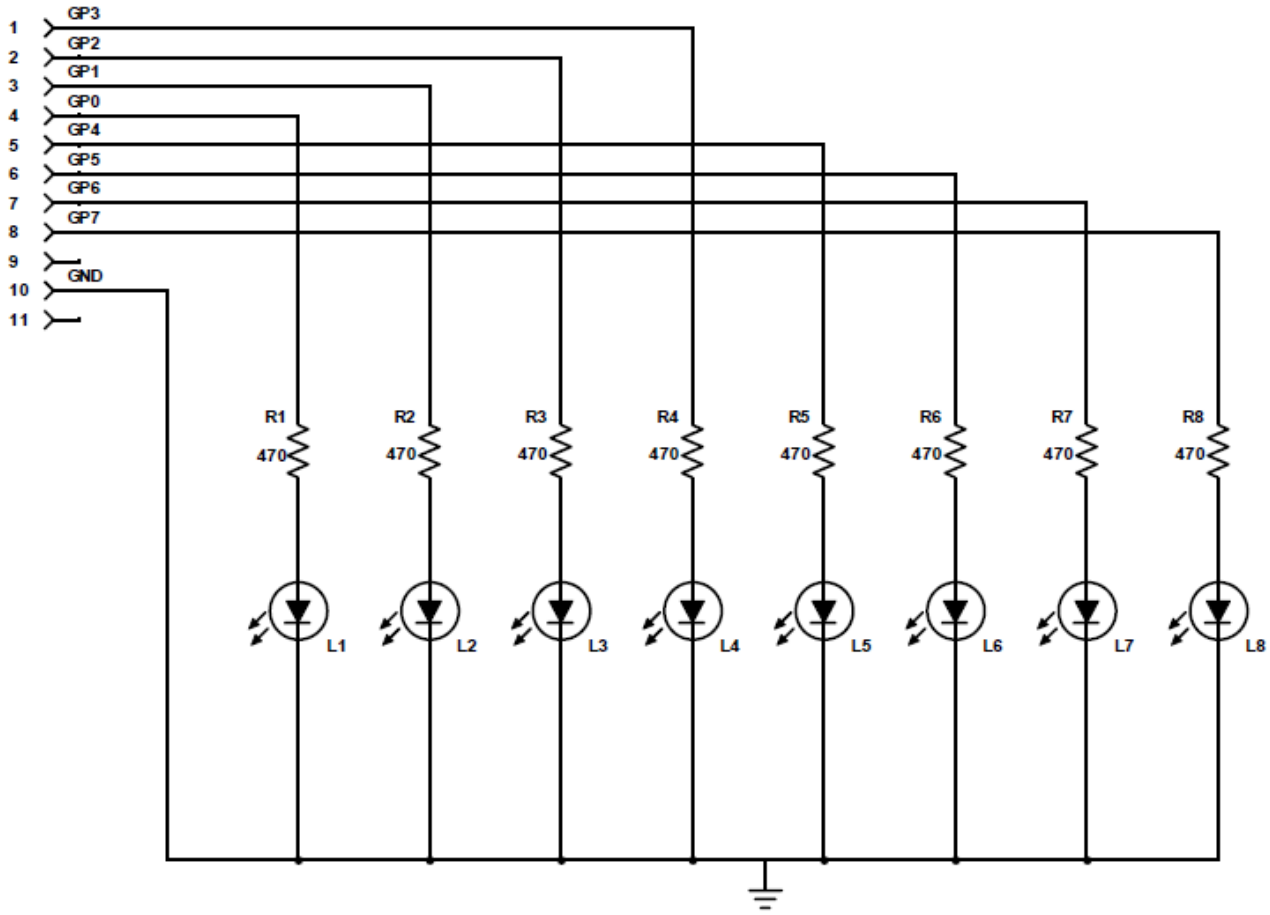


Figure 1: LED Board