General Purpose Input Output LPC1769 Testing

Anahit Sarao
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: indianvip60@gmail.com

Abstract

This project is the beginning of understanding the basics of microcontroller design. The LPC1796 from forth known as LPC was used as a base module. The LPC contains a wide variety of functions, this report focuses upon the General Purpose Input Output known as GPIO and external Vcc functionalities. To achieve a thorough understanding, this project required the use of software suites such as LPCXpresso known as LPCX along with hardware designs for devices such as power regulator, switches, LEDs by combined methods of soldering and wire wrapping. The main challenge faced was designing a future proof PCB layout which allows easy scalability for the future. This was overcome by first bread boarding multiple designs then putting the best implementation on the PCB. The result was a neatly crafted hardware build which utilized software to detect user directed input for enabling or disabling LEDs.

1. Introduction

The LPC uses a Cortex-M3 processor which runs at a max of 120MHz clock speed. The Cortex-M3 breakouts to many ports which can be used for many different embedded applications. To be able to flash software to allow the CPU to delegate and handles the needed instructions sets while being able to debug the code a custom connector CMSIS-DAP is provided by NXP. This allows the board to be flashed using the provided Integrated Development Environment (IDE) LPCX. The project utilizes all of these tools to allow the GPIO ports to work as inputs and outputs. The CMSIS-DAP is configured through a micro-usb which also provides the microcontroller with a steady 5V supply. To be able to use the board untethered from a computer this project additionally adds an independent external power source.

2. Methodology

To provide an elaborate but concise and effective report steps were taken to achieve high quality work. This document is an official representation of all the designs, material, documentation whether provided by the original author or produced by the author stated above.

2.1. Objectives and Technical Challenges

To be able to test GPIO two ports were selected one for input other for output. The input port is the parent node with a child node which contains the output port. The parent is always running checking the signals sent into the input port. Once a signal is detected the child is instructed to either turn on or turn off the corresponding LED. The code was written inside LPCX using the header files provided by NXP which allowed full control of every aspect of the microcontroller.

2.2. Problem Formulation and Design

This being the first project the biggest factors which caused minor to serious problems will not be repeated in future projects. Getting all of the proper components and putting them on a PCB board was a problematic. For example, once the board and power supply are tied in it is very hard to remove parts and relocate them without having to adjust every other component. Finding a proper design that would allow more expansion projects upon the current design was very difficult and it took multiple bread boarding prototypes until a suitable design was found. Another problem was getting the code to communicate with the IDE. Since this board is deprecated and macOS operating system has less support answers were hard to find inside the NXP community support. Once everything is set up and all the components are ordered these problems will not happen in future projects.

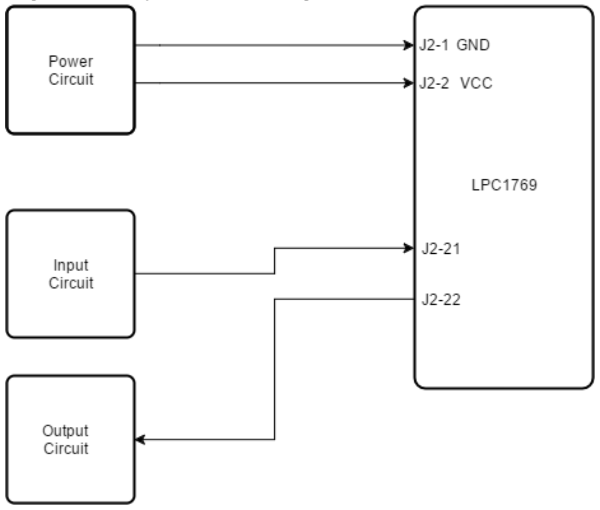
3. Implementation

The following sections provide technical and analytical information for the hardware and software design.

3.1. Hardware Design

Figure one provides an abstraction level block diagram of the major components of the circuit. From left to right the power circuit externally powers the LPC board. Next the input circuit is fed into the input GPIO and a resulting output is fed into the output circuit from the Output GPIO. This is a good way to understand the general flow of signals to and from the board. The project incorporated all of these modules on a prototyping board using wire wrapping and soldering.

Figure 1: Project Block Diagram



Breaking down each module will result in a detailed circuit schematic showing all the hardware used to properly connect the circuits into a software controlled microcontroller unit. The circuit design schematics are provided within the appendix of this report. Each schematic had pin labels and components used. The table below provides a full list of all ports utilized in this project.

The pin rails were not provided by the manufacturer hence male type header pins were soldering onto the board. These pin connections and proper port configuration is important as any incorrect connection can lead to hardware damage.

Table 1: Pin Usage

Name	Pin	Purpose	Port/Pin Location
VCC	J2-1	External Power	EXT_VIN
GND	J2-2	Ground	GND
GPIO	J2-21	GPIO Input	P0.2
GPIO	J2-22	GPIO output	P0.3

Table two shows the total bill of material for this projects. Most of the material was obtained but later on adjusted to ensure the circuit has a better design.

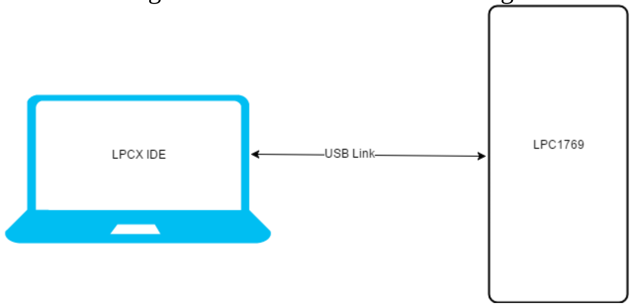
Table 2: Bill of Material

Name	Description
LPC1769 rev D	ARM Cortex CPU Module
Switch x 2	SPDT Switch
Resistor	1K Ohms
9 Volt Adapter	
LM7805	Voltage regulator
Red LED	
Resistor x 3	390 Ohms
Green Led	

3.2. Software Design

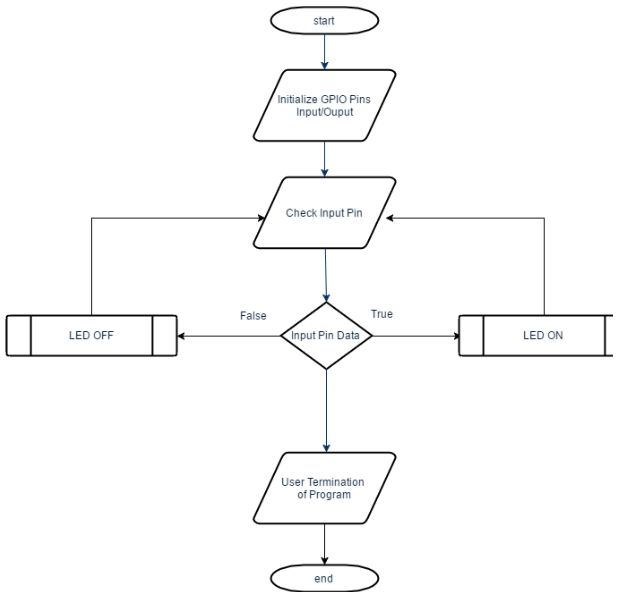
The basis of the software design is to be able to properly utilize the microcontroller. The microcontroller uses a JTAG connection to be able to transfer user written program onto the flash memory. This also allows the user to use the debugging features and many other enhancements such as SWO profiling, memory watching, power management toolkit. Due to OSX operating system environment not being fully supporting only the basic features were working the rest resulted in errors. Figure two shows the basics flowchart of the IDE to board which sustains a bidirectional data connection.

Figure 2: Software Connection Diagram



The software design was very simple once the hardware was constructed. Using the provided CMSIS_CORE_LPC17xx project which contains the needed header and booting configurations to flash and connect to the microcontroller through USB. A new project was started and a library link to the core package was initialized. The software for this project uses the FIODIR register which controls data flow through pins. In addition, register FIOSET was used to set pins to logic high and FOCLR was used to set pins to logic low. The combinational usage of these registers plus looping creates the programing flow showing in figure three below.

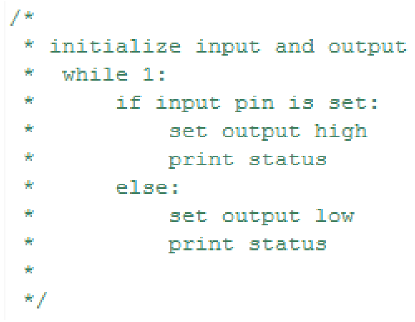
Figure 3: Software Flowchart



The program is never terminated by the microcontroller or the circuits. It is an endless loop only the user can forcefully terminate the program.

For the main function figure four provides a very simple pseudo code representation. The input and output pins are initialized properly. After the pins are configured an endless loop begins to get for the current state of the input pin. If the state is high the output will be set to high if the state if low the output will be set to low.

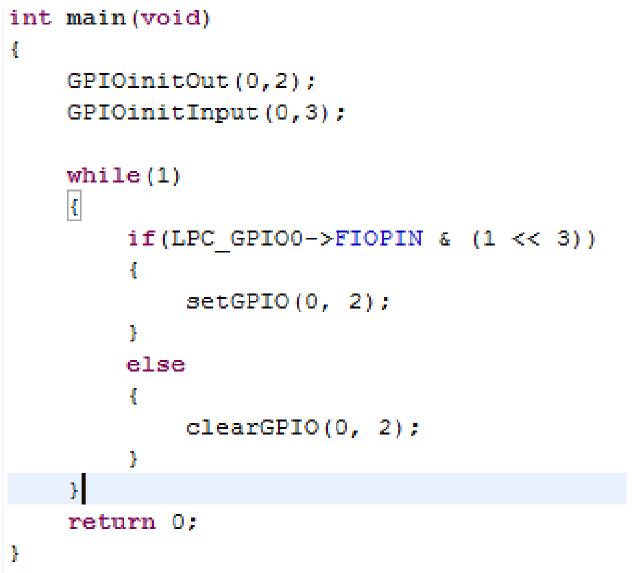
Figure 4: Pseudo Code for Main Function



```
/*
 * initialize input and output
 *  while 1:
 *      if input pin is set:
 *          set output high
 *          print status
 *      else:
 *          set output low
 *          print status
 *
 */
```

The main function utilized the GPIOinitOut() and GPIOinitInput() functions to initialize the input and output pins. These functions take in two parameters which describes the port and pin location on the board. This makes sure the pins being initialized can be utilized for the feature being requested. Figure five provides the algorithm used for this project. After the initialization there is while loop and an if else statement for checking input pin and setting the output pin.

Figure 5: Algorithm Code Block



```
int main(void)
{
    GPIOinitOut(0,2);
    GPIOinitInput(0,3);

    while(1)
    {
        if(LPC_GPIO0->FIOPIN & (1 << 3))
        {
            setGPIO(0, 2);
        }
        else
        {
            clearGPIO(0, 2);
        }
    }
    return 0;
}
```

Full source code is provided within the appendix section of this report.

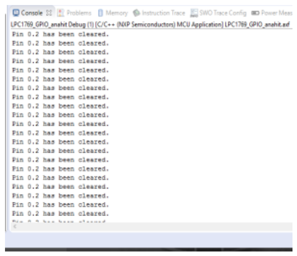
4. Testing and Verification

The steps taken for verifying the hardware and software cohesion are list as:

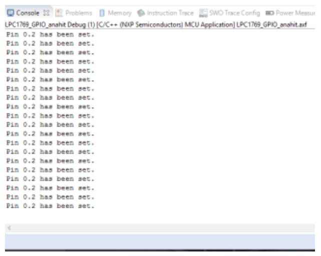
1. Load code onto LPC board
2. Make proper connections
3. Check for short circuit or open circuit
4. Run debugger on IDE
5. Turn on switch to see if board has power by observing green LED
6. Turn on second switch to see if red LED state changes from on to off or vice versa
7. During second switch toggling see if console shows correct state in relation to the red LED

By following these steps, it was verified that the board was programmed and the hardware was working correctly. Figure six and seven show console outputs of the set state and the clear state.

Figure 6: Clear Console Output



```
LPC1769_GPIO_anahit Debug (1) [C/C++ (NXP Semiconductors) MCU Application] LPC1769_GPIO_anahit.axf
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
Pin 0.2 has been cleared.
```



```
LPC1769_GPIO_anahit Debug (1) [C/C++ (NXP Semiconductors) MCU Application] LPC1769_GPIO_anahit.axf
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
Pin 0.2 has been set.
```

Figure 7: Set Console Output

5. Conclusion

In conclusion this project was very successful in helping understand and get familiar with LPCX and the basics of microcontrollers. The board can now be fully run of a 9V battery which is properly regulated and all switch configurations work as expected. The GPIO tests were successful however due to the very fast clock speed the input and output commands could utilize a delay statement. A more advance design would be to add circuit protection, delay statements, and solder the entire circuit to achieve greater stability and continuous performance. These advancements and fixes can be implemented before the next project which builds upon this project. This experience was very positive and educating even though there were many problems such as wires detaching, connections touching other junctions and the board not being fully utilized by the IDE.

6. Acknowledgement

Thanks the professor and the TA for answering questions related to the project.

7. References

[1] H. Li, "LPCXpressoLPC1769revD", CPU Datasheets of CMPE 127, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2017, pp. 1-5.

[2] H. Li, "CMPE127-Microprocessor-Systems", (2017), GitHub Repository, https://github.com/hualili/CMPE127-Microprocessor-Systems

8. Appendix

Figure 8: Overview of front side layout

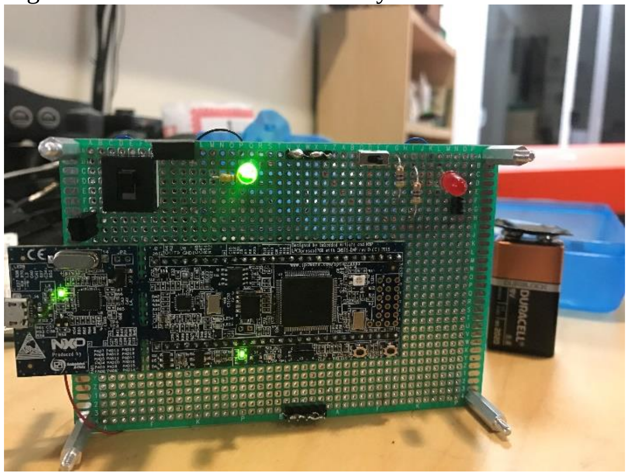


Figure 9: Overview of back side connections

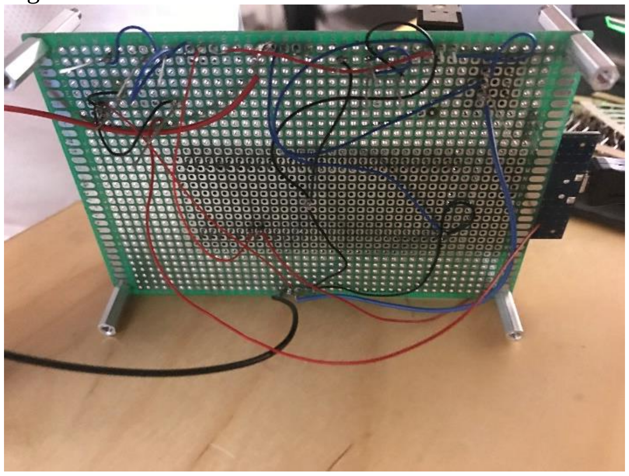


Figure 10: Red LED due to switch 2 toggle (Figure 8 shows switch 2 in its other state and shows red LED off)

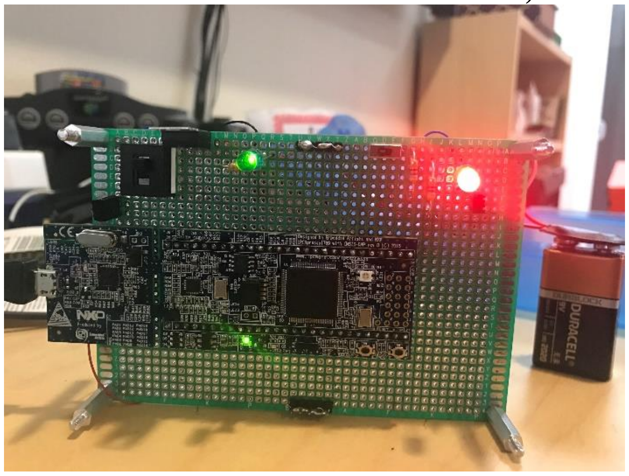
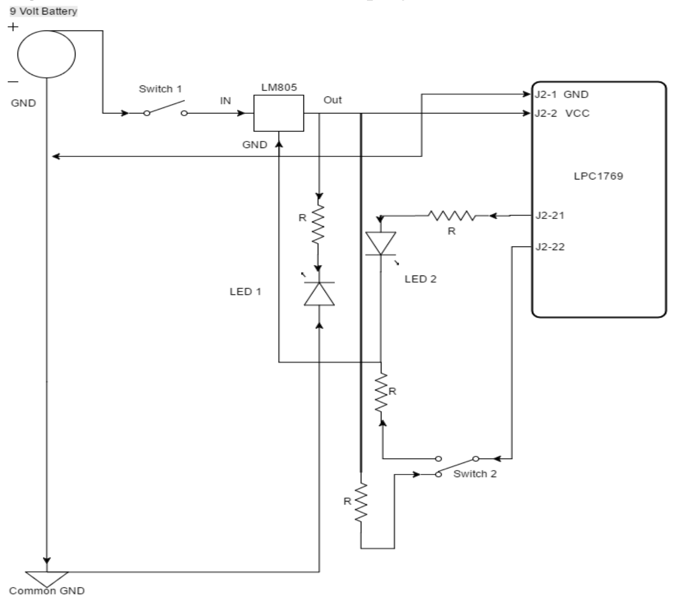


Figure 11: Circuit Schematics for project



Source Code:

```
/*
========================================
========================================
===
 Name        : lab1.c
 Author      : Anahit Sarao
 Version     : 1.0.1
 Copyright   : TeamStandy
 Description : main definition
========================================
========================================
===
 */

#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>
#include <stdio.h>

void GPIOinitOut(uint8_t portNum,
uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIODIR |= (1 <<
pinNum);
	}
	else if (portNum == 1)
	{
		LPC_GPIO1->FIODIR |= (1 <<
pinNum);
	}
	else if (portNum == 2)
	{
		LPC_GPIO2->FIODIR |= (1 <<
pinNum);
	}
	else
	{
		printf("Not a valid
port!\n");
	}
}

void GPIOinitInput(uint8_t portNum,
uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIODIR &= ~(1
<< pinNum);
	}
	else if (portNum == 1)
	{
		LPC_GPIO1->FIODIR &= ~(1
<< pinNum);
	}
	else if (portNum == 2)
	{
		LPC_GPIO2->FIODIR &= ~(1
<< pinNum);
	}
	else
	{
		printf("Not a valid
port!\n");
	}
}

void setGPIO(uint8_t portNum, uint32_t
pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIOSET = (1 <<
pinNum);
		printf("Pin 0.%d has been
set.\n",pinNum);
	}
	else
	{
		printf("Only port 0 is
used, try again!\n");
	}
}

void clearGPIO(uint8_t portNum,
uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIOCLR = (1 <<
pinNum);
		printf("Pin 0.%d has been
cleared.\n", pinNum);
	}
	else
	{
		printf("Only port 0 is
used, try again!\n");
	}
}
/*
 * initialize input and output
 *    while 1:
 *          if input pin is set:
 *                set output high
 *                print status
 *          else:
 *                set output low
 *                print status
 *
 */
int main(void)
{
      GPIOinitOut(0,2);
      GPIOinitInput(0,3);

      while(1)
      {
            if(LPC_GPIO0->FIOPIN & (1
<< 3))
            {
                  setGPIO(0, 2);
            }
            else
            {
                  clearGPIO(0, 2);
            }
      }
      return 0;
}
```