iotemplate

Version 1.0.0 – 16 June 2016

Kit Bishop

Document History		
Version	Date	Change Details
Version 1.0.0	16 June 2016	First released version

Contents

1. Overview

iotemplate is template C++ program that maybe used as a template that uses any or all of the **libnewgpio**, **libnewi2c** or **libarduino** libraries.

Notes:

- The **iotemplate** library may make use of any of the following libraries:
    - **libnewgpio** library that is available and documented at https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/libnewgpio
    - **libnewi2c** library that is available and documented at https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/libnewi2c
    - **libarduino** library that is available and documented at https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/libarduino

iotemplate consists of single program in static and dynamic forms.

This program is described in more detail in this document.

The program was developed on a KUbuntu-14.04 system running in a VirtualBox VM and uses the OpenWrt toolchain for building the code:

The toolchain used can be found at:

- https://s3-us-west-2.amazonaws.com/onion-cdn/community/openwrt/OpenWrt-Toolchain-ar71xx-generic_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-x86_64.tar.bz2

and details of its setup and usage can be found at:

- https://community.onion.io/topic/9/how-to-install-gcc/22

iotemplate comes with **NO GUARANTEES** ☺ but you are free to use it and do what you want with it.

2. Files Supplied

iotemplate is supplied in files in a GitHub repository at https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/iotemplate. This repository contains the following important directories and files:

- **iotemplate.pdf** – this documentation as a PDF file
- **Makefile** – the Makefile for **iotemplate** program
- **hdr** – directory containing header (***.h**) files for **iotemplate** program
- **src** – directory containing source (***.cpp**) files for **iotemplate** program
- **bin** – directory containing the built program code:
    - **dynamic/< PROGNAME>** – the dynamically linked version of the program, where **<PROGNAME>** is as specified in the **Makefile**
    - **static/< PROGNAME>** – the statically linked version of the program, where **<PROGNAME>** is as specified in the **Makefile**

3. Usage and Installation

Installing and using the program is simple. It primarily consists of linking the program and if needed (see below) the used library code.

3.1. Using iotemplate statically linked program

To use **iotemplate** statically linked program you simply need to copy the program to the Omega and run it.

3.2. Using and Installing iotemplate dynamically linked program

To use **iotemplate** dynamically linked program you need to copy it and any of the libraries **libnewgpio**, **libnewi2c** and **libarduino** that your program uses to your Omega and then run the program.

Any of these libraries that your program makes use of will need the **.so** versions of these libraries to be copied 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 libraries to **/root/lib**
- Added the following lines to my **/etc/profile** file:

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

4. Using Makefile

A **Makefile** is supplied that can be used to build the library.

4.1. Modify Makefile

The **Makefile** will need modifying:

- You **MAY** wish to change **PROGNAME** to specify the name of the final program created.
  The default value is:
  PROGNAME=iotemplate

- You **NEED** to and **MUST** change **TOOL_BIN_DIR** to the "bin" directory of your OpenWrt uClibc toolchain. E.G. make appropriate change to **<xxxx>** in:

  TOOL_BIN_DIR=<xxxx>/OpenWrt-Toolchain-ar71xx-generic_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-x86_64/toolchain-mips_34kc_gcc-4.8-linaro_uClibc-0.9.33.2/bin

- You **MAY** need to change **LIBNEW-GPIO_DIR** to relative directory of libnewgpio if you are not using the sources as originally supplied.
  The default if using the standard **source** directory structure as supplied is:

  LIBNEW-GPIO_DIR=../libnewgpio
- You **MAY** need to change **LIBNEW-I2C_DIR** to relative directory of libnewi2c if you are not using the sources as originally supplied.
  The default if using the standard **source** directory structure as supplied is:

  LIBNEW-GPIO_DIR=../libnewi2c

- You **MAY** need to change **LIBARDUINO_DIR** to relative directory of libarduino if you are not using the sources as originally supplied.
  The default if using the standard **source** directory structure as supplied is:

  LIBARDUINO_DIR=../libarduino

4.2. Makefile targets

The **Makefile** implements the following set of targets:

- **make**
  The default target. Performs a complete build of both static and dynamic link versions of the program.
  This is directly equivalent to:
  make static dynamic

- **make static**
  Performs a complete build of just the static link version of the program.

- **make dynamic**
  Performs a complete build of just the dynamic link version of the program.

- **make clean**
  Removes all previous build files, both static and dynamic link versions.
  This is directly equivalent to:
  make clean-static clean-dynamic

- **make clean-static**
  Removes all previous build files for static link versions only
  .
- **make clean-dynamic**
  Removes all previous build files for dynamic link versions only

If the following is added to the **make** command line:

builddep=1

then **libnewgpio**, **libnewi2c** and **libarduino** libraries that the program depends on will also be built before building the program.

5. Contents of main program for iotemplate

For reference, the contents of the **main.cpp** file supplied with **iotemplate** is:

```
#include "GPIOPin.h"
#include "I2CDevice.h"
#include "ArduinoSystem.h"

int main(int argc, char** argv) {
    // The code for your main program

    // To access an Omega GPIO pin, create an instance of GPIOPin for the pin
number
    // and use methods on that instance
    // e.g.:
    //      GPIOPin * aPin = new GPIOPin(<pin-number>);

    GPIOPin * aPin = new GPIOPin(0);

    // To access an I2C device connected to the Omega, create an instance of
I2CDevice for the i2c address
    // and use methods on that instance
    // e.g.:
    //      I2CDevice * i2cDev = new I2CDevice(<i2c-addr>);

    I2CDevice * i2cDev = new I2CDevice(0x10);

    // To access an Arduino connected to the Omega (via I2C), create an
ArduinoSystem(or ArduinoPort)
    // instance and use methods on that instance
    // Note: will require Arduino library from arduino-omega to be used in a
sketch running
    // on the Arduino
    // e.g. for access to Arduino IO functions on an Arduino using default I2C
address (0x08) and
    // default port number (0) from the Omega, use:

    ArduinoSystem * arduinoSys = new ArduinoSystem();
}
```

6. Further Development

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