General Purpose Input Output Testing

Nadim Sarras
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: nadim.sarras@gmail.com

Abstract

The purpose of this project centered on using the LPC1769 CPU MODULE to test and program the General Purpose Input/ Output pins. Debugging and working with the functionality of these pins led to a deeper understanding of the role that binary logic plays in every kind of hardware design. This project provided knowledge of the concept of how software can be used to control hardware and manipulate logic values of certain pins. The main technical challenge within the project was discovering and obtaining all necessary, major and minor parts required for the project. However, this challenge was overcome by visiting multiple electronic parts stores, and purchasing components from correlating online retailers. By the end of the project the LPCXpresso was able to successfully demo a bright LED to signify a functional power circuit as well as a toggling LED to signify logic levels of the input/output of the LPC1769 demo board.

1. Introduction

The key component to the entire project is the LPC1769 board. The LPCXpresso module has many features to it as outlined in section one of the references provided by the pin schematics. [1] However the focus of this project was on the GPIO functionality of the module. The GPIO are the General Purpose Input Output pins of the board. These pins can be considered as the versatility of the module as there are almost no restrictions to what can be done with General Purpose Pins. Within this project, the GPIO pins were used to input and output logic values that toggled through a switch, which in turn was used to turn an LED tied to the output, on or off respectively.

2. Methodology

In order to achieve the goal of this project and to achieve full functionality, a great amount of planning occurred before any construction occurred. This section will describe the accumulation of materials, technical challenges, and design of the project.

2.1. Objectives and Technical Challenges

The objective of this project involved programming and being able to use the general purpose input and output pins of the LPCXpresso. In addition, the objective involved being able to program hardware and understand the correlation between voltage levels, logic values, and software written in C code. These are the basics for any engineer who would like to be able to program hardware and work with embedded software.

Furthermore, there were many technical issues involved within the project. The most difficult of which involved obtaining all the necessary minor and major parts in efficient time. This was a problem because certain aspects of the projects could not be worked on until all parts arrived, and some parts did not fit very well into the proto type board.

2.2. Problem Formulation and Design

Due to the fact that there were many question marks pertaining to parts and how everything would mesh together it was concluded that having a design of the circuit would assist with any confusion. The design involved many components such as a block diagram, circuit schematics, and a flow chart for the software. The block diagram was used to visualize the overall layout and function of the project. While the circuit schematics provided the detailed description of how to construct all of the circuits within the project. The flowchart consisted of the necessary components to consider when creating the code for the LPCXpresso board. In addition, special accommodations had to be made based on the parts decided for the circuit. For example, the dip switch did not have two separate pins for two inputs, therefore both the inputs were soldered together and attached to the dip switch pin. As for the dc power jack and standoffs, holes had to be drilled into the proto type board in order to plant those components on the board.

3. Implementation

This section will go over the steps taken as well as the thought process behind the hardware and software design of the project.

3.1. Hardware Design

In figure one below is the block diagram of the circuit. It is very simple but very efficient at showing what the major functions of the project should incorporate. Both the power and switch circuit are served as inputs to the LPC

board. Resulting in an output circuit which will toggle the LED ON/OFF depending on the LED.

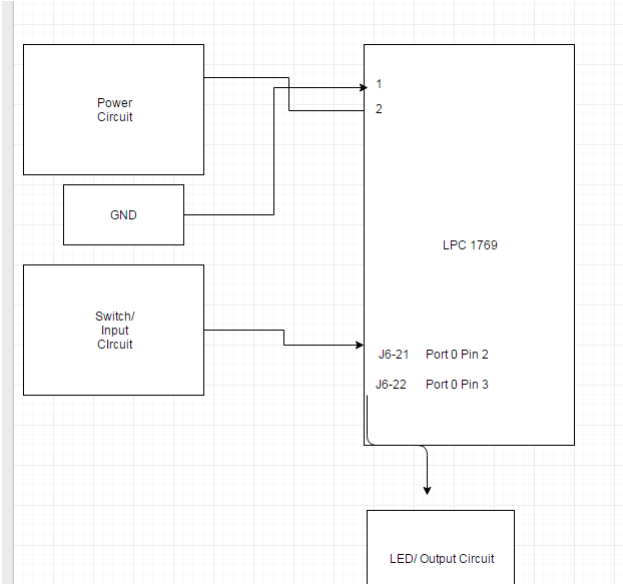


Figure 1: LPCXpresso Block Diagram

Furthermore the circuit schematic provided the highest amount of detail when constructing the circuit as the schematic showed where the transistors and wires were specifically connected within each component.

The Circuit Design Schematic can be found in the Appendix. The schematic provides a detailed visual of where the wire connections were made as well as what the values of the resistors are. The schematic also includes what pins were used from the LPC 1769 board.

In addition the pins used where a critical part of this project as selecting the wrong pin could be disastrous. It is also one of the objectives of this project to fully understand and be able to use the pins on the LPCXpresso board.

Table 1: Pin Usage

Functional Name	Pin Number	Purpose	CPU Model LPC 1769 Port Num / Pin
GND	J6-1	Ground	GNDX
VIN	J6-2	Power Supply	EXT_POWX
GPP	J6-21	Input pin	Port 0/ Pin 2
GPP	J6-22	Output Pin	Port 0/ Pin 3

Table 2: Bill of Materials

Part	Function
LM7805C	Voltage Regulator
LPC1769CD	CPU Module
4x Stand Offs	General
4 Strips Male Header Pins	General
4 1kΩ Resistors	Circuit Component
2 330Ω Resistors	Circuit Component
Soldering Kit	Solder
2 LED	Circuit Component
Power Supply, 9v/2A	Power Circuit
USB Cable	Loading Program/ Debugging
Female DC Jack	Power Circuit
Dip Switch	Provides up to 8 working switches for circuit

3.2. Software Design

The software design process of this project consisted of understanding and implementing the registers within the LPC 1769 board. These registers included the FIODIR register which dictates the flow of the data through a pin, for example FIODIR will tell the register whether to send or receive data. FIOPIN was used to gather the actual logic value of the pin and to determine if there is either an output or input. FIOSET was used to logically set the value of the pin to high. Therefore the pin would input or output a voltage level, which was eventually used to power the LED. FIOCLR was used to set the pin values to low, thus clearing or turning off any leading LEDS.

The flowchart shown in figure 2 displays the first critical thinking steps in the software design.

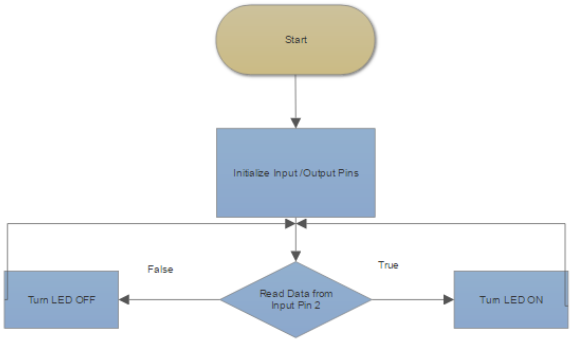


Figure 2: Flowchart Design

The software process begins with the initial switch. Depending on the value of the switch the general purpose pin 2 will input a value of 0 or 1. That value is then run through the cpu module and outputted through port 0, pin 3 of the board. The output will be 0 if the input was 0 and the output will be 1 if the input was 1. Therefore the input drives the output. If pin 3 outputs a 1, it is treated as a high the leading LED will light up. However if the LED is already on and the input changes to 0, then the output will respectively change to 0, resulting in a low output, thus turning off the LED. The entire command should be run in a while loop so the user may toggle the switch on and off.

A Psuedo code interpretation of the following algorithm would be as follows; first initialize inputs and outputs, check the value of the input, and set output accordingly. The resulting output will either turn the connected LED on or off.

```
//Initialize Input /Outputs
//while(1){
//  check input
//  if(input is high)
//  {
//      make output high
//  }
//  else{
//      make output low
//  }
//}
```

Figure 3: Psuedo Code

```
int check=0;
GPIOinitIn(0,2);
GPIOinitOut(0,3);
while(1){

    check = readGPIO(0,2);
    if(check==1){
        setGPIO(0,3);
        printf("GPIO input = %d", check);
        printf("The LED is on \n");
    }
    else{
        clearGPIO(0,3);
        printf("GPIO input =%d",check);
        printf("The LED is off \n");
    }

}
```

Figure 4: Algorithm Implementation

Here the Algorithm for programming the LPCXpresso board is shown. Both the input and output pins are initialized in order to be able to read a stable value. This is done by shifting the register by the pin number and & one. The & one will make the resulting number a zero or one, making it easy to use for if statements. As shown in the figure below.

```
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
    {
        puts("Not a valid port!\n");
    }
}
```

Figure 4: Algorithm

4. Testing and Verification

The testing procedure involved the following steps;

1. Make Sure code is debugged and loaded into board
2. All connections are tight and there are no loose wires
3. Run the debugger and load the program in the board
4. Turn on the first switch and observe the red LED to see if Power Circuit is on
5. Turn on the Second Switch and observe the yellow LED to see if the LED receives the output from the cpu module
6. Toggle the 2nd switch (Switch 8 on the dip switch) in order to observe the yellow LED turn on and off
7. Meanwhile pay attention to the console input values and printf statements of whether the yellow LED is on or off.

A screenshot of the console output is shown below.

```
Pin 0.3 has been cleared.
GPIO input =0The LED is off
Pin 0.3 has been cleared.
GPIO input =0The LED is off
Pin 0.3 has been cleared.
GPIO input =0The LED is off
Pin 0.3 has been set.
GPIO input = 1The LED is on
Pin 0.3 has been set.
GPIO input = 1The LED is on
Pin 0.3 has been set.
GPIO input = 1The LED is on
Pin 0.3 has been set.
GPIO input = 1The LED is on
Pin 0.3 has been set.
GPIO input = 1The LED is on
```

Figure 5: Console Output

The project can also be verified by using a multi meter to measure the voltage coming in and out of the GPIO pins and comparing them to the expected values of the console output (in logice terms of 0 for low, 0 volts and 1 for high, 3.3+ volts).

5. Conclusion

In conclusion, this lab was extremely effective in learning how to use general purpose registers and being able to program them effectively. This project also provided great exposure to all the glue logic parts, as well as techniques including wire wrapping and soldering. Understanding how to program the cpu module and the directions of the pins was another big factor of this experiments. Overall, I had a tough but positive experience from this project.

6. Acknowledgement

I would like to acknowledge the TA's for helping me set up the IDE on my board for the first time as there were technical issues.

7. References

[1] [1] H. Li, "LPCXpressoLPC1769revB", *CPU Datasheets of CMPE 127*, Computer Engineering Department, College of Engineering, San Jose State University, March 1, 2016, pp. 7.

8. Appendix

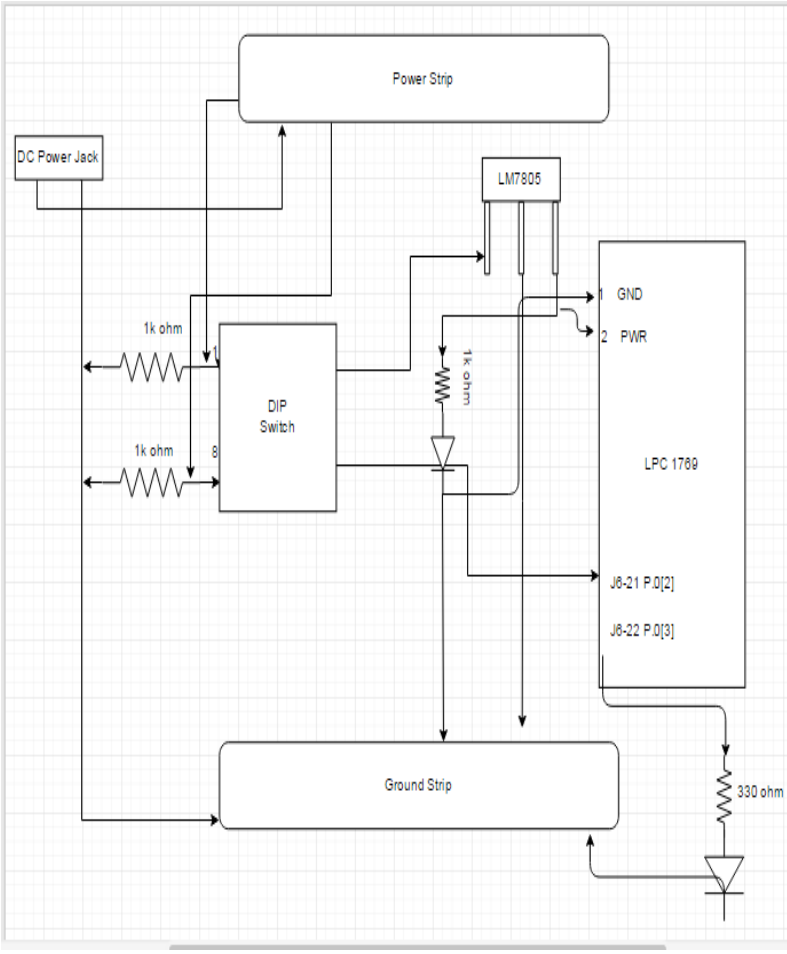


Figure 6: Full Schematic

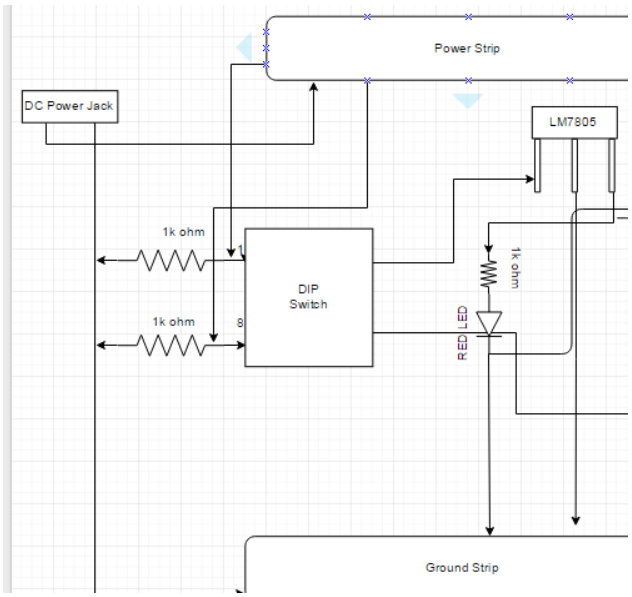


Figure 7: Power Circuit Schematic

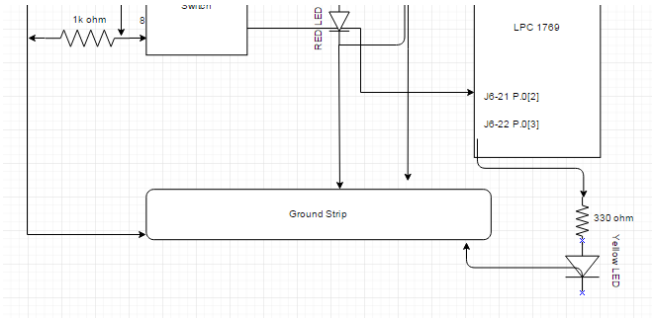
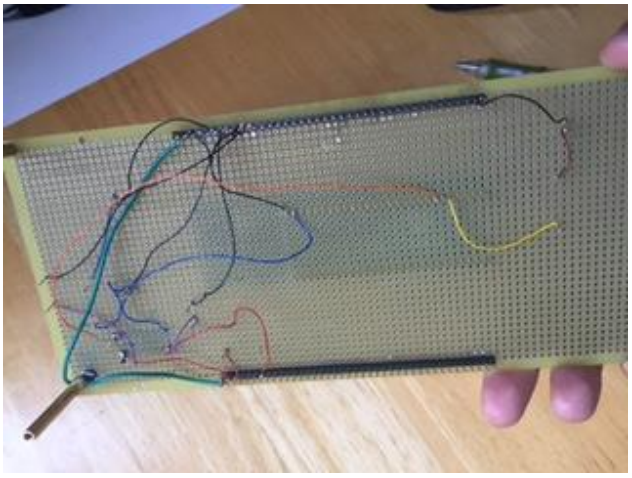
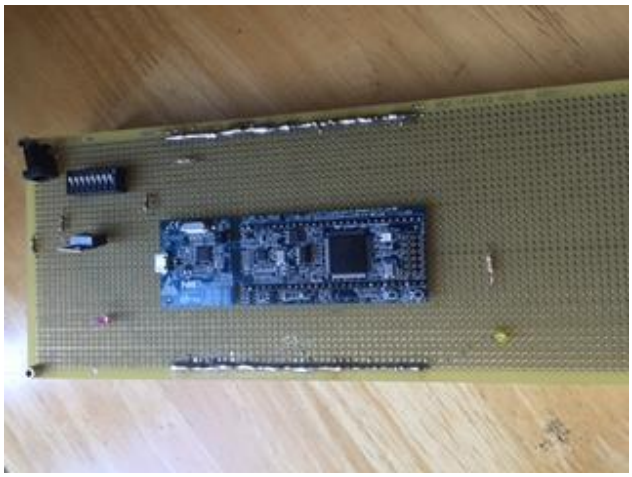
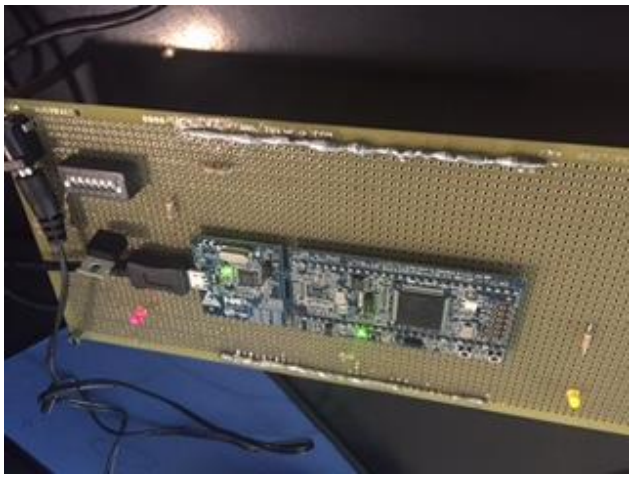


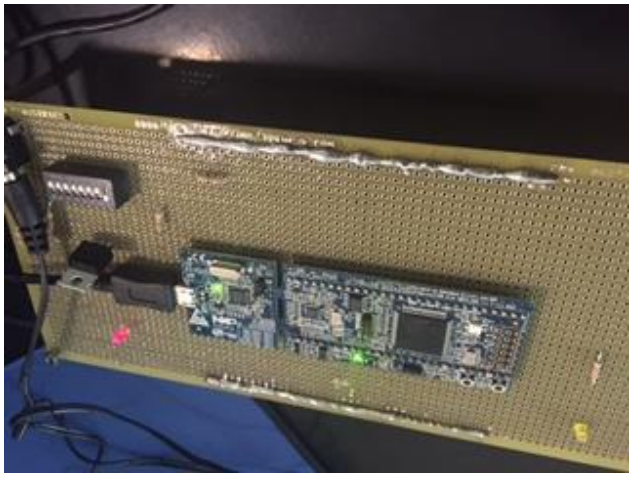
Figure 8: Input/Output Schematic

Pictures #1-4









Attachment #1 Actual Code

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

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

//Initialize the port and pin as outputs.
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
    {
        puts("Not a valid port!\n");
    }
}
void GPIOinitIn(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
    {
        puts("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);
    }
    //Can be used to set pins on other ports for future modification
    else
    {
        puts("Only port 0 is used, try again!\n");
    }
}

//Deactivate the pin
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);
    }
    //Can be used to clear pins on other ports for future modification
    else
    {
        puts("Only port 0 is used, try again!\n");
    }
}

int readGPIO(uint8_t portNum, uint32_t pinNum)
{
    if(portNum == 0)
    {
        if((LPC_GPIO0 -> FIOPIN & (1<<pinNum)))
            return 1;
        else
            return 0;
    }
    else
    {
        printf("Only port 0 is used, try again!\n");
        return 0;
    }
}

int readGPIO(uint8_t portNum, uint32_t pinNum)
{
    if(portNum == 0)
    {
        if((LPC_GPIO0 -> FIOPIN & (1<<pinNum)))
            return 1;
        else
            return 0;
    }
    else
    {
        printf("Only port 0 is used, try again!\n");
        return 0;
    }
}

int main(void)
{
    int check=0;
    GPIOinitIn(0,2);
    GPIOinitOut(0,3);
    while(1){

        check = readGPIO(0,2);
        if(check==1){
            setGPIO(0,3);
            printf("GPIO input = %d", check);
            printf("The LED is on \n");
        }
        else{
            clearGPIO(0,3);
            printf("GPIO input =%d",check);
            printf("The LED is off \n");
        }

    }
```