new-gpio

Version 1.1 – 5 December 2015

Kit Bishop

Document History

Version	Date	Details
Version 1.0	26 November 2015	Initial version
Version 1.1	5 December 2015	Added PWM control methods in GPIOAccess and GPIOPin classes in libnew-gpio.so library Added PWM control in new-gpiotest program Some reorganisation of contents of this document

Contents

1. Background

new-gpio is alternative C++ code for accessing the Omega GPIO pins.

The rationale for producing this code was two-fold:

- A desire for GPIO access with different features and capability than **fast-gpio**
- An exercise in developing C++ code for the Omega

new-gpio consists of two main components:

- **libnew-gpio.so** – a dynamic library containing the classes used to intercat with GPIO pins
- **new-gpiotest** – a simple test program for interacting with GPIO pins using **libnew-gpio.so**

These components are described in more details in this document, as are the files contained in the package supplied with this document.

The software was developed under NetBeans 8.1 running on a KUbuntu-14.04 system running in a VirtualBox VM.

Details of setting up the tool-chain for building and the usage of NetBeans can be found at:

- How to install gcc (https://community.onion.io/topic/9/how-to-install-gcc)
- Using NetBeans to compile C/C++ code for Omega (https://community.onion.io/topic/125/using-netbeans-to-compile-c-c-code-for-omega)

new-gpio comes with **no guarantees** ☺ but you are free to use it and do what you want with it.

NOTE: Some of the code in the class **GPIOAccess** as described below was derived from code in **fast-gpio**

2. Files Supplied

new-gpio is supplied in an archive file named **new-gpio.tar.bz2**. This archive contains the following:

- **new-gpio.pdf** – this documentation as a PDF file
- **libnew-gpio.so** – the built dynamic library
- **new-gpiotest** – the built test program
- **new-gpio-src** – a directory containing the source files for the **libnew-gpio.so** library
- **new-gpiotest-src** – a directory containing the source file for the **new-gpiotest** program

3. Basic Installation

Installing the software is simple. It primarily consists of copying the library and test program to suitable locations on your Omega.

3.1. Installing libnew-gpio.so

Copy the **libnew-gpio.so** file to the **/lib** directory on your Omega.

Alternatively, you can copy the library to any location that may be set up in any **LD_LIBRARY_PATH** directory on your Omega. For example, I use the following for testing:

- Created directory **/root/lib**
- Copied the library to **/root/lib**
- Added the following lines to my **/etc/profile** file:

```
LD_LIBRARY_PATH=/root/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

3.2. Installing the new-gpiotest Program

Copy the **new-gpiotest** program file to any suitable directory on your Omega from which you wish to run it.

4. Description of the libnew-gpio.so Library

The **libnew-gpio.so** library contains three main components. These components and their source files are:

- **GPIOTypes** – defines a few basic types used elsewhere
  File:
  GPIOTypes.h
- **GPIOAccess** – a class used for direct access to the Omega GPIO hardware.
  Contains only **static** methods for access.

Files:

GPIOAccess.h
GPIOAccess.cpp

- **GPIOPwmPin** – a support class used only internal by **GPIOAccess** to provide PWM facilities for a pin
  Files:
  GPIOPwmPin.h
  GPIOPwmPin.cpp
- **GPIOPin** – a class used to represent instances of a GPIO pin.
  Contains methods to interact with the specific pin.
  Files:
  GPIOPin.h
  GPIOPin.cpp

4.1. GPIOTypes

The file **GPIOTypes.h** contains definitions of some basic types used elsewhere.

4.1.1. enum GPIO_Result

enum GPIO_Result is used to represent the returned result of GPIO operations. It has values:

- **GPIO_OK = 0** – represents a successful result
- **GPIO_BAD_ACCESS = 1** – indicates a failure to access the GPIO hardware registers
- **GPIO_INVALID_PIN = 2** – indicates that a pin number has been used that is not accessible by GPIO
- **GPIO_INVALID_OP = 3** – indicates that an invalid operation has been attempted on a pin. E.G. attempting to set a pin that is in input mode, or reading a pin that is in output mode

4.1.2. enum GPIO_Direction

enum GPIO_Direction is used to represent the direction for a GPIO pin. It has values:

- **GPIO_INPUT = false** – represents an input pin
- **GPIO_OUTPUT = true** – represents an output pin

4.2. Class GPIOAccess

The **GPIOAccess** class is the main method by which all access is made to the GPIO hardware.

The class contains only static methods and no instance of this class will ever actually be created hence there are no constructors or destructors.

4.2.1. GPIOAccess Public Methods

4.2.1.1. *static GPIO_Result setDirection(int pinNum, GPIO_Direction dir);*

Sets the direction for a pin.

Parameters:

- **int pinNum** – the number of the pin
- **GPIO_Direction dir** – the direction to set the pin to

Returns:

- An indication of the success or failure of the method

4.2.1.2. *static GPIO_Result getDirection(int pinNum, GPIO_Direction &dir);*

Queries the direction of a pin.

Parameters:

- **int pinNum** – the number of the pin
- **GPIO_Direction &dir** – returns the current direction of the pin

Returns:

- An indication of the success or failure of the method

4.2.1.3. *static GPIO_Result set(int pinNum, int value);*

Sets the output state of a pin. Only valid for output pins.

Parameters:

- **int pinNum** – the number of the pin
- **int value** – the value to set the pin to

Returns:

- An indication of the success or failure of the method

4.2.1.4. *static GPIO_Result get(int pinNum, int &value);*

Queries the input state of a pin. Only valid for input pins.

Parameters:

- **int pinNum** – the number of the pin
- **int &value** – returns the current state of the pin

Returns:

- An indication of the success or failure of the method

4.2.1.5. *static GPIO_Result setPWM(int pinNum, int freq, int duty);*

Starts the PWM output on a pin with the given frequency and duty values.

NOTE: PWM output on a pin is run on a separate thread for that pin. When this method is called the thread will be started (or its data updated if it is already running) and the call to the method then returns. The thread continues to run until one of the following occurs:

- the **stopPWM** method is called for the pin
- the process that started the thread (i.e. made the call to this method) terminates

Parameters:

- **int pinNum** – the number of the pin
- **int freq** – sets the PWM frequency in Hz
- **int duty** – sets the PWM duty cycle percentage

Returns:

- An indication of the success or failure of the method

4.2.1.6. static GPIO_Result startPWM(int pinNum);

Starts the PWM output on a pin using the last used frequency and duty values for the pin.

Parameters:

- **int pinNum** – the number of the pin

Returns:

- An indication of the success or failure of the method

4.2.1.7. static GPIO_Result stopPWM(int pinNum);

Stops any current PWM output on a pin.

Parameters:

- **int pinNum** – the number of the pin

Returns:

- An indication of the success or failure of the method

4.2.1.8. static int getPWMFreq(int pinNum);

Returns the currently set PWM frequency for a pin.

Parameters:

- **int pinNum** – the number of the pin

Returns:

- The PWM frequency in Hz

4.2.1.9. static int getPWMDuty(int pinNum);

Returns the currently set PWM duty cycle percentage for a pin

Parameters:

- **int pinNum** – the number of the pin

Returns:

- The PWM duty cycle percentage

4.2.1.10. static bool isPWMRunning(int pinNum);

Returns an indication of whether or not PWM is currently running on a pin

Parameters:

- **int pinNum** – the number of the pin

Returns:

- **true** if PWM is running; **false** if PWM is not running

4.2.1.11. static bool isPinUsable(int pinNum);

Returns an indication as to whether or not a specific pin number can be used for a GPIO pin.

Parameters:

- **int pinNum** – the number of the pin

Returns:

- **true** or **false –** indicating whether or not **pinNum** is a valid GPIO pin

4.2.1.12. static bool isAccessOk();

Returns an indication as to whether or not the GPIO hardware is accessible.

Parameters:

- <none>

Returns:

- **true** or **false –** indicating whether or not the hardware is accessible

4.3. Class GPIOPin

The **GPIOPin** class represents instances of a GPIO pin.

4.3.1. GPIOPin Constructor and Destructor

4.3.1.1. Constructor - GPIOPin(int pinNum);

Creates a new GPIOPin instance for a given pin.

Parameters:

- **Int pinNum** – the pin number

4.3.1.2. Destructor - ~GPIOPin(void);

Destroys an instance of a GPIOPin.

NOTE: This also ensures that any PWM thread for the pin is terminated.

Parameters:

- <none>

4.3.2. GPIOPin Public Methods

4.3.2.1. GPIO_Result setDirection(GPIO_Direction dir);

Sets the direction of the GPIOPin.

Parameters:

- **GPIO_Direction dir** – the direction to set the pin to

Returns:

- An indication of the success or failure of the method

4.3.2.2. GPIO_Result getDirection(GPIO_Direction &dir);

Obtains the current direction of the GPIOPin

Parameters:

- **GPIO_Direction &dir** – is set to the current direction of the pin

Returns:

- An indication of the success or failure of the method

4.3.2.3. GPIO_Direction getDirection();

Directly returns the direction of the GPIOPin.

Parameters:

- <none>

Returns:

- the current direction of the pin

4.3.2.4. *GPIO_Result set(int value);*

Sets the value of the GPIOPin.

Parameters:

- **Int value** – the value to set the pin to

Returns:

- An indication of the success or failure of the method

4.3.2.5. *GPIO_Result get(int &value);*

Obtains the current value of the GPIOPin

Parameters:

- **int &value** – is set to the current value of the pin

Returns:

- An indication of the success or failure of the method

4.3.2.6. *int get();*

Directly returns the value of the GPIOPin.

Parameters:

- <none>

Returns:

- the current value of the pin

4.3.2.7. *GPIO_Result setPWM(int freq, int duty);*

Starts the PWM output on the GPIOPin with the given frequency and duty values.

NOTE: PWM output on a pin is run on a separate thread for that pin. When this method is called the thread will be started (or its data updated if it is already running) and the call to the method then returns. The thread continues to run until one of the following occurs:

- the **stopPWM** method is called for the pin
- the GPIOPin destructor for the pin is called
- the process that started the thread (i.e. made the call to this method) terminates

Parameters:

- **int freq** – sets the PWM frequency in Hz
- **int duty** – sets the PWM duty cycle percentage

Returns:

- An indication of the success or failure of the method

4.3.2.8. *GPIO_Result startPWM();*

Starts the PWM output on the GPIOPin using the last used frequency and duty values

Parameters:

- <none>

Returns:

- An indication of the success or failure of the method

4.3.2.9. *GPIO_Result stopPWM();*

Stops any current PWM output on the GPIOPin

Parameters:

- <none>

Returns:

- An indication of the success or failure of the method

4.3.2.10. *int getPWMFreq();*

Returns the currently set PWM frequency for the GPIOPin

Parameters:

- <none>

Returns:

- The PWM frequency in Hz

4.3.2.11. *int getPWMDuty();*

Returns the currently set PWM duty cycle percentage for the GPIOPin

Parameters:

- <none>

Returns:

- The PWM duty cycle percentage

4.3.2.12. bool isPWMRunning();

Returns an indication of whether or not PWM is currently running on the GPIOPin

Parameters:

- <none>

Returns:

- **true** if PWM is running; **false** if PWM is not running

4.3.2.13. int getPinNumber();

Returns the pin number for the GPIOPin

Parameters:

- <none>

Returns:

- The pin number

5. Usage of the new-gpiotest Program

The **new-gpiotest** program accepts a set of parameters to control its operation.

The program will document it's usage when the command **new-gpiotest help** is used.

In addition, the usage is shown whenever any errors are detected in the parameters.

The usage information displayed is:

```
./new-gpiotest
Usage
       ./new-gpiotest <op> <pin> <val>
       or:
       ./new-gpiotest pwm <pin> <freq> <duty>
       or:
       ./new-gpiotest pwmstop <pin>
Where:
       <op> is one of:
              list - to list info
              set - to set pin value
              get - to get and return pin value
              setd - to set pin direction
              getd - to get and return pin direction
              help - to display usage
       <pin> is one of
              0, 1, 6, 7, 8, 12, 13, 14, 15, 16, 17, 18, 19, 23, 26, all
              A <pin> of all can only be used for an <op> of list, set or
setd
       <val> is only required for set and setd:
              for set, <val> is 0 or 1
              for setd, <val> is in or out
       <freq> is PWM frequency in Hz > 0
       <duty> is PWM duty cycle % in range 0 to 100
```

Notes:

1. The return value from the command will be one of the following:
    - **255** – indicates an error has occurred – either in the parameters or in executing the command
    - **0** – indicates normal successfully completion for an operation (**<op>**) other than **get** or **getd**
    - For a successful **get** operation:
        - **0** – indicates the pin is **off**
        - **1** – indicates the pin is **on**
    - For a successful **getd** operation:
        - **0** – indicates the pin is an **input** pin
        - **1** – indicates the pin is an **output** pin

2. When the **pwm** operation is used, the program forks a separate process to perform the PWM output.
    This separate process continues after the program returns until such time as the **pwmstop** operation is performed on the same pin.
    The ID of the separate process can be discovered by viewing the contents of the file **/tmp/pin<n>_pwm_pid** where **<n>** is the relevant pin number.

6. Further Development

Development of **new-gpio** is on-going. There will be changes and additions to the code in the future.

6.1. Work In Progress

In particular, work is in progress with new-gpio for the following:

- Development of a method of attaching interrupt service code to input pins that will be called when the input state of a pin changes

6.2. For the Future

In addition, it is intended that further work be done in the future based on **new-gpio**. In particular:

- Similar code for **i2c** access
- Java class wrappers that will provide access to GPIO and i2c from Java on the Omega