**Migration from Arduino to LPC1769 for PWM and GPIO**

Nicholas Foulk & Kevin Gadek

*Computer Engineering Department, College of Engineering*

*San Jose State University, San Jose, CA 94303*

*E-mail:* [nickfoulksjsu@gmail.com](mailto:nickfoulksjsu@gmail.com) *, kgadek@sbcglobal.net*

**Abstract**

*The objective of this project was to migrate the PWM and GPIO circuits from the Arduino to the LPC 1769 environment. The PWM and GPIO circuits are used for the proxy motor control circuit. The LPC1769 environment allows for a more accurate duty cycle configuration and better function declaration as opposed to the Arduino. Transitioning between 5V and 3.3V is one of the main challenges faced.*

**1. Introduction**

The LPC1769 module schematic was used to find the available pins for PWM and GPIO signals. Afterwards the LPC1769 user manual was reviewed for a list of available PWM and GPIO functions. Only 5V pins were used in the Arduino environment, so the duty cycle or frequency has to be altered for an equivalent performance.

**2. Methodology**

The objectives and technical challenges will include the overall outcome and problems faced in this project. The problem formulation and design will break down the process of identifying the problem and planning out a solution.

**2.1. Objectives and Technical Challenges**

The objective of this project was to migrate the pulse width modulation and general purpose peripherals from the Arduino to LPC1769 environment. The objective entails building a schematic and block diagram, writing a new program, and wire wrapping between a prototype and bread board.

**2.2. Problem Formulation and Design**

The problem proposed for setting up the prototype board with a PWM and GPIO circuit was to provide 5V supply and a modifiable duty cycle function:

1. In order to provide the TTL Dual in-line packages with 5 volts, the wall mount of on the prototype board had to be used for the time being. A logic level shifter will be used in later revisions.
2. The biggest problems being faced when using TTL DIP is that a bread board is required for mounting and one side of the wiring has to be wire wrapped because of the LPC 1769. Female to male wires had to be used in order to transition from prototype to bread board. Later revisions will include a different digital multiplexer.

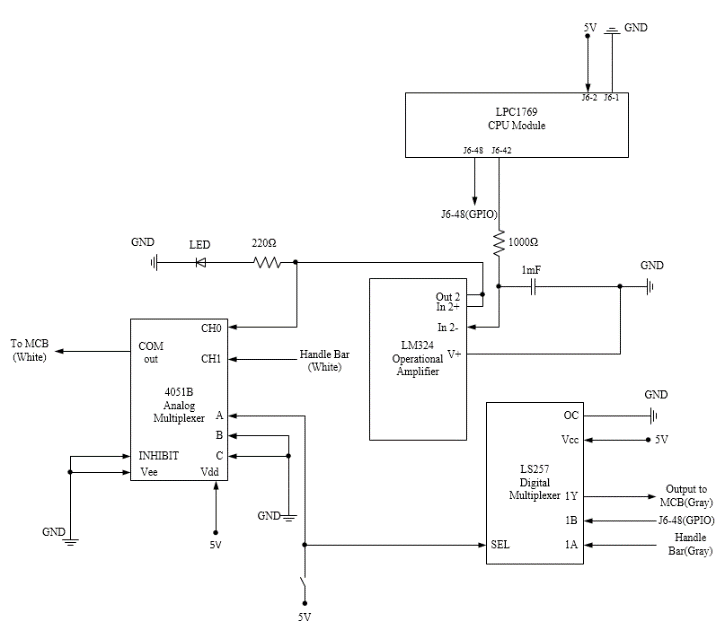


Figure 1: Schematic for the PWM and GPIO design

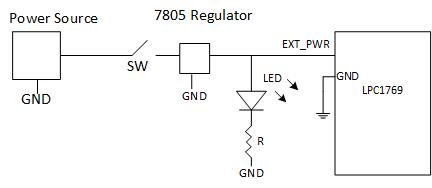


Figure 2: Power supply schematic for 5V external power

**3. Implementation**

This section provides the hardware design, and software design to implement the PWM, GPIO, and power circuit in this project. The hardware design includes a system block diagram, a schematic design, and a list of components used. The software design includes flow charts, algorithms, and pseudo code for describing the process of designing the coded program.

**3.1. Hardware Design**

The design of each system block diagram and schematic design represented in Problem Formulation and Design is explained in this section. This section also includes a Bill of Material used for the project.

1. Figure 1 includes a power supply to 5V circuit and LPC1769 to PWM and GPIO circuit. The power supply circuit has a resistor and capacitor for a more stable supply from the LM7805 voltage regulator.
2. The overview picture in figure 2 shows the use of the LPC module being connected to the LM7805 for an initial power supply. There are also a few connections from the LPC module to the RC filter or PWM circuit with all of the TTL devices used afterwards, including an operational amplifier and digital multiplexer.
3. The schematic design represented in figure 3 shows the entire prototype board showing the importance of the Vcc and ground connections of the all the different peripherals.
4. For a Bill of Materials see Appendix B, Table I.

**3.2. Software Design**

The software design section provides flow charts to show the implementations of software. This section contains discussion of the algorithms and pseudo code of the project to get a better understanding of the program designed.

The algorithm design for the program is as follows:

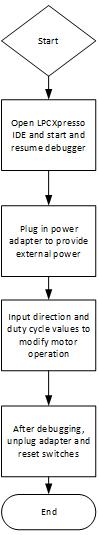
1. Initialize the GPIO circuit make variables for user input.
2. Start an infinite while loop to continually run the PWM function.
3. Have user input different duty cycles and direction values into console.
4. Call and initialize Init\_PWM function from user arguments to modify duty cycle and direction.

Figure 3: Flow chart for the basic flow level of implementation of the program

The pseudo code for the program is as follows:

Int main()

{

Initialize GPIO

Declare 32 bit variables

While loop (1)

{

Ask user for Direction

Scan Direction

Ask user for duty cycle

Scan duty Cycle

If (cycle = 0)

{

Init\_PWM(0)

}

If (cycle = 1)

{

Init\_PWM(1)

}

If (cycle = 2)

{

Init\_PWM(2)

}

If (cycle = 3)

{

Init\_PWM(3)

}

}

Return 0

}

int init\_PWM(int value)

{

Turn on peripheral power for PWM pin;

Select peripheral clock for PWM cclk/8;

Select P2.0 as PWM1.1 operation;

Reset prescale counter and TCR register;

Set MR0 register to 1000;

Set MR1 register to value to set duty cycle;

Latch MR0 and MR1 to enable write for match registers;

Enable PWM output;

Enable TC and PWM;

}

init\_GPIO()

{

Set P2.6 as GPIO output;

}

void set\_dir(uint32\_t dir)

{

if(dir)

Set P2.6 as HI;

else

Set P2.6 as LO;

}

**4. Testing and Verification**

In order to test the program's ability to modify the direction and speed of provided motor, the GPIO output is fed directly to the motor to provide for direction control and PWM signal is fed through a unity gain and RC filter circuit to the motor to allow for speed control. To allow for easy testing of the program's functionality, a feedback program was created that would continually prompt a user to input new values for direction and duty cycle and verify the successful modification of the motor's operation.

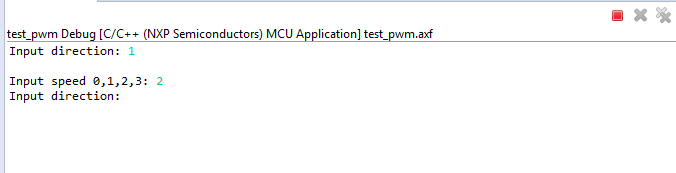


Figure 5: LPCXpresso console output

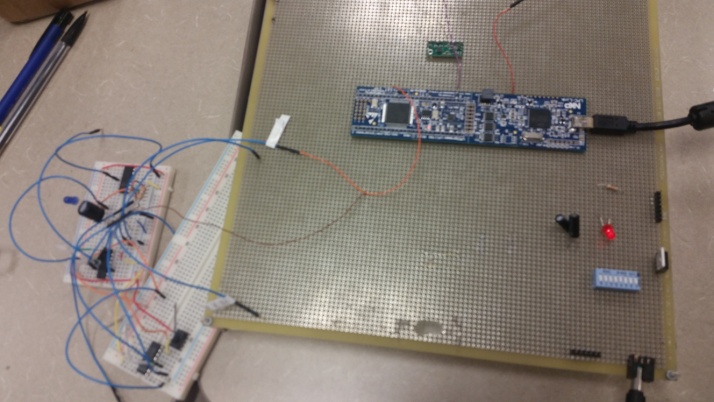


Figure 6: Picture of prototype board

**5. Conclusion**

The resistor and capacitors were connected to pin 42 of the LPC 1769 Expresso board and connected to the negative input of the operational amplifier. The output was connected to the positive input for following and the first channel of the analog multiplexer. The generated PWM signal is used to mimic the characteristics of the white wire handle bar analog signal.

**6. Acknowledgement**

Dr. Harry Li provided support to help define the problem this project presented. Kevin found the available LPC1769 API and set up an initialize function for the PWM circuit.

**7. References**

[1] NXP Semiconductors, LPCXpresso1769 datasheet

[2] Fairchild, LS257 datasheet

[3] Texas Instruments, LM324 datasheet

[4] Texas Instruments, 4051B datasheet

**8. Appendix**

Table I: Bill of Materials

|  |  |  |
| --- | --- | --- |
| Item # | Description | Notes |
| 1 | Wall Mount |  |
| 2 | LM7805ct | +7V in/5V out |
| 3 | Red LED |  |
| 4 | DIP switch | 2nd for LM 4th for GPIO circuit |
| 5 | LPC1769 | CPU Module |
| 6 | Resistors | 3(150,270,10k) |
| 7 | Capacitors | 210µF |
| 8 | Wire Wrapping Tool |  |
| 9 | Solder |  |
| 10 | 28 Gauge Wire |  |
| 11 | Motor | For testing purposes |
| 12 | LM324 | Op-Amplifer |
| 13 | 4051B | Analog mux |
| 14 | LS257 | Digital mux |

Table II: Pin Layout for LPC 1769

|  |  |  |
| --- | --- | --- |
| Pin | Description | Notes |
| J6-1 | Ground |  |
| J6-21 | Output |  |
| J6-42 | PWM1.1 | PWM output |
| J6-48 | P2.6 | GPIO Dir. Ctrl. |

Source Code:

/\*

===============================================================================

Name : test\_pwm.c

Author : $(author)

Version :

Copyright : $(copyright)

Description : main definition

===============================================================================

\*/

#ifdef \_\_USE\_CMSIS

#include "LPC17xx.h"

#endif

#include <cr\_section\_macros.h>

#include <stdio.h>

void init\_PWM(uint32\_t PWMinval);

void init\_GPIO();

void set\_DIR(uint32\_t dir);

int main(void) {

uint32\_t dir, cycle;

init\_GPIO();

while(1){

printf("Input direction: ");

scanf("%i", &dir);

printf("\nInput speed 0,1,2,3: ");

scanf("%x", &cycle);

set\_DIR(dir);

if(cycle == 0)

{

init\_PWM(0);

}

if(cycle == 1)

{

init\_PWM(250);

}

if(cycle == 2)

{

init\_PWM(500);

}

if (cycle == 3)

{

init\_PWM(900);

}

}

return 0;

}

void init\_PWM(uint32\_t PWMinval)

{

//Power for PWM1, sets 6th bit of PCONP register to 1 which is PWM1;

//PCONP register is peripheral power control register

LPC\_SC->PCONP |= (1 << 6);

//peripheral clock select for PWM cclk/8

LPC\_SC->PCLKSEL0 |= ((1 << 13) | (1 << 12));

//pin select

LPC\_PINCON->PINSEL4 |= (0x1<<0); //sets 01 for bits [1:0] of PINSEL4 register, which sets P2.0 to PWM1.1 operation

LPC\_PWM1->TCR = (1<<1); //counter reset, set 2nd bit of TCR register to 1;

LPC\_PWM1->PR = 0; //count frequency, prescale register

LPC\_PWM1->PC = 0; //prescale counter

LPC\_PWM1->MCR = (1 << 1); //reset TC on Match 0

//set period & duty cycle

LPC\_PWM1->MR0 = 1000; //set PWM period/cycle to 1khz

LPC\_PWM1->MR1 = PWMinval; //set 50% duty cycle; period/2

//write enable for match registers

LPC\_PWM1->LER = (1<<0)|(1<<1); //latch MR0 and MR1 (must be used for those registers to be overwritten)

//pwm enable, settings

LPC\_PWM1->PCR = (1<<9); //PWM output enable, single-edged operation, must be set else otherwise PWM is a counter

LPC\_PWM1->TCR = (1<<0) | (1<<3); //TC enable, PWM enable

}

void init\_GPIO()

{

LPC\_GPIO2->FIODIR |= (1 << 6); //set P2.6 as GPIO output

}

void set\_DIR(uint32\_t dir)

{

if(dir)

LPC\_GPIO2->FIOSET = (1 << 6); //sets P2.6 as HI

else

LPC\_GPIO2->FIOCLR = (1 << 6); //sets P2.6 as LO

}