EE 128
Lab 2: Simple Parallel I/O

Leya Zeng 862017097
Tanish Arora 862012234

Section 021
TA: Mohsen Karimi

Abstract

This lab is become familiar with development environment for 9S12DG256-based digital system, its hardware and software techniques for developing, testing and debugging microcontroller-based digital system. Using C programming to control the behavior of 9S12DG256-based digital system.

Lab References

- Dragon12-JR Trainer User's Manual
- Freescale MC9S12DG256 Device User Guide
- Freescale HCS12 Core User Guide
- Reference Guide for D–Bug12 Version 4.x.x

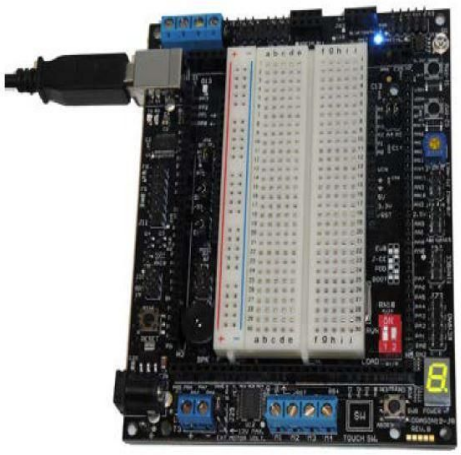


Figure 1. Dragon12-JR 9S12DG256 EVB

Lab Equipment

- PC running MS Windows
- Digital Multimeter (DMM)
- Dragon12-JR 9S12DG256 evaluation board

Lab Parts

- 2 each 8-bit LED bars
- 1 each 4-bit DIP switch (piano type)
- 2 each 470Ω, 10-pin bussed resistor network (SIP)

Lab Software

- Freescale CodeWarrior for HC12 v5.1 (C cross compile & programming environment)
- RealTerm (terminal emulation program)
- Other terminal emulators maybe used that support file transmission. (TeraTerm)

Lab Background

Dragon12-JR 9S12DG256 Evaluation Board

In this lab we are using the Dragon12-JR board, a low-cost, entry-level evaluation board (EVB), that allows for complete access to the 9S12DG256 microcontroller signals, and for this lab we

are using EVB jumper setting. Applications for single chip operation can be downloaded into the internal static RAM or flash memory.

Board Features are listed on lab manual.

Lab – Part 1

For this part of the lab is not required in the report, included is a brief introduction.

This part of the lab is a step-by-step tutorial to learn how to connect the Dragon12-JR evaluation board (EVB) to the development PC, use RealTerm to interact with the EVB, and develop a program in Freescale CodeWarrior.

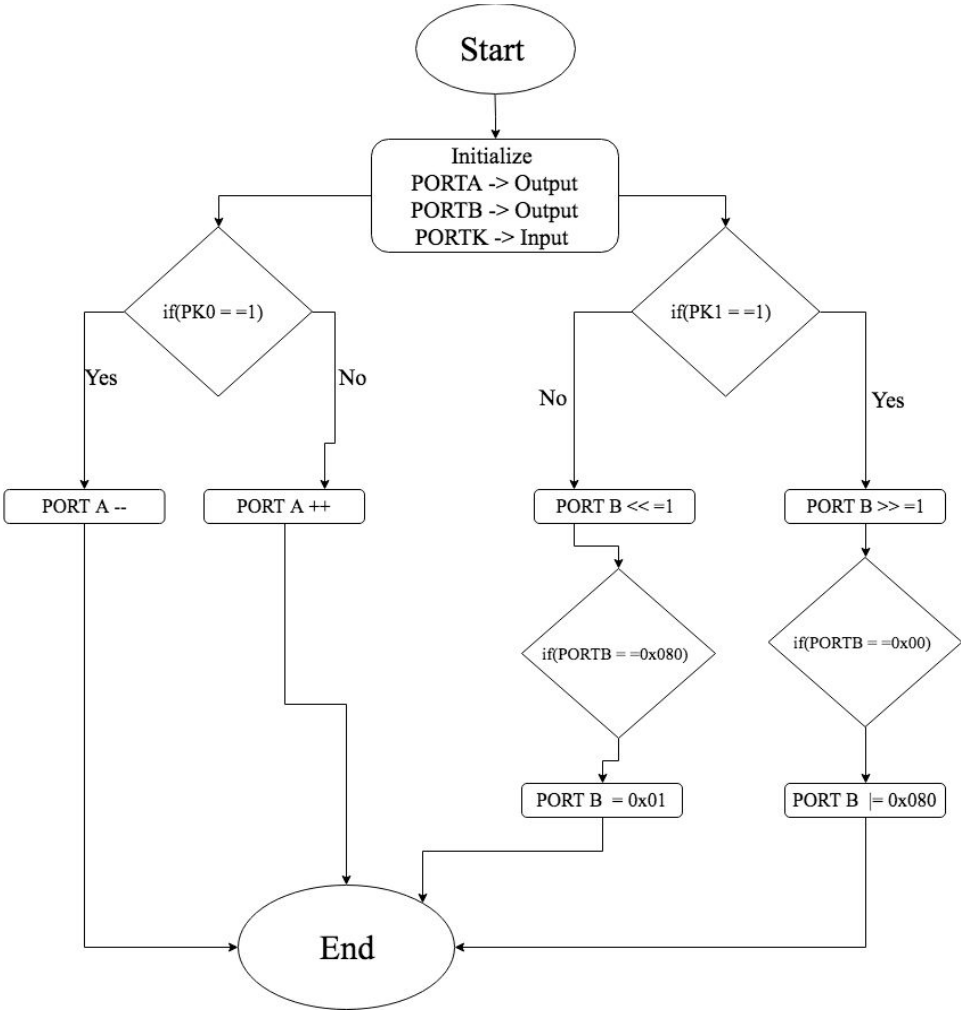
Lab – Part 2

Abstract

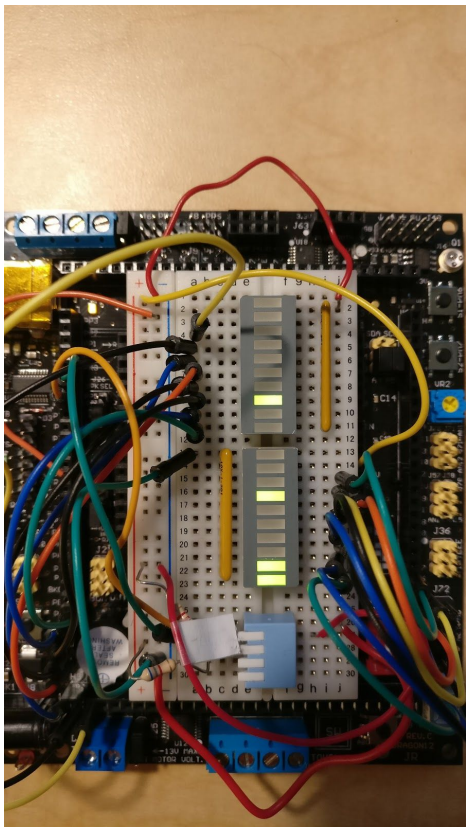
This part of the lab, we implement our understanding of the 9S12DG256-based digital system from the first part of the lab, and build a two part system where one port outputs an 8-bit, bi-directional, binary counter and the other port outputs an 8-bit, bi-directional, one-hot rotator. We would use a DIP switch in order to control the directions of both former and the latter. As recommended, we used PORT A and B as outputs while PORT K as an input. After programming, we check our outputs on two LED bars and see if our Logic is right!

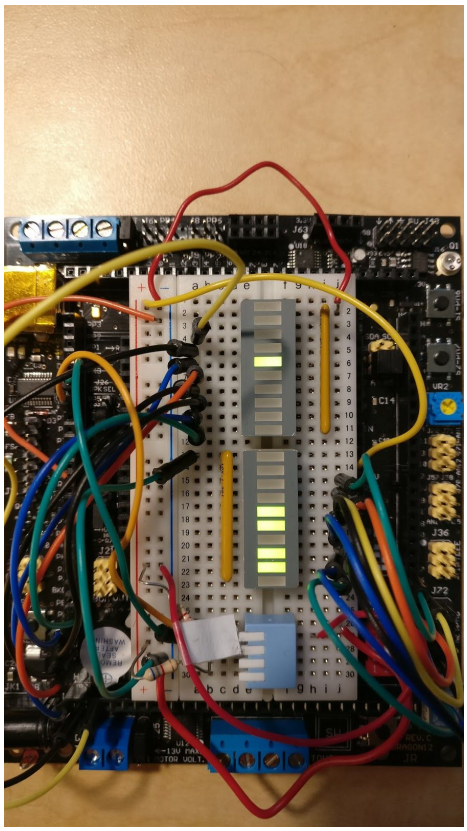
Experiment System Specification:

Flowchart diagram

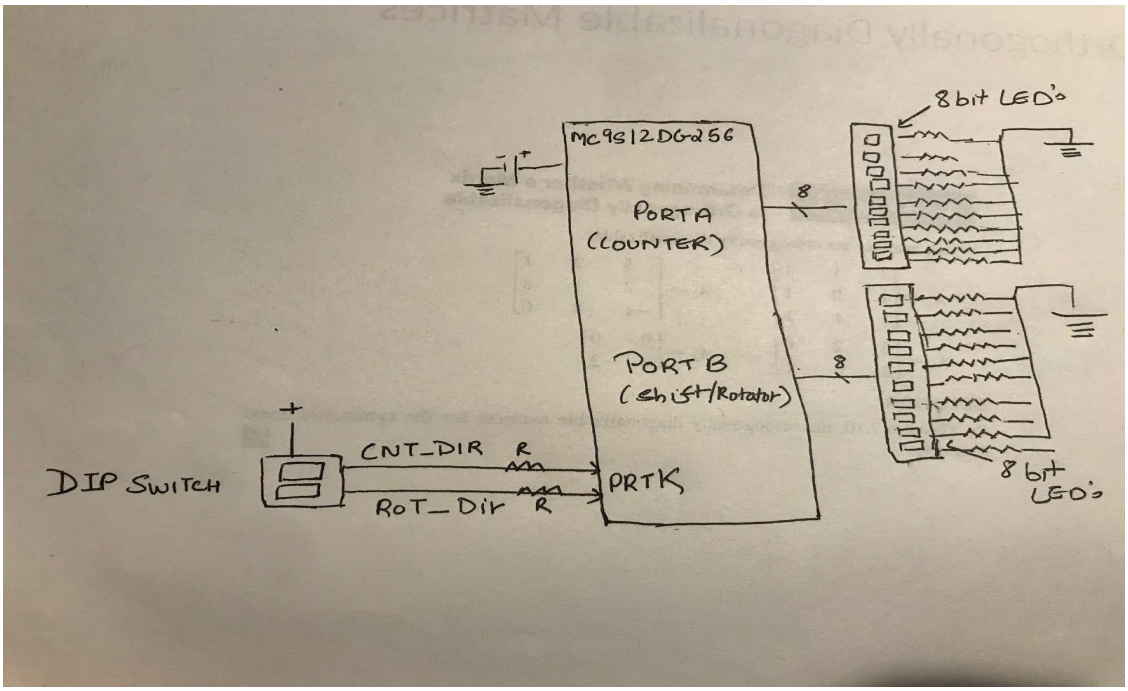


Photos





Hardware Design :



Software Design

Program Listing :

main.c

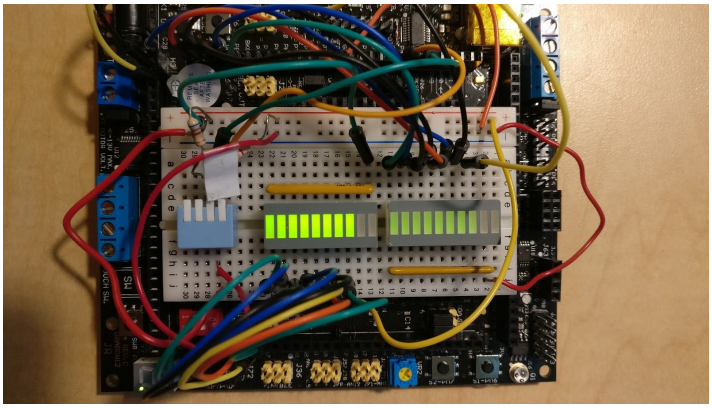
```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include <mc9s12dg256.h>

void main(void) {
  /* put your own code here */
  unsigned long i;
  // initialize port A - output
  DDRA = 0xFF;
  PORTA = 0x00;
  // initialize port B - output
  DDRB = 0xFF;
  PORTB = 0x01;
  // initallize port K - input [on/off dip switch]
  DDRK = 0x00;
  PORTK = 0x03;

  while(1) {
    for (;;) {
      for (i = 0; i < 150000; i++);

      if (!(PORTK & 0x01))
        PORTA++;
      else
        PORTA--;

      if (!(PORTK & 0x02)) {

        PORTB <<= 1;
        if (PORTB == 0x80) {
          for (i = 0; i < 150000; i++);
          PORTB = 0x01;
        }

      }
      else {
        PORTB >>= 1;
        if (PORTB == 0x00) {
          PORTB |= 0x80;
        }
      }
    }
  }
  /* please make sure that you never leave main */
}
```

Problems Encountered

We encountered lot of bugs and technical issues while doing this lab especially with the bit rotator due to some bit shifting issues with the code, it took us a while to understand them and debug them. At the end, Everything turned out to be fine and we were able to finish the lab before the due date.

Lab Questions

1. Size of S-record file : 548 bytes. According to memory map file the size should be 114 bytes
2. Tested with home desktop computer with intel core i7-8809G, which has clock speed of 3.10GHz, without software delay, from physical visual point of human there's no way to tell how fast the program is running, all LED are on at the same time.



3. #define PORTX _IO8(8) //PortX data register checks value on Pin 2

   #define DDRX _IO8(9) //Direction data register checks value for 0x09 which is Pin 0 and Pin2

4. Voltage drop across each resistor (V) = (5-1.7)V = 3.3V

   Now using ohm's law (V=IR), I = $3.3 \div (1.5 * 1000)$ A = **2.2mA.**

Conclusion

In this lab, we worked with Codewarrior in order to get started with the controller and the system, and gained experience in using the microcontroller to output out the binary counter and shift rotator. It was fun learning new concepts about RealTerm and Codewarrior, and astonishing to see so many uses of the same. For this Lab, Leya worked on the software design part of it and Tanish worked on the hardware part of it.