LP1769 GPIO Interface Testing

Divya Khandelwal, MS,
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: divya.khandelwal@sjsu.edu

Abstract

In this project, we have developed a prototype to test the General-Purpose Port interface of the LPC1769 module. The aim is to toggle an LED connected to an output port of LPC1769 by changing the state of a switch, connected at the input port. The key in this project is to correctly read and write to I/O pins of LPC1769.

1. Introduction

The LPC1769 is a Cortex-M3 microcontroller for embedded applications featuring a high level of integration and low power consumption at frequencies of 120MHz [1]. Along with numerous other features, this module has 70 General Purpose I/O (GPIO) pins with configurable pull up/down resistors.

For this project, we have created a prototype circuit board using LPC1769 module. We have used MCUXpresso from NXP as the IDE to work with this module using C programming language.

2. Methodology

In this section, system layout and its hardware and software configurations are discussed. In addition, the objectives for this project have been listed with the challenges faced during implementation.

2.1. Objectives and Technical Challenges

The objectives of this project are as follows:

1. To be able to create a prototype circuit with LPC1769 module, power-supply and I/O testing circuits.
2. To gain hands on experience with MCUXpresso environment, its debugging tools and C programming language.

The technical challenges faced are as follows:

1. Soldering the components on the wire-wrapping board with proper pin connections.
2. Correctly using the GPIO registers in the C program, to configure the I/O pins of the module.

2.2. Problem Formulation and Design

In this section, we discuss the system design for the project. Fig.1 shows the system block diagram.

Implementation

In this section the hardware and software design of the project is described.

3.1. Hardware Design

Fig 2. shows the input and output testing circuits which are used to test the GPP interface of the LPC1769.

We have configured P0.3 as input port pin using a pull-down resistor circuit and P0.3 as the output port circuit P0.21.

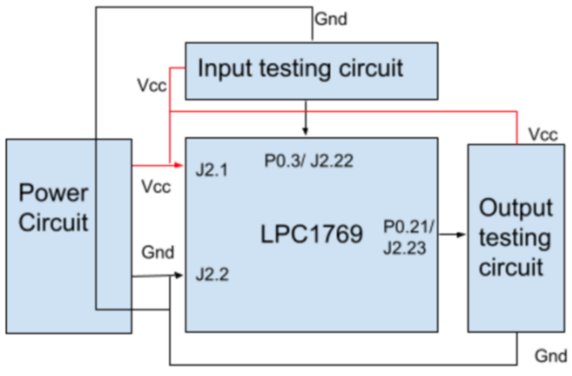

Figure 2. Flow chart of the system

3.2. Software Design

The main purpose of the circuit is to test GPP interface using an input switch and output LED. State diagram in Fig.3 explains the working.

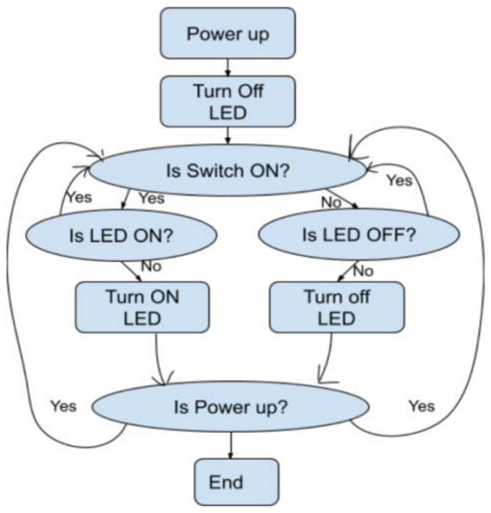

Figure 3. Flow chart of the system

Some GPIO Registers are needed to be configured appropriately to perform each of the action mentioned in the above flow chart.

1. LPIO_GPOI -> FIODIR = Direction of the I/O (Input pin =0 / Output pin = 1)
2. LPIO_GPIO -> FIOPIN = It reads value at the input pin (Switch state =1 or 0)

3. LPIO_GPIO ->FIOSET = It sets the output pin (LED = ON)
4. LPIO_GPIO -> FIOCLR = It clears the output pin (LED = OFF)

   Fig.4 show the state diagram which depicts how the registers are employed in the entire functionality.

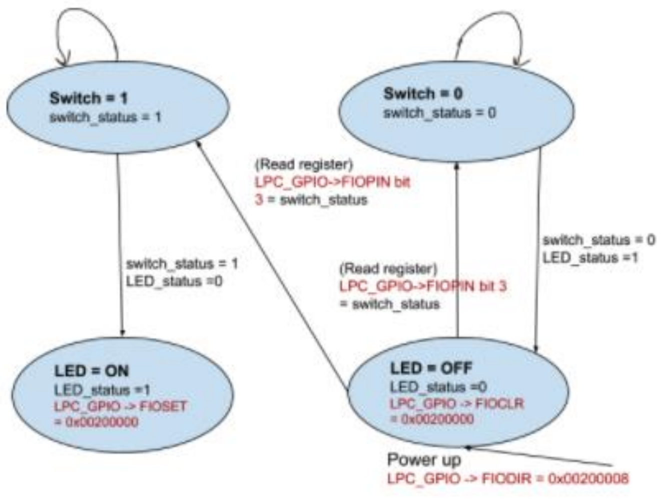

Figure 4. State diagram of the system

5. Testing and Verification

The testing and verification is done to make sure that actual results are same as predicted results.

5.1 Testing

1. Make sure the Power plug is connected to the right-angle power connector.
2. Check for the power circuit switch to be in ON state.
3. Toggle the input switch to check the blinking output LED.
4. Make sure the program is flashed in the CPU module by either CPU Flash tool or using the USB probe and debug mode.

The Fig.5 and Fig.6 show the prototype board, one where output LED is glowing and one where it is off. It depends on the state of the switch of the input circuit (seen right to the CPU module).

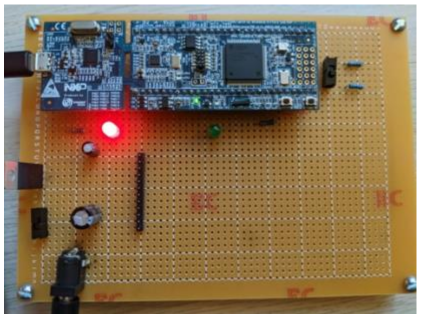

Figure 5. Output LED (green LED) off

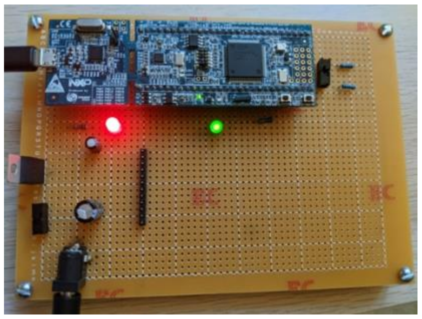

Figure 6. Output LED (green LED) ON

5. Conclusion

Pin P0.3 and Pin P0.21 are successfully configured as input and output pins. The LED connected at P0.21 successfully toggles with the state of switch placed at P0.3.

6. Acknowledgement

This project was successfully implemented under the mentorship of Dr. Harry Li, Department of Computer Engineering, San Jose State University, CA.

7. References

[1] H. Li, “2018F-107-lecGPP-2018-9-10”, *Lecture Notes of CMPE 242*, Computer Engineering Dept., College of Engineering, San Jose State University, September 10, 2018 pp. 1.
[2] NXP Inc., “LPC 1769/68/67/66/65/64/63 data sheet”, Revision 9.10, September 8, 2020

8. Appendix

Below is the source code of the C program for GPP interface testing

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

#include <cr_section_macros.h>

#include <stdio.h>

#define INPUT_PORT_NUMBER 0
#define INPUT_PORT_PIN 3
#define OUTPUT_PORT_NUMBER 0
#define OUTPUT_PORT_PIN 21

/*Print the details of student
 * */
void printDetails();

/*Initialize the port and pin as inputs.
 * @param Port Number of the LPC1769 which we want to
use as input (1 port = 32 pins i.e 32 bits)
```

```
 * @param pin Number of the port selected that we want to
use as input
 * @return null
 * */
void GPIOinitInput(uint8_t portNum, uint32_t pinNum);

/*Initialize the port and pin as outputs.
 * @param Port Number of the LPC1769 which we want to
use as output (1 port = 32 pins i.e 32 bits)
 * @param pin Number of the port selected that we want to
use as output
 * @return null
 * */
void GPIOinitOut(uint8_t portNum, uint32_t pinNum);

/*Read the GPIO pin.
 * @param Port Number of the LPC1769 which we are
using as input.
 * @param pin Number of the selected input port
 * @return status of the input pin - 1 or 0
 * */
uint32_t readGPIO(uint8_t portNum, uint32_t pinNum);

/*Set the GPIO output pin.
 * @param Port Number of the LPC1769 which we are
using as output.
 * @param pin Number of the selected output port
 * @return null
 * */
void setGPIO(uint8_t portNum, uint32_t pinNum);

/*Clear the GPIO output pin.
 * @param Port Number of the LPC1769 which we are
using as output.
 * @param pin Number of the selected output port
 * @return null
 * */
void clearGPIO(uint8_t portNum, uint32_t pinNum);

int main()
{
        printDetails();
        //state of switch and LED in input and output
testing circuit
        uint32_t switch_status, led_status = 0 ;

        //Set pin
INPUT_PORT_NUMBER.INPUT_PORT_PIN as input
        GPIOinitInput(INPUT_PORT_NUMBER,
INPUT_PORT_PIN);
    //Set pin
OUTPUT_PORT_NUMBER.OUTPUT_PORT_PIN as
output
        GPIOinitOut(OUTPUT_PORT_NUMBER,OUT
PUT_PORT_PIN);

        //When powered up, LED is OFF
        clearGPIO(OUTPUT_PORT_NUMBER,
OUTPUT_PORT_PIN);

    while(1)
    {

        //Make
INPUT_PORT_NUMBER.INPUT_PORT_PIN as input
        switch_status =
readGPIO(INPUT_PORT_NUMBER,
INPUT_PORT_PIN);

                if (switch_status == 1 && switch_status
!= led_status)
                {
                        //Activate pin 0.21

        setGPIO(OUTPUT_PORT_NUMBER,OUTPUT
_PORT_PIN);

                }

                else if( switch_status == 0 &&
switch_status != led_status)
                {
                        //Deactivate pin 0.21

        clearGPIO(OUTPUT_PORT_NUMBER,
OUTPUT_PORT_PIN);

                }
                else if(switch_status !=0 &&
switch_status !=1)
                {
                        puts("Not a valid switch
state!\n");
                }
                led_status = switch_status;
    }

    return 0;
}

void printDetails()
{
        printf("Hello, the World\n");
        printf("Divya Khandelwal\n");
        printf("SID: 6885\n");
}

void GPIOinitInput(uint8_t portNum, uint32_t pinNum)
{

        if (portNum == 0)
        {
                // Set the pin number as 0 by left shift
operation of 0.....00000000 so that 0 comes at the position
of pinNum
                // Bitwise Or it with the Register
```

```
		LPC_GPIO0->FIODIR |= (0 <<
pinNum);
	}
	else if (portNum == 1)
	{
		LPC_GPIO1->FIODIR |= (0 <<
pinNum);
	}
	else if (portNum == 2)
	{
		LPC_GPIO2->FIODIR |= (0 <<
pinNum);
	}
	else
	{
		puts("Not a valid port!\n");
	}
}


void GPIOinitOut(uint8_t portNum, uint32_t pinNum)
{

	if (portNum == 0)
	{
		// Set the pin number as 1 by left shift
operation of 0.....00000001 so that 1 comes at the position
of pinNum
		// Bitwise Or it with the Register
		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");
	}
}


uint32_t readGPIO(uint8_t portNum, uint32_t pinNum)
{
	if (portNum == 0)
	{
		return (LPC_GPIO0->FIOPIN >>
pinNum) & 1 ;
	}
	else if(portNum == 1)
	{
		return (LPC_GPIO1->FIOPIN >>
pinNum) & 1 ;
	}
	else if(portNum == 2)
	{
		return (LPC_GPIO2->FIOPIN >>
pinNum) & 1 ;
	}
	else
	{
		puts("Not a valid port!\n");
	}
	return -1;
}


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");
	}
}
```