Computer Engineering 2DX4

2021 Laboratory Manual

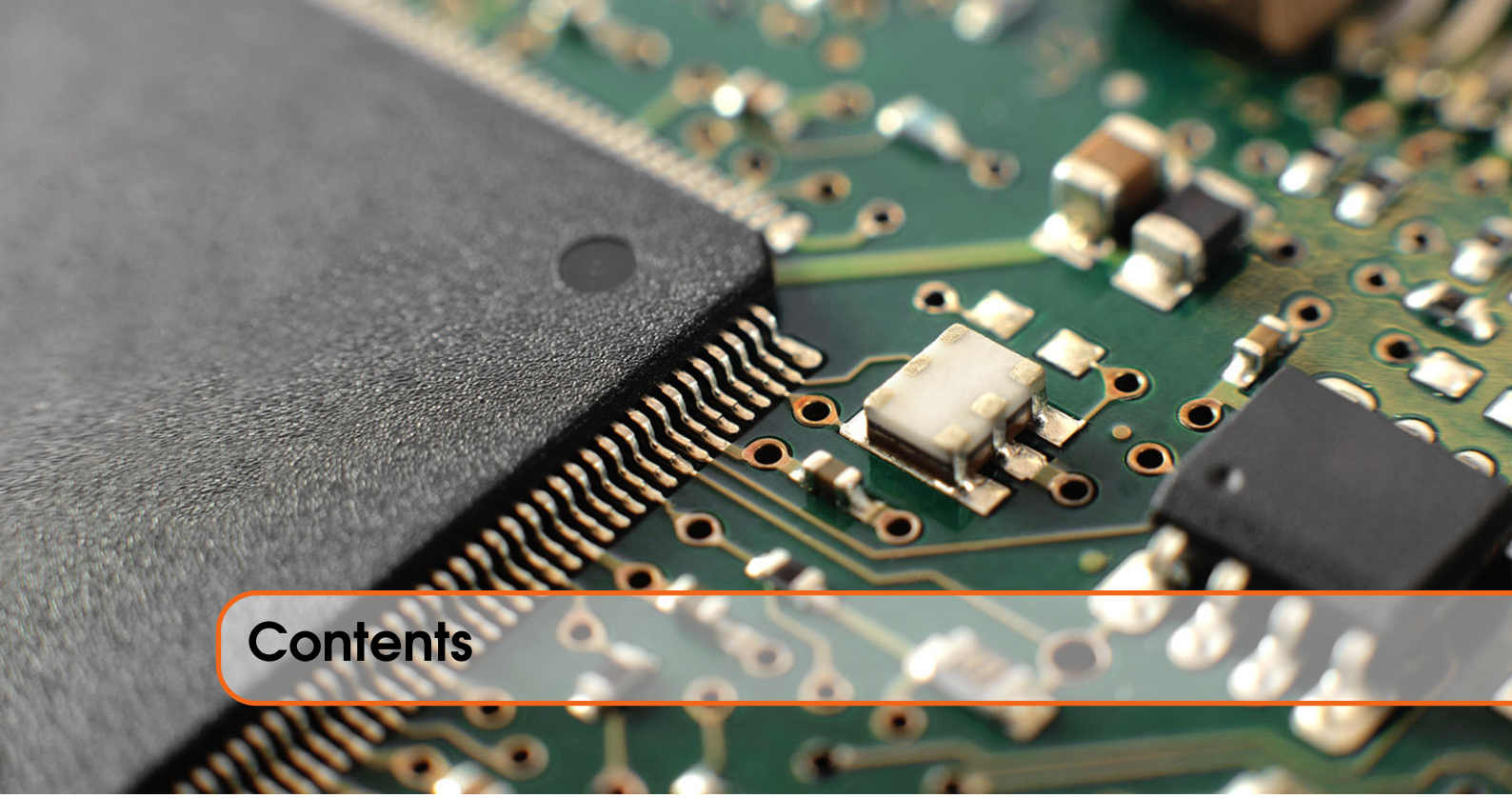
Last Update: March 30, 2021

Dr. Thomas E. Doyle

. . . the designer of a new system must not only be the implementor and the first large-scale user; the designer should also write the first user manual. . . If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important. Donald E. Knuth

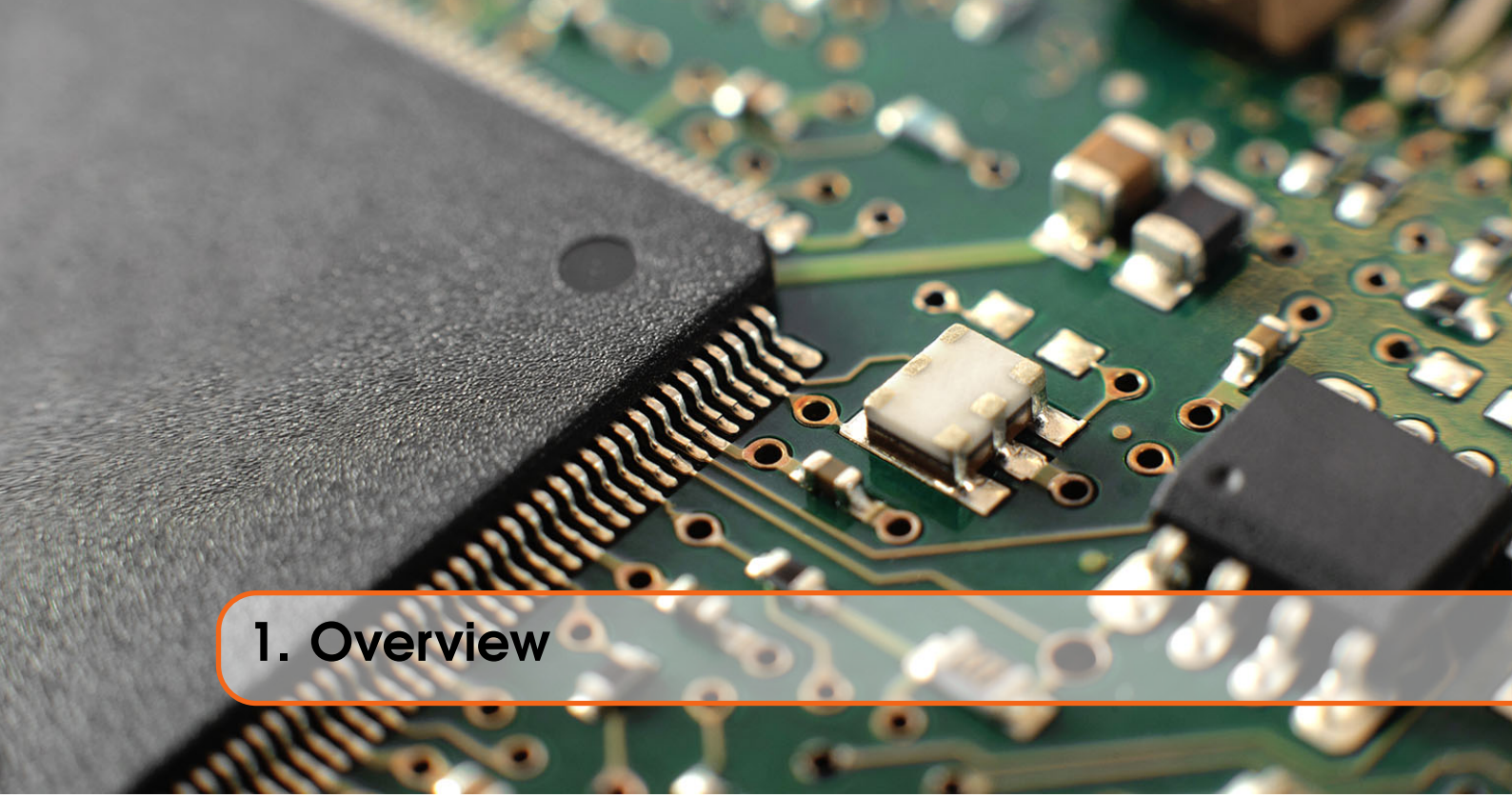
PUBLISHED BY DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, MCMASTER UNIVERSITY

January 2021



Contents

Part One: Overview



1. Overview

1.1 Microprocessor Systems Project (2DX4) Course Description:

Microprocessor systems, introduction to the design process, project development by small teams of students, oral presentations and engineering report writing.

1.2 Philosophy

Intelligent devices are ubiquitous and advancing computationally at an accelerated rate. Traditional course in Microprocessors have approached the topic from the computer architecture perspective and then as an appendix included the characteristics that permit computing to be “intelligent."

2DX4 has been designed as a project based course where lecture theory is connected with hands on experience in studio and integrated in weekly labs. Each week constitutes a knowledge thread that combine under the themes of Observe, Reason, and Act.

1.3 Using this Lab Manual

This lab manual was created using the typeset document preparation system called LaTeX. This file contains embedded and linked media content. The embedded content is designed to run inside the Adobe Acrobat Reader. Please download a copy of the manual and open with Adobe Reader on your local computer. This manual will see continuing updates throughout the term, so for this reason we do not recommend that you print this manual. The Adobe reader is free and can be downloaded here: `https://get.adobe.com/reader/`.

1.4 Evaluation

The final mark in the Microprocessor Systems Project (2DX4) course consists of the components listed in the 2DX4 course outline. The laboratory component of the course mark is also defined

there.

Laboratory exercises may offer bonus mark(s) for extra and/or advanced work by the student. **The bonus mark(s) will only apply to lab submissions that make a clear attempt to meet all non-bonus criteria (including pre-lab)**. For example, by not attempting to answer a question from the Laboratory Prelab section, the bonus mark(s) will not be considered in evaluation of the exercise. It should also be noted, although the bonuses may make it possible to achieve greater than 100% in the laboratory component of CoE 2DX4, the maximum grade assigned to the laboratory component will not be greater than 110%.

Labs run every week of the course. You are required to check the course website daily to confirm the schedule and receive corrections and/or clarifications to the lab exercises. Unless otherwise stated, prelabs are due 15-minutes after the beginning of lab. Laboratory summary or report and code are due during the student's assigned section. The student is advised that a submission after the laboratory session ends is considered late; no exceptions.

It should be stressed that due to high course enrollment, students shall not be admitted to other laboratory sessions than those assigned. An automatic zero will be applied to students attending the incorrect session.

Please enter and exit the lab punctually and pay close attention to the time limits that your TAs give you for submitting your lab exercises. Failure to have your lab execution checked because the lab time expired may result in a 0 for the lab. Failure to exit the lab punctually may result in being assigned an automatic zero for the entire lab exercise. TAs are instructed to not accept/review work that is late.

1.5 Virtual Labs Run Online

Labs will be run virtually using Microsoft Teams. At the beginning of each lab the TAs will be online to provide a brief overview of the exercises and the milestone objectives. As you complete the milestones you will need to request a TA from the Lab Team General channel for evaluation. Similarly, if you need assistance you can also request it in the General channel. Unless you have a specific follow up question, refrain from using the @specific-TA as requests are queued-handled from the General channel.

Please be mindful that your TAs will need to visit multiple students. It is important to have your questions and/or demonstrations ready.

Do not forget to upload your final work to Avenue by the deadline. Check you have uploaded the correct file(s). We cannot accept late files.

1.6 Missed Labs

MSAF process (weight does not go to final, instead it will increase relative weight of all other regular labs), MSAF applies to the individual student. There are no makeup labs. If you know in advance of an absence, your instructor may be able to arrange alternate lab time, but only if coordinated prior to scheduled lab and space is available.

1.7 Penalties

The lab ends at 5:20 and all submissions are due by then. Any submission less than 10-minutes late will be assigned a 20% penalty. Any submission more than 10-minutes late will be assigned a 100% penalty. Avenue to Learn is considered the official time.

II Part Two: Observe

2. Lab 1: Digital Signals

2.1 Objective

The objective of this lab is to gain a better understanding of the embedded hardware architecture and how to program it to detect and manipulate simple digital signals using assembly language.

2.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Text: Digital Logic textbook on combinational and synchronous sequential logic design – see Design at the Register Transfer Level
2. Text:“Embedded Systems: Introduction to ARM Cortex-M Microcontrollers", 5th edition, 6th printing, January 2019, by Valvano – Chapters 1-4
3. Lab Manual: “Getting to Hello" chapter 11
4. Technical: MSP432E401Y - Microcontroller Data Sheet
5. Technical: MSP432E4 SimpleLink Microcontrollers - Technical Reference Manual
6. Technical: MSP432E401Y - User’s Guide
7. Technical: Cortex - M3/M4F Instruction Set Technical User’s Manual
8. Technical: Cortex-M4 Technical Reference Manual

R Review how to create a new project in Keil in assembly (not C).

2.2 Pre-Laboratory Preparation

[20 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time. Prelab will not be accepted it is late.

1. Complete the Lab Safety Quiz [0 – required].
2. Briefly explain the relationship between *machine language*, *op code*, and mnemonic with an example. [1]
3. Using the Keil MDK, compile, load, and run the “Hello World" blinkLED.c code as outlined in chapter 11. Embed a very short video (10s) in your report demonstrating this achievement. To embed the short video, record and upload to Google Drive, then put the link to the video in your prelab for question 7. Make sure that it is shareable (so your TAs can view it later). Your first name must be clearly displayed in the video. You can just write it on a piece of paper and film it the video[1]. [6]
4. In relation to the MSP432E401Y board, what core processor is used and define its registers (purpose and number of bits). Is the core processor an 8-, 16-, 32-, or 64-bit architecture, and what does that mean? [1]
5. Record the exact name of the reference documentation on the core processing unit and its operation codes/language for the MSP432E401Y. [1]
6. Record the exact name of the reference documentation on the microprocessor logic systems and peripherals for the MSP432E401Y. [1]
7. Create a flow chart showing the steps to configure a GPIO port on the MSP432E401Y. For each step of configuring a GPIO port, define the relevant register’s purpose. [6 marks]
   (a) State the relevant register addresses for configuring Port M. [2 marks]
   (b) State the relevant register addresses for configuring Port N. [2 marks]

While your textbook is a valuable resource, the manufacturer of the device(s) always provide official technical references and normally these are free. Similar to books you buy on software, the textbook is a guide written *from* these technical references. The textbook can never cover the same depth as the original references and you will certainly see this with microcontrollers. It is imperative you become familiar with the technical manuals from the beginning of this course.

Pre-lab work is designed to ensure you come prepared to lab and also to help you complete the lab. This is to be done with your lab partner. If you do no have a lab partner or your partner is away please note you are still responsible to complete this work. Ensure both partners have a copy of the lab and pre-lab after submission.

2.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 2.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
normally open push button	generic
LED	onboard

Obtain the data sheets for each of the above digital devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

[1] If you choose to link to an online video please be aware of your privacy – never post your student number and full name online

2.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

2.4.1 Hello World Assembly

Regardless of the language (C or assembler) the steps to configuring and programming a general purpose input and output (GPIO) port remain the same. Review steps to programming port and review the assembly code from prior Studio(s).

Milestone 2.1 Create a new project, this time for assembly code, and write a program to blink one of the onboard LEDs from Port N. To do this you will need to look up the relevant Port N addresses.

40 marks ■

2.4.2 Pushing your button(s) Assembly

The momentary switch appears deceptively simple. While schematically shown with only two connections, they physically have 4 pins, as shown in figure 2.1. The connection of these pins is device dependent – never assume the connections! The datasheet should be consulted even for something as simple as a momentary switch, or a quick set of tests done to see (at the very least) if the device is “normally open" (n.o.) or “normally closed" (n.c.) when not being pressed.

Figure 2.1: Image of a generic momentary switch:

The next question is how to connect a momentary switch? With our own equipment we don not have the benefit of a pre-wired board. Figure 2.2 illustrates four methods to connect such a switch, but unfortunately only one will work correctly. The first time connecting a momentary switch, most will wire it as shown in figure 2.2-c) and this might appear to work; however with a normally open momentary switch, when the button is not pressed the input to the microcontroller is not connected to any voltage input (referred to as *floating*). When an input to a microcontroller floats it cannot be assumed to be any value. The proper wiring requires the input to be *pulled-up* or *pulled-down* when

the button is not being pressed. Figure 2.2-b) properly implements a “pull-up resistor" whereby the input to the microntroller is pulled to a logic high when the button is NOT pressed and then forced the input to the micro to logic low when the button is pressed. Ensure you understand this arrangement as it may be counter-intutive: button unpressed = 1, button pressed = 0.

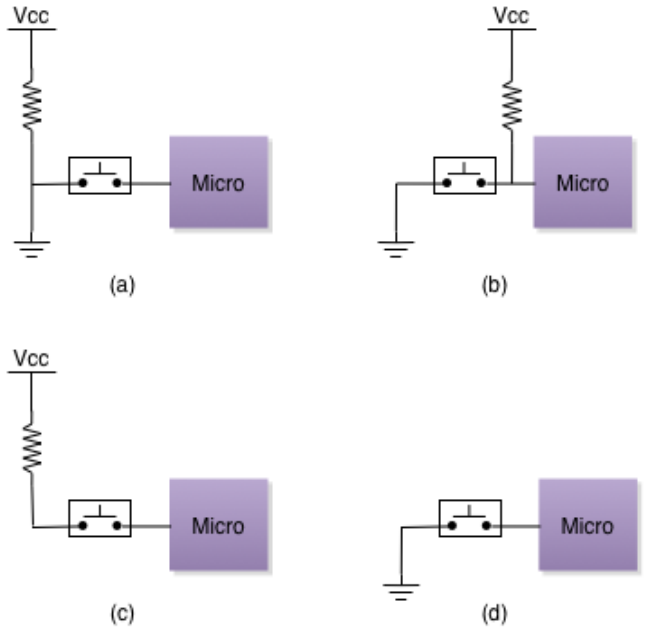


Figure 2.2: Connecting a momentary switch: a) c) and d) incorrectly wired momentary switch input to a microcontroller, b) a correctly wired pull-up resistor for momentary switch input to a microcontroller

Now that we can program a GPIO pin as an output to turn an LED on (logic high) and off (logic low), the next step is to extend our code to capture a GPIO pin’s input.

Milestone 2.2 Return to reviewing the configuration steps of the GPIO, but in this case, use Port M bit 0 to capture if a push button is open (not pressed) or closed (pressed). When the button is pressed, turn on the designated LED. When the button is not pressed, turn off the designated LED. This milestone is to be done in assembly code.

If your student number’s least significant digit is:

1. 0, 4, or 8 then use LED 1
2. 1, 5, or 9 then use LED 2
3. 2, or 6 then used LED 3
4. 3, or 7 then use LED 4

40 marks ■

2.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 2.2: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstration and explain of a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

2.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab report has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

2.7 Supplied Reference Code

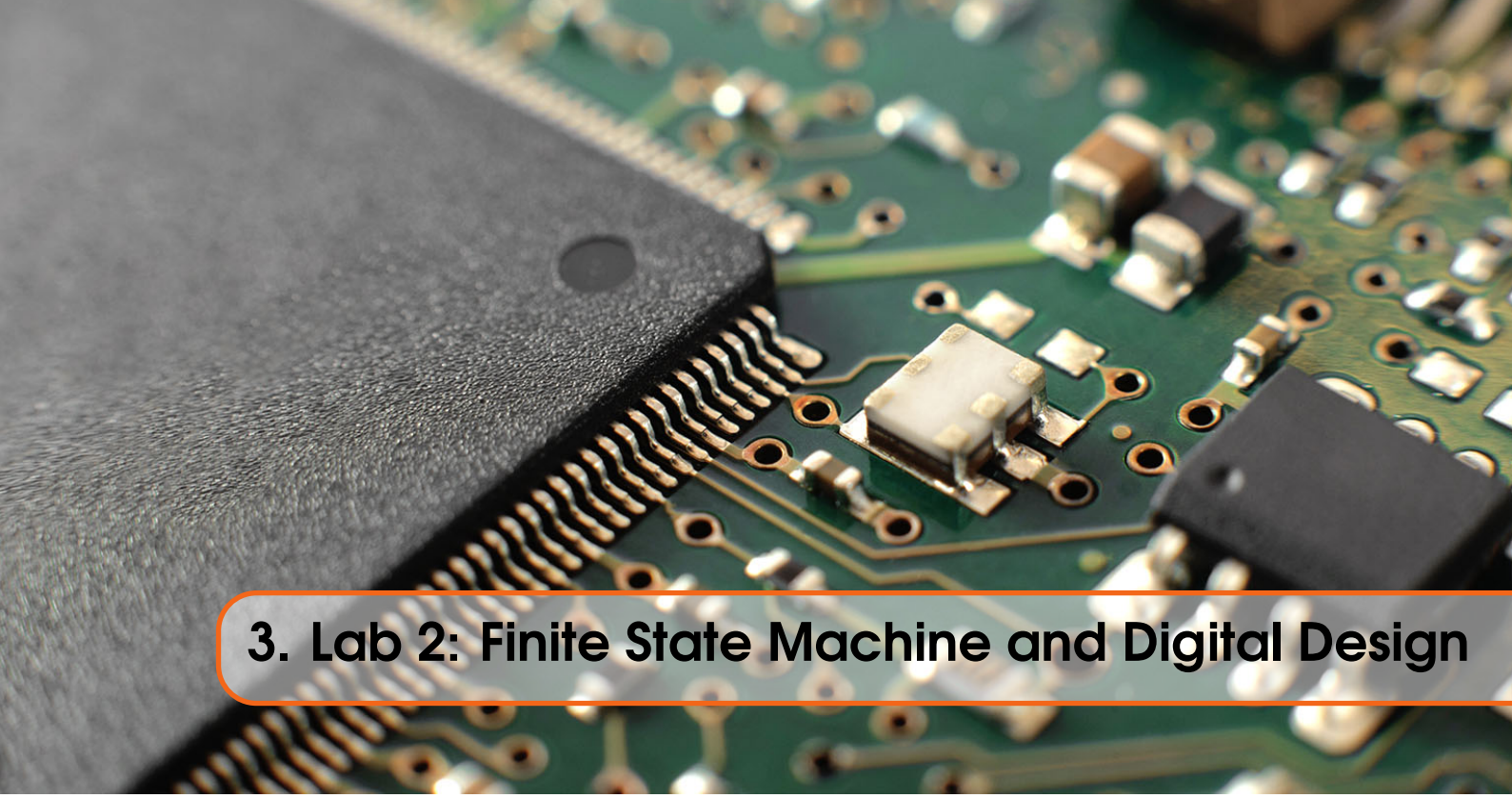
```
//0. Documentation
//**************************************************************************
// Simple output C program based on the modification of the blinkLED.c code
// provided by TI.
//
// There are 7 steps to initialize a GPIO port for general use
//    i. Activate the clock for the port by setting the corresponding bit in
//        the RCGCGPIO register and then wait for status bit in PRGPIO
//        register to be true.
//   ii. Unlock port if using PD7.
//  iii. Disable analog function of the pin. (Upon reset default = disabled)
//   iv. Clear bits in PCTL to select regular digital function (see textbook
//        tables 4.1&4.2, noting upon reset the default function is digital).
//    v. Set the data direction register (i.e., input or output).  In the DIR
//        register, a bit set to: 0 = input, 1 = output.
//   vi. Clear bits in the alternate function register (remember pins / ports
//        can have alternate functionality when configured as such.
//  vii. Enable the digtal port
//
//  Note: Step i must be done first, but remaining steps may be performed in
//    any order.  If your intention is to use a GPIO pin in its default
//    digital state, then steps iii, iv, and vi can be omitted
//    (also ii if not using PD7).
//
//  TEDoyle
//  October 26, 2019

//**************************************************************************
//1. Preprocessor Directives
//**************************************************************************
```

```c
// There are library files that define the peripherals, ports, and pins.
// Review the msp432e401y.h file to become familiar with the symbolic labels
// that have been preassigned to help simplify your programming. These labels
// can be different than your textbook, thus library familarity is important.

#include "msp.h"    // this libabry file chooses the approproate MSP library

//2. Functions
//*****************************************************************************

// For now, no functions outside of main().

//3. Main
//*****************************************************************************

int main(void)
{
  // step i.
  // Enable GPIO clocks for N & M peripheral
  SYSCTL->RCGCGPIO |= SYSCTL_RCGCGPIO_R12;  // Port N (onboard LED)
  SYSCTL->RCGCGPIO |= SYSCTL_RCGCGPIO_R11;  // Port M (GPIO for Lab 0 EPROM)

  // Check if the peripheral access is enabled.
  while(!(SYSCTL->PRGPIO & SYSCTL_PRGPIO_R12)); //N
  while(!(SYSCTL->PRGPIO & SYSCTL_PRGPIO_R11)); //M

  //skip steps ii, iii, iv

  // step v.
  /* Enable the GPIO pins for the onboard LEDs (PN1 = out, PN0 = out). */
  GPION->DIR = 0x03;

  /* Enable the GPIO pins for the EPROM (PM7:0 = out). */
  GPIOM->DIR = 0xFF;

  //skip steps vi

  // step vii.
  GPION->DEN = 0x03;
  GPIOM->DEN = 0xFF;

  // main code
  uint32_t counter = 0;

  GPION->DATA = 0x01;   //initialize the bit values

while(1)
  {
      // Toggle GPIO pin value
      GPION->DATA ^= BIT0;
      GPION->DATA ^= BIT1;
      for(counter = 0; counter < 1000000; counter++) {} //orig set to 200000
      GPIOM->DATA = 0xFF;                               //EPROM bits for Lab0
  }
}
```

Listing 2.1: C code example for Lab 0

2.8 Grading Summary Table

Component	Weight	Grade
Pre-Lab Questions	20	
Milestone 1	40	
Milestone 2	40	
Total	**100**	
Deductions		
Final Score		



3. Lab 2: Finite State Machine and Digital Design

3.1 Objective

The objective of this lab is to incorporate digital design methodology for the development of embedded systems. Using a Finite-State-Machine (FSM) approach, and extending our knowledge of GPIO in this lab you will gain a better understanding of the embedded hardware architecture and how to program it to detect and manipulate digital signals using assembly language.

3.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Technical: MSP432E401Y - Microcontroller Data Sheet
2. Technical: MSP432E401Y - Technical Reference Manual
3. Technical: MSP-EXP432E401Y - Schematic
4. Technical: ARM Cortex M4 Processor Technical Reference Manual
5. Technical: ARM M Architecture Reference Manual
6. Technical: The Cortex-M4 Instruction Set

3.2 Finite State Machine

When designing complex systems, it is beneficial for the designer (and any one reviewing the design) to abstract away from the implementation details and to focus on describing the solution. Similar to how flowcharts and UML are used to describe procedural and object-oriented program, the finite-state-machine (FSM) approach is useful for digital system design.

As your designs become more complex, the FSM approach becomes more powerful.

This lab is intended to introduce the FSM concept with your early design problems so that you may use it when the system design becomes more complex. While there are other resources, you can refer to your 2DI4 notes or your textbook section 6.5.2.

3.3 Pre-Laboratory Preparation

[20 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time. Prelab will not be accepted it is late.

1. Flowcharts are useful to describe procedural steps in a simple system. Create a flowchart that illustrates a combinational digital lock that opens only when the parallel entry is 1001. When the digital lock is open, the output of the circuit should be a logic 1 to turn on an LED. When the input parallel sequence fails, the design should reset. The LED should remain off until the lock is opened. While not required for the pre-lab, consider how this would looks as a finite state machine. (Don't worry about hardware settings for this question) [10]

2. Starting with a finite state machine, design a 4-bit digital lock that opens only when the 1-bit serial sequential input is entered as 1,0,1,1. When the digital lock is open, the output of the circuit should be a logic 1 to turn on an LED. When the input serial sequence fails, the design should reset to the initial state. The LED should remain off until the lock is opened. (Don't worry about hardware settings for this question) [10]

3.4 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 3.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
Normally open push button	generic
LED	onboard

Obtain the data sheets for each of the above digital devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

3.5 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

3.5.1 Parallel I/O – Combinational Lock

Generally an embedded system is handling much more functionality than a single button or switch. Review steps to configuring a port for input and output. For this section we will implement a combinational lock system with a data load functionality.

Milestone 3.1 Configure Port M for bits 0-2 (PM0, PM1, PM2) to be inputs. Configure your system as:

1. Connect PM[0-2] to three push buttons or toggle switches that represents a binary value input.
2. Connect PM3 to one push button or toggle switch that will be used to load the binary value for logic testing.
3. If your student number ends in an odd number, D2 on represents a successful code load and D1 on represents an unsuccessful code load. If your student number ends in an even number, D1 on represents a successful code load, D2 on represents a unsuccessful code load.

Your specific code is based on your student number. Observing the last 3 digits, inspect each digit. If a digit is even then the bit code at that position is 1. If a digit is odd then the bit code at that position is 0. For example, if the student number was 12346078, then the last 3 digits being 078, then the combinational code for your lock would be 101.

Write the assembly code to implement this parallel I/O system as a digital lock.
20 marks ■

3.5.2 Sequential I/O – Sequential Lock

You can likely reuse the port configuration from the prior milestone.

Milestone 3.2 Configure Port M for 2-bits to be inputs. Configure your system as:

1. Connect PM0 to one push button or toggle switch that represents a binary value input.
2. Connect PM4 to one push button or toggle switch that will be used to load the binary value for logic testing. This is the synchronizing clock that we would have used in 2DI4.
3. If your student number ends in an odd number, D2 on represents a successful code load and D1 on represents an unsuccessful code load. If your student number ends in an even number, D1 on represents a successful code load, D2 on represents a unsuccessful code load.

Your specific sequential code is based on your student number. Observing the last 4 digits, inspect each digit. If a digit is even then the bit code at that position is 1. If a digit is odd then the bit code at that position is 0. For example, if the student number was 12346078, then the last 4 digits being 6078, the sequential serial code for your lock would be 1,1,0,1.

Write the assembly code to implement this synchronous sequential system as a digital lock. Your design from the pre-lab can be updated and represented the FSM approach to designing this system. **30 marks** ■

3.5.3 Troubleshooting

When debugging realtime systems it can be a challenge for IDEs to help troubleshoot because pausing an embedded program can modify the values that the programmer seeks to inspect (plus it's not realtime when it's paused). The benefit of LEDs for realtime debugging should never be underestimated.

Milestone 3.3 Return to your Sequential Lock FSM. Assign a unique binary code to each state in the FSM (2 or 3 bits depending on your design). Modify your code to output the present state binary code to LEDs, thus allowing you to see exactly which state your program is in during operation. (using the 4 onboard LEDs, set 2-3 LEDs for state value and 1 LED for successful

unlock) You may pick the onboard LEDs you use. **30 marks** ■

3.6 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone 3

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 3.2: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

3.7 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

3.8 Grading Summary Table

Component	Weight	Grade
Lab Questions	20	
Milestone 1	20	
Milestone 2	30	
Milestone 3	30	
Total	**100**	
Deductions		
Final Score		

4. Lab 3: Analog Signals

4.1 Objective

The objective of this lab is to gain a better understanding of the embedded hardware architecture and how to program it to acquire analog signals using C language that is cross-assembled for the ARM.

4.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Text:“Embedded Systems: Introduction to ARM Cortex-M Microcontrollers", 5th edition, 6th printing, January 2019, by Valvano – Chapter 10.1, 10.4, 10.5
2. Technical: MSP432E401Y - Microcontroller Data Sheet
3. Technical: MSP432E401Y - Technical Reference Manual (Refer to chapter 10)
4. Technical: MSP-EXP432E401Y - Schematic
5. Technical: ARM Cortex M4 Processor Technical Reference Manual
6. Technical: ARM M Architecture Reference Manual
7. Technical: The Cortex-M4 Instruction Set

R Review how to create a new project in Keil in C. Return to Lab 1 for example C code. From this lab forward all programming will be done in C for assembly.

R The ADC is a complex system on the MSP432E401Y. You are provided code to get you started; however, to effectively use the code you will need to understand the ADC process and the concepts. Ensure you review your textbook, lecture, studio and support videos.

4.2 Pre-Laboratory Preparation

[30 marks total]

Prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab is must be submitted within the first 15-minutes of lab time.

1. What method of analog to digital conversion does the MSP432E401Y ARM microcontroller use for its ADC? How many bits? [3 marks]
2. Converting an analog signal to digital will result in quantization error - show the calculations and explain the MSP432E401Y's maximum quantization error. [3 marks]
3. Draw a flowchart outlining the steps to configure the ADC on the MSP432E401Y. Hint: there are 13 steps. [13 marks]
4. Complete the table below assuming a 12-bit ADC (same as MSP432E401Y's ADC), assuming a full-scale voltage of 3.3V [11 marks]:

Analog Voltage (V_H)	x-bit ADC (hex)
0.00	
0.33	
0.66	
1.00	
1.33	
1.66	
2.00	
2.33	
2.66	
3.00	
3.30	

4.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 4.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
Bonus: 7 LEDs	generic

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

4.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

4.4.1 Analog to digital conversion (ADC) of DC Signal

In this milestone you will need to select and configure one of the available analog pins to engage the ADC system. Review steps to programming port and the textbook section on the ADC.

Milestone 4.1 Connect the signal generator and complete the table with the binary values and the percent of maximum digital value. You will need to write your code to copy the converted digital value to a register for your inspection. Your TA will be asking how the ADC value is found.

25 marks ■

Analog Voltage (V_H) DC	x-bit ADC (hex)	Percent of Max Value
0.00		
0.33		
0.66		
1.00		
1.33		
1.66		
2.00		
2.33		
2.66		
3.00		
3.30		

R Recall Week 1 Studio 0. In studio you were trying to measure a much smaller signal (maximum 10mV). Consider that such a small amplitude would only produce values from 0 to 12 of a maximum of 4095 using a 12-bit ADC. We used an op amp to provide voltage gain so that our 10mV would be amplified to the ADC's V_{RHP}, thus using the full 4095 ADC digital range.

For this lab, we are using the signal generator to set the signal amplitude to the ideal value instead of using an op amp.

4.4.2 ADC of Sinusoidal Waveform

A sinusoidal waveform is a periodic signal that varies with time. As discussed in lecture, taking a measurement of this waveform (referred to as *sampling*) requires the programmer to choose a rate by which to take measurements (the sampling rate). Sampling too slow results in aliasing. Sampling too fast can consume your computational resources, such as memory and cpu. Based upon the assigned sinusoid frequency, implement Shannon's sampling theory for the reproduction of the original sinusoid waveform's frequency.

R Pay very close attention to the waveform output of the signal generator (hint, it should be displayed and confirmed on scope before connecting to microcontroller).

Before proceeding, check the following:

1. Ensure the sinusoid signal is within the voltage range of the ADC (V_{RHN} V_{RHP}).
2. Ensure the signal generator output and the microcontroller power supply are connected to the same reference.

If you're not sure, please ask your TA.

Add the student numbers of the lab group together and observe the least significant digit for the assigned sinusoid frequency.

Least Significant Digit	0-3V Sinusoid Frequency (Hz)
0, 3, 6, 9	100
1, 4, 7	120
2, 5, 8	140

Milestone 4.2 Configure signal generator to output a sinusoid 3Vpp (0-3V DC) at the assigned frequency. Configure ADC to sample the sinusoid without aliasing (reproduce frequency content). Clearly state your sampling rate and explain how this was achieved in your code.

In Studio we were able to use the functional debugging approach by setting up an IDE Watch where a global array was defined and it could be inspected during execution and post-execution. Our data of interest was small enough that we could transcribe by hand the values into an excel document for plotting. Given our sampling rate, and meaningful duration of time will mean hundreds of data points which will be impossible to transcribe during your lab time. Instead we will use Keil's Comman Line tools to export our data to a file for analysis – see the pre-lab video for a demonstration of using the Keil Command Line for exporting data from our microcontroller's memory. (see Appendix for assistance with Keil debug commands: LOG, EVal, D, FUNC, and creating .ini files)

Plot 500ms of captured waveform data.

35 marks ■

4.4.3 ADC of Square Waveform

Repeat the previous milestone by changing the sinusoid to a square wave.

Milestone 4.3 Configure signal generator to output a square wave 3Vpp (0-3V DC) at the assigned frequency. Configure ADC to sample at the same frequency you used for the previous milestone. Explain the result.

Plot 500ms of captured waveform data.

10 marks ■

4.4.4 BONUS: Frequency Counter

What assumptions can you make about a sinusoid? What additional assumptions are there when a sinusoid is input to a microcontroller's ADC?

Milestone 4.4 Write a program to calculate the frequency of a waveform which is assumed to be non-aliased upto 100Hz. Your TA will ask you to modify the signal generator frequency to a value of 100Hz or less and your program must automatically measure and update the frequency count. Use the functional debugging approach we employed an IDE Watch in Studio where an global variable was be inspected during execution and post-execution.
BONUS 10 marks ■

4.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone 3
4. BONUS Milestone 4

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 4.2: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

4.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab report has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

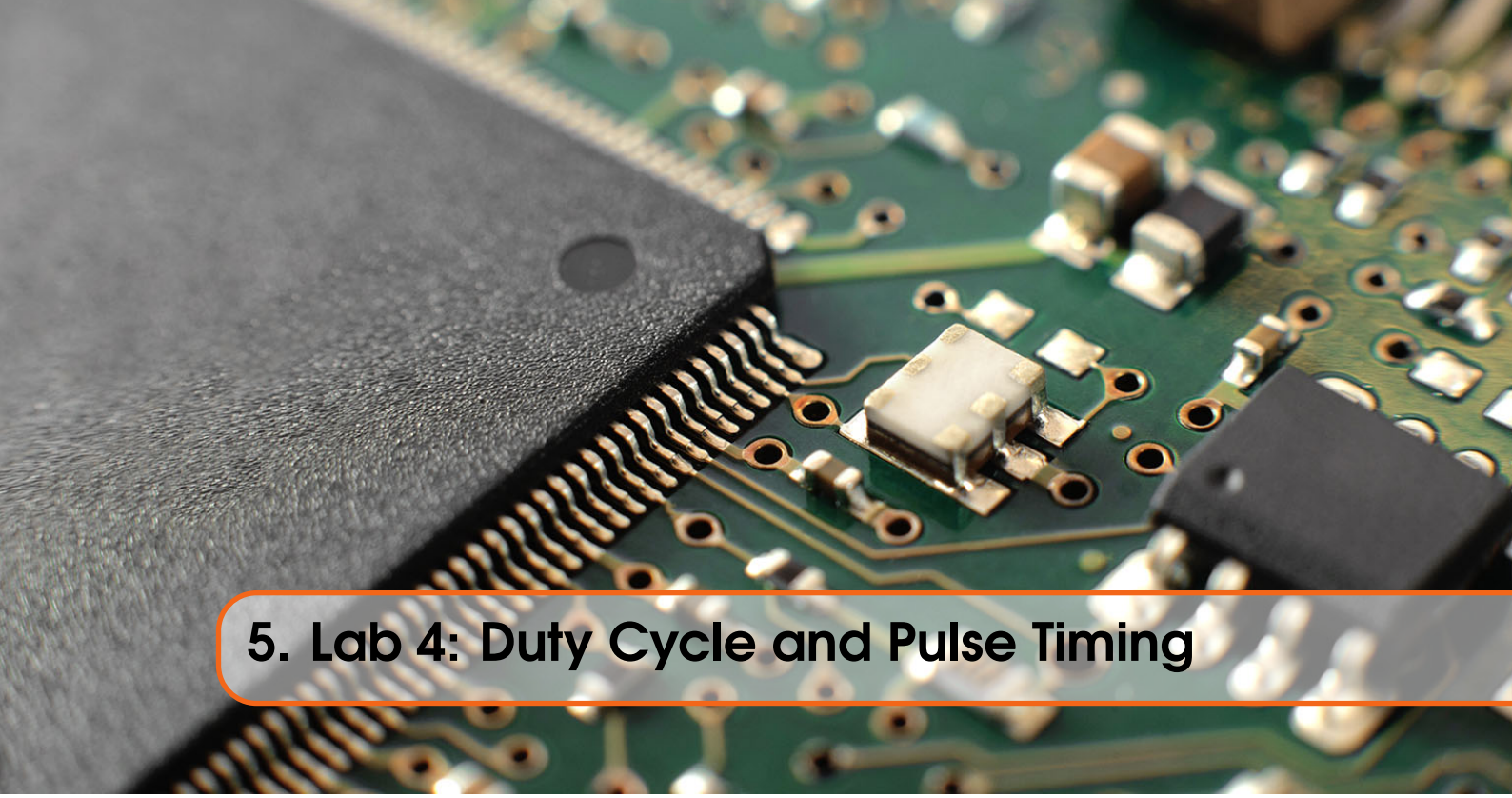
   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab report to Avenue.

4.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	30	
Milestone 1	25	
Milestone 2	35	
Milestone 3	10	
BONUS Milestone 3	10	
Total	**100**	
Deductions		
Final Score		

III Part Three: Reason



5. Lab 4: Duty Cycle and Pulse Timing

5.1 Objective

The objective of this lab is to use embedded properties and timing using C language that is cross-assembled for the ARM. To illustrate the timing we will be using LEDs and a stepping motor.

5.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Text:“Embedded Systems: Introduction to ARM Cortex-M Microcontrollers", 5th edition, 6th printing, January 2019, by Valvano – Chapter 8.7-8.8
2. Technical: MSP432E401Y - Microcontroller Data Sheet
3. Technical: MSP432E401Y - Technical Reference Manual
4. Technical: MSP-EXP432E401Y - Schematic
5. Technical: ARM Cortex M4 Processor Technical Reference Manual
6. Technical: ARM M Architecture Reference Manual
7. Technical: The Cortex-M4 Instruction Set
8. Technical: Velleman Stepper Motor with Driver - User Guide (`https://www.velleman.eu/downloads/29/vma401_a4v01.pdf`)
9. Technical: Velleman VMA401 - Stepper Motor Basics (`https://www.velleman.eu/downloads/29/infosheets/vma401_stepper_motor.pdf`)
10. Technical: Texas Instrument ULN2003 Datasheet - Motor Driver (`https://www.velleman.eu/downloads/29/infosheets/uln2003_datasheet.pdf`)
11. Reference: Primer on the stepper motor `https://www.youtube.com/watch?v=B86nqDRskVU`.

R Review how to create a new project in Keil in C.

5.2 Pre-Laboratory Preparation

[25 marks total]

All following laboratories will require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab is must be submitted within the first 15-minutes of lab time. No prelab is accepted if you are more than 15 minutes late.

1. Draw a flowchart for an ARM microcontroller program to flash an LED with a 50% duty-cycle. Set the period of one cycle as a variable to be later defined.[5 marks]
2. Based upon your flowchart, write a C program to flash the onboard LED. Assuming a 50% duty-cycle the period of one cycle is defined based upon the least-significant-digit (LSD) of the your student number. Same as you did in lab 1, embed a very short video (10s) in your report demonstrating this achievement. Include structured and commented source code (must have your name and student number) in the appendix of your lab summary. [20 marks]
   LSD 0, 3, 6, 9: cycle period is 0.50 s.
   LSD 1, 4, 7: cycle period is 1.00 s.
   LSD 2, 5, 8: cycle period is 2.00 s.

5.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 5.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
Stepper Motor	RB-Vel-133
RGB LED	generic

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

5.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

5.4.1 Variable Duty Cycle for PWM of an LED

Your pre-lab and prior lab work have implemented 50% duty-cycle square wave. In this milestone, you will create a program that varies the the duty-cycle of your square wave. Varying the time high is one method of implementing pulse-width-modulation (PWM). For this milestone, you will be using only the RED input to a RED-GREEN-BLUE LED on the lab breadboard (you can test your code at home using your AD2). Refer to the bonus milestone for control of the other inputs (for now just ground the GREEN and BLUE).

Milestone 5.1 Create a function called `IntensitySteps` that generates a square wave (frequency of 100 Hz) with a duty-cycle that steps from 10% to 100% (10% increase per step) and then decreases from 100% to 10% (10% decrease per step).

The function `IntensitySteps` should call a separate function (`DutyCycle_Percent`) which accepts an unsigned integer input indicating the percentage (out of 255) duty-cycle that should be output. For example, a value of 25 would generate ∼10% duty-cycle.

In order to see the effect of duty-cycle on the LED in the lab, each duty-cycle should be repeated ten times (e.g., 10% duty-cycle repeated ten times, 20% duty-cycle repeated ten times etc.). You can verify the duty-cycle of your square wave on the oscilloscope or using your AD2.
35 marks ■

5.4.2 Stepper Motor

Stepper motors are popular for microcontroller application because you can control the position using a binary sequence. For this milestone we will be using PortM for four outputs labelled as A, B, A', and B'. For your convenience, the four bits selected from PortM should be contiguous. These four pins/bits will be used to output a control sequence to the stepper motor. Figure 5.1 illustrates the stepper internal configuration. Referring to the stepper motor datasheet you will find one step pattern of interest for our unipolar stepper motor design, called "full step". Applying the bit sequence 0b1100, 0b0110, 0b0011, 0b1001 will make the rotor to turn clockwise. A really good primer on this stepper can be found here: `https://www.youtube.com/watch?v=B86nqDRskVU`

Milestone 5.2 Write a C program to control the rotation of the provided stepper motor. Do/determine the following:

1. write function(s) that control direction of rotation and number of steps (assume full step mode),
2. empirically determine the minimum time delay required between steps for each mode (hint - it cannot be 0),
3. determine how many step it takes for full step mode to complete 1 full revolution, and
4. demonstrate 1 full revolution clockwise following by 1 full revolution counter-clockwise.

40 marks ■

5.4.3 BONUS: RGB LED

Milestone 5.3 Implement a PWM method that implements separate duty-cycle functions (or one function accepting 3 inputs). Similar to milestone 1's `DutyCycle_Percent`, each function input shall be an unsigned 8-bit value to control the RED, GREEN, and BLUE percent duty-cycle. Three output pins must be configured and connected to the RGB LED. Demonstrate to the TA that you are able to modify the RGB colour of the LED based upon any percent duty-cycle input for each of RED, GREEN, and BLUE. Your TA will provide you new percentages for

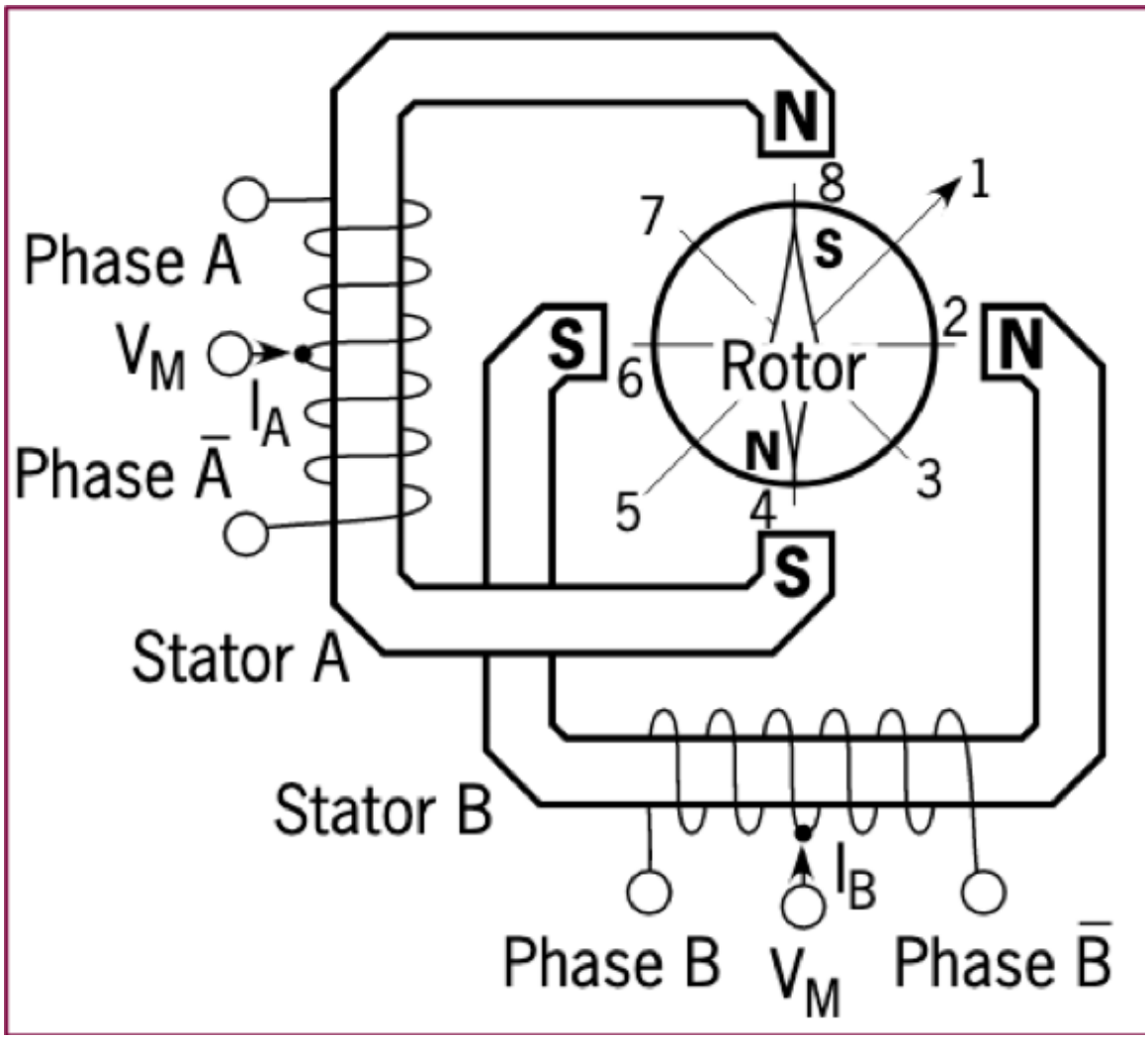


Figure 5.1: Unipolar Stepper Motor

each.
10 marks ∎

5.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone Bonus

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

5.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

Table 5.2: Evaluation Rubric

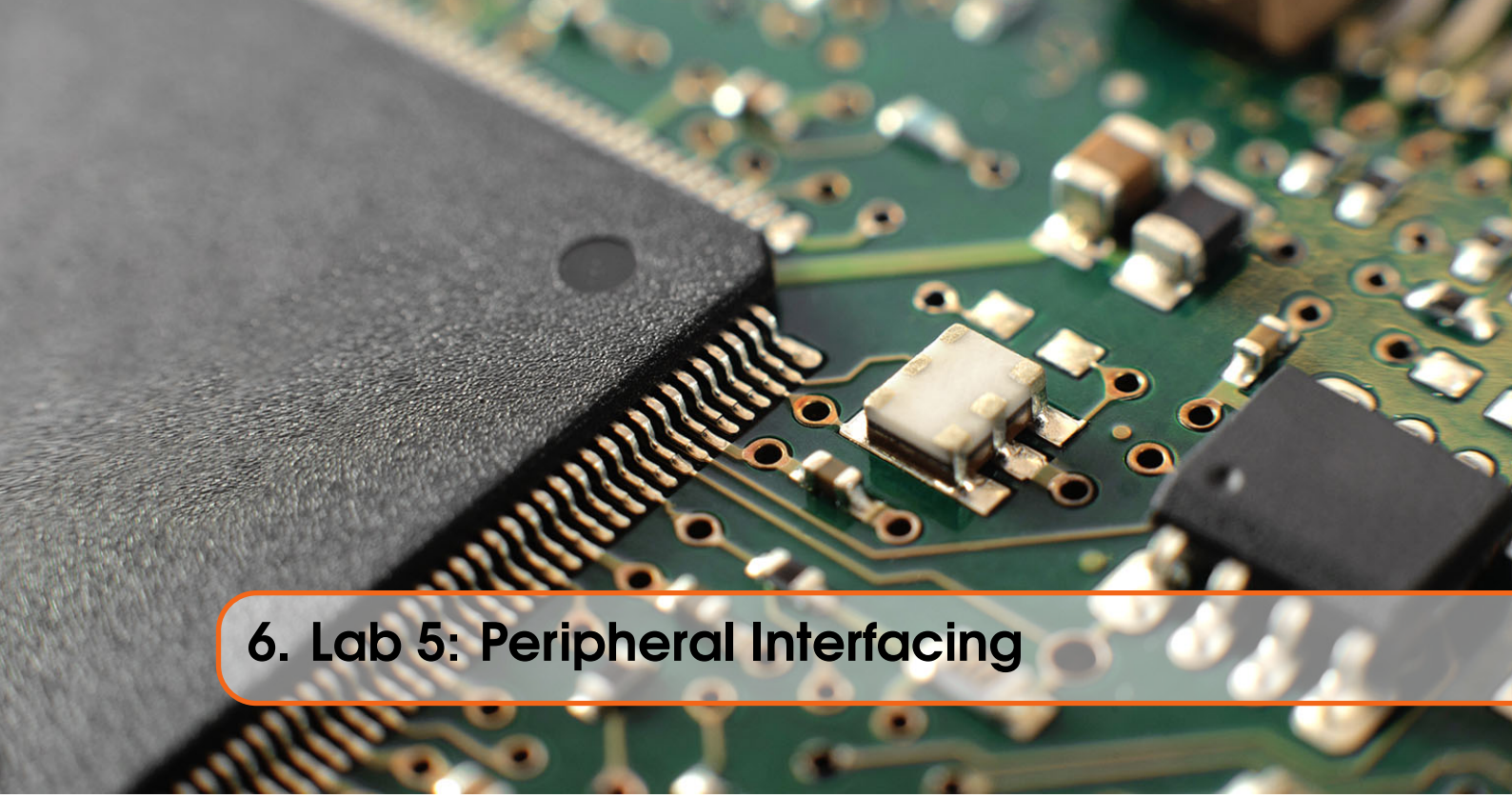
Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

5.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	25	
Milestone 1	35	
Milestone 2	40	
Milestone Bonus	10	
Total	**100**	
Deductions		
Final Score		



6. Lab 5: Peripheral Interfacing

6.1 Objective

The objective of this lab is to start interfacing peripherals and integrating embedded concepts. In this lab we will be decoding keypad input. The keypad must be interfaced via hardware and software to create a functional user interface.

6.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Text:“Embedded Systems: Introduction to ARM Cortex-M Microcontrollers", 5th edition, 6th printing, January 2019, by Valvano – Chapter 8.3, 8.5 (optional 8.4 uses different LCD).
2. Technical: MSP432E401Y - Microcontroller Data Sheet
3. Technical: MSP432E401Y - Technical Reference Manual
4. Technical: MSP-EXP432E401Y - Schematic
5. Technical: ARM Cortex M4 Processor Technical Reference Manual
6. Technical: ARM M Architecture Reference Manual
7. Technical: The Cortex-M4 Instruction Set
8. Technical: 4x4 Keypad (`https://www.mouser.ca/datasheet/2/626/Keypads_96-335254.pdf`)
9. Reference: American Standard Code for Information Interchange (ASCII) `http://en.wikipedia.org/wiki/ASCII`

R Review how to create a new project in Keil in C.

6.2 Pre-Laboratory Preparation

[25 marks total]

All following laboratories will require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab is must be submitted within the first 15-minutes of lab time. No

prelab is accepted if you are more than 15 minutes late.

1. Referring to textbook section 4.2.2, figure 4.12, from a programmer's perspective explain how the two states of the switch (open, closed) work for negative logic, external. You may assume the port pin is fully pre-configured as an input. [5]
2. Review see textbook figure 8.15. This figure illustrates an electrical structure similar to the lab's Grayhill 96BB2-006-R 4x4 keypad. From the figure, determine if the switches are configured as:
   (a) positive logic, external
   (b) negative logic, external
   [5]
3. Create a flowchart for a program that scans the 4x4 keypad using 4-output and 4-input GPIO pins and identifies which single button has been pressed (assume only one can be pressed at a time). [15]

6.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 6.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
4x4 Keypad	Grayhill 96BB2-006-R

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

6.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

6.4.1 4x4 Keypad Button Identification

In Studio you have already completed 1-button, 1-row/column, and a 2x2 keypad decoding. For this milestone you will be extending that work to identify which button was press on the full 4x4 keypad.

Milestone 6.1 Using the scanning technique illustrated in your textbook (Valvano chapter 8.5) and Studio, write a C program determine which key has been pressed and store the 8-bit PM[3:0] PE[3:0] value of the scan in Keil memory according to table 6.3. (Note: the binary value will be used in a following milestone)

Figure 6.1: Input Devices Keypad 4x4 Black Phone Legend/White

Table 6.2: Pins and Ports for Lab 3

Port	Pins	Application
Port E	3:0	Keypad scanning (output)
Port M	3:0	Key press decode (input)

25 marks

Table 6.3: Key Press Decoding

Key Pressed	Binary Value	Scanning Input Code PM[3:0]	Scanning Output Code PE[3:0]
0	0000		
1	0001		
2	0010		
3	0011		
4	0100		
5	0101		
6	0110		
7	0111		
8	1000		
9	1001		
A	1010		
B	1011		
C	1100		
D	1101		
*	1110		
#	1111		

6.4.2 Key Decode

Once a unique code for a key is identified, it must be decoded to provide meaning to the key. For example, your code may have identified that key 11101110 has been pressed, but what does that mean to the program or in the application context? For this part of the lab you will take the identified key and decode to its binary value, as shown in table 6.3. Note this could just as easily be the ASCII character code.

Milestone 6.2 Write a function that take the unique binary key code as input (PM[3:0] PE[3:0]) and return the binary value from table 6.3. This binary value should be placed in in Keil memory for TA confirmation.
25 marks ■

6.4.3 LED Display

Configure four additional GPIO pins as outputs to externally display the decoded key's binary value.

Milestone 6.3 Extend the above programming to take the 4-bit binary value of the decoded key press and output to the LEDs. The LEDS should be ordered such that the TA can directly read the binary value. (The code for button 0 will not light any LED)
25 marks ■

6.4.4 BONUS: ASCII Code and LED Display

Milestone 6.4 Using your code from this lab's milestones, modify the decode component to be the 7-bit or 8-bit ASCII code of the button pressed and then output that binary value to LEDs. Ensure the order/value of the LEDs can be easily read by the TA.
10 marks ■

6.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone 3
4. Milestone Bonus

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 6.4: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

6.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

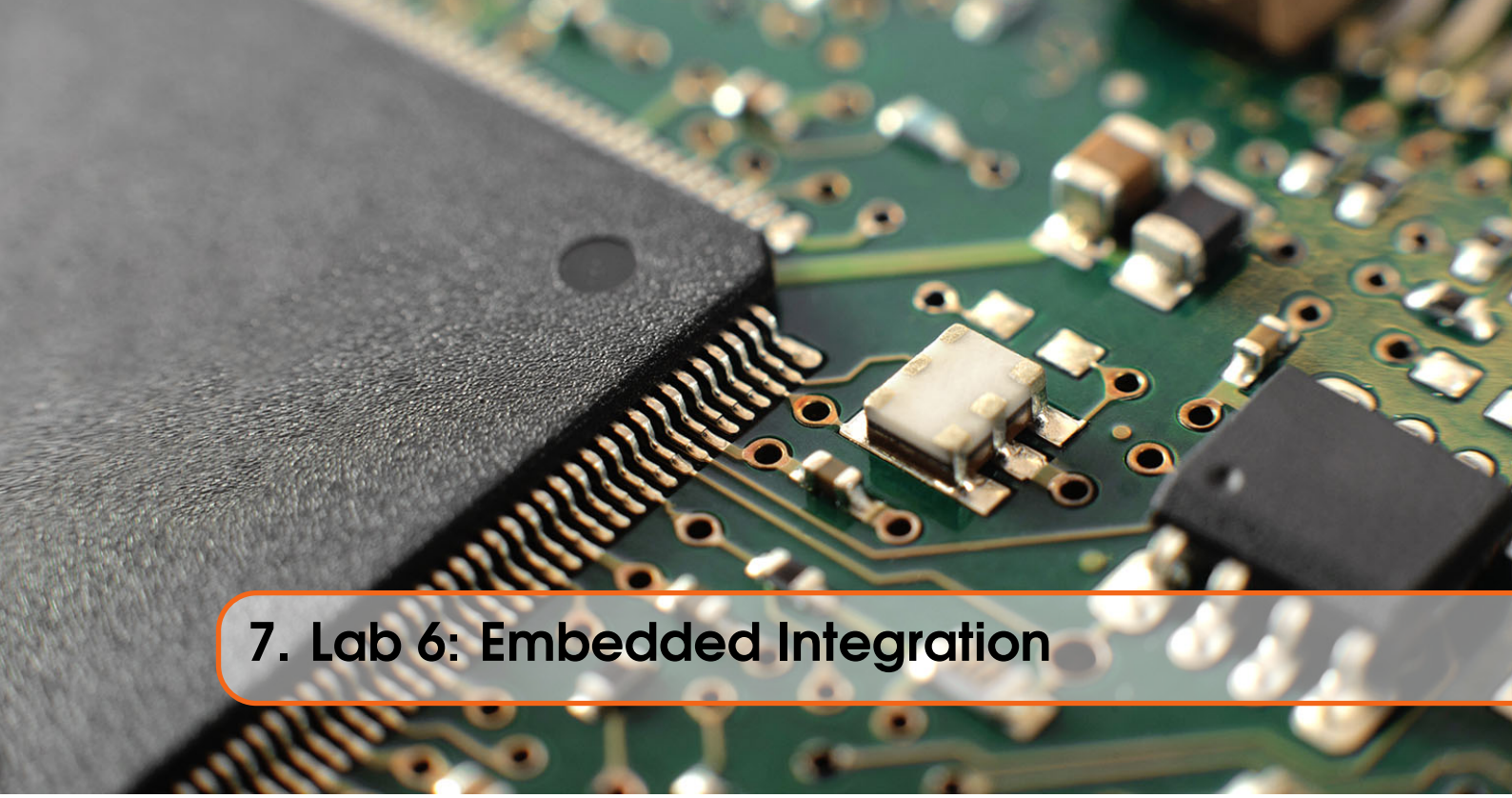
1. Your lab summary has a cover page clearly indicating the lab title, date, each name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

6.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	25	
Milestone 1	25	
Milestone 2	25	
Milestone 3	25	
Milestone Bonus	10	
Total	**100**	
Deductions		
Final Score		



7. Lab 6: Embedded Integration

7.1 Objective

The objective of this lab is to start integrating embedded components with the goal of a functional system. In this lab we will be combining your previous labs and studios to create a keypad controlled stepper motor. This lab will also help address several of your project milestones.

7.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Lectures, labs, and studios from weeks 0-5.

7.2 Pre-Laboratory Preparation

[25 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time.

1. If a stepper motor were to have 36 steps per rotation, how fast would the motor turn with a 10ms delay per step? Show calculations and answer in RPM. You may assume memory access time etc. are negligible. [5 marks]
2. If a stepper motor were to have 200 steps per rotation, calculate the delay per step for the motor to spin at 60 RPM. You may assume memory access time etc. are negligible. [5 marks]
3. Draw the flowchart for milestone 2. You may assume the keypad decoding is done by a separate function that returns the key code. [15 marks]

7.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

Table 7.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
4x4 Keypad	Grayhill 96BB2-006-R
Stepper Motor	RB-Vel-133

7.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

Table 7.2: Pins and Ports for Lab 6

Port	Pins	Application
Port L	3:0	Keypad scanning (output)
Port M	3:0	Key press decode (input)
Port H	3:0	Stepper motor control
Port G/D	G[1:0] D[3:1]	LCD DCS, Reset, Communication via SSI2 (Bonus)
Port A	1:0	Virtual com port UART (Bonus)

7.4.1 4x4 Keypad Button Identification

In Studio you have already completed 1-button, 1-row/column, and a 2x2 keypad decoding. Last lab you extended that work to identify which button was press on the full 4x4 keypad. This first milestone is intended to get that keypad scanning routine running again so we can extend from it and integrate additional functionality. Refer to the pin map on figure 7.1 as an example of how to select pins from the microcontroller realestate and ensure pin functionality is not being overlapped (this is important for your project too!).

Milestone 7.1 Use the scanning technique C program from last lab to determine which key has been pressed, take the 4-bit binary value of the decoded key press and output to the *onboard* LEDs. The onboard LEDS should be ordered such that the TA can directly read the binary value. (The code for button 0 will not light any LED) according to table 6.3.

25 marks ■

7.4.2 Key Decode for stepper Motor Control

Similar to last lab, once a key is decoded its meaning is based on the context of the application. For this lab we will be remapping the keypad functionality. Table 7.3 defines the key mappings.

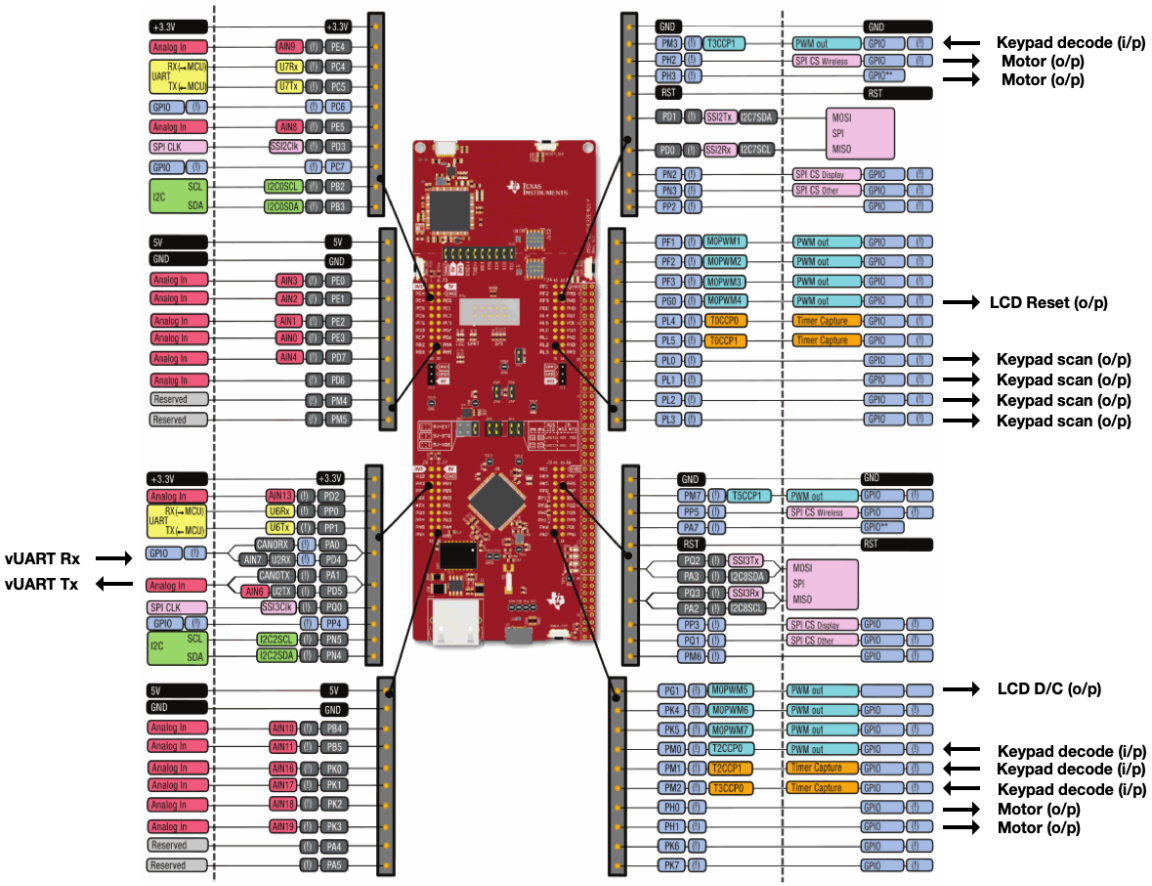


Figure 7.1: Pin Map for Integration

Table 7.3: Keypad Mapping

Key	Meaning
A-D	Angle of LED flash
*	Clockwise rotation of motor direction[1]
#	Counter-clockwise rotation of motor direction
1-9	Speed of motor steps / rotation (Start)
0	Stop and reset logic

Milestone 7.2 Write a C program that decodes a sequence of keypad entries. The sequence is: 1) Angle, 2) Direction, 3) Speed. Once Speed is selected the stepper motor should begin its rotation and continue until 0 is pressed. The 0 key will also be used to reset the sequence entry if a mistake has been made. An LED should be used to provide status/state information about the motor. A state-machine design approach is recommended.

Angle Throughout the rotation of the stepper motor it must flash an LED (different LED from status) at specified angles from its starting point (Setting/determining a home position is not required for this lab). You should use Wave Drive or Full Step mode for the motor. A/B/C/D = 11.25/45/90/360 degrees, respectively.

Direction Looking at the keyed motor shaft, the motor should turn clockwise or counter-clockwise. */# = clockwise/counter-clockwise, respectively.

Speed The speed selection will set the time delay between motor steps and also start motor rotation. The button value (1-9) shall be multiplied by 10ms for the delay between motor steps. For example, 1 = 10ms, 2 = 20ms, 3 = 30ms, etc.

45 marks ■

7.4.3 BONUS 1: Status Display via LED

Milestone 7.3 Using 7 external LEDs display the configuration values of the stepper motor (angle, direction, speed). Use 2 bits for the angle choice (A=00,B=01,C=10,D=11), 1 bit for the direction (clockwise vs. counter-clockwise), and 4 bits for speed of motor steps/rotation (1-9).

10 marks ■

7.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone Bonus 1

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

7.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

Table 7.4: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

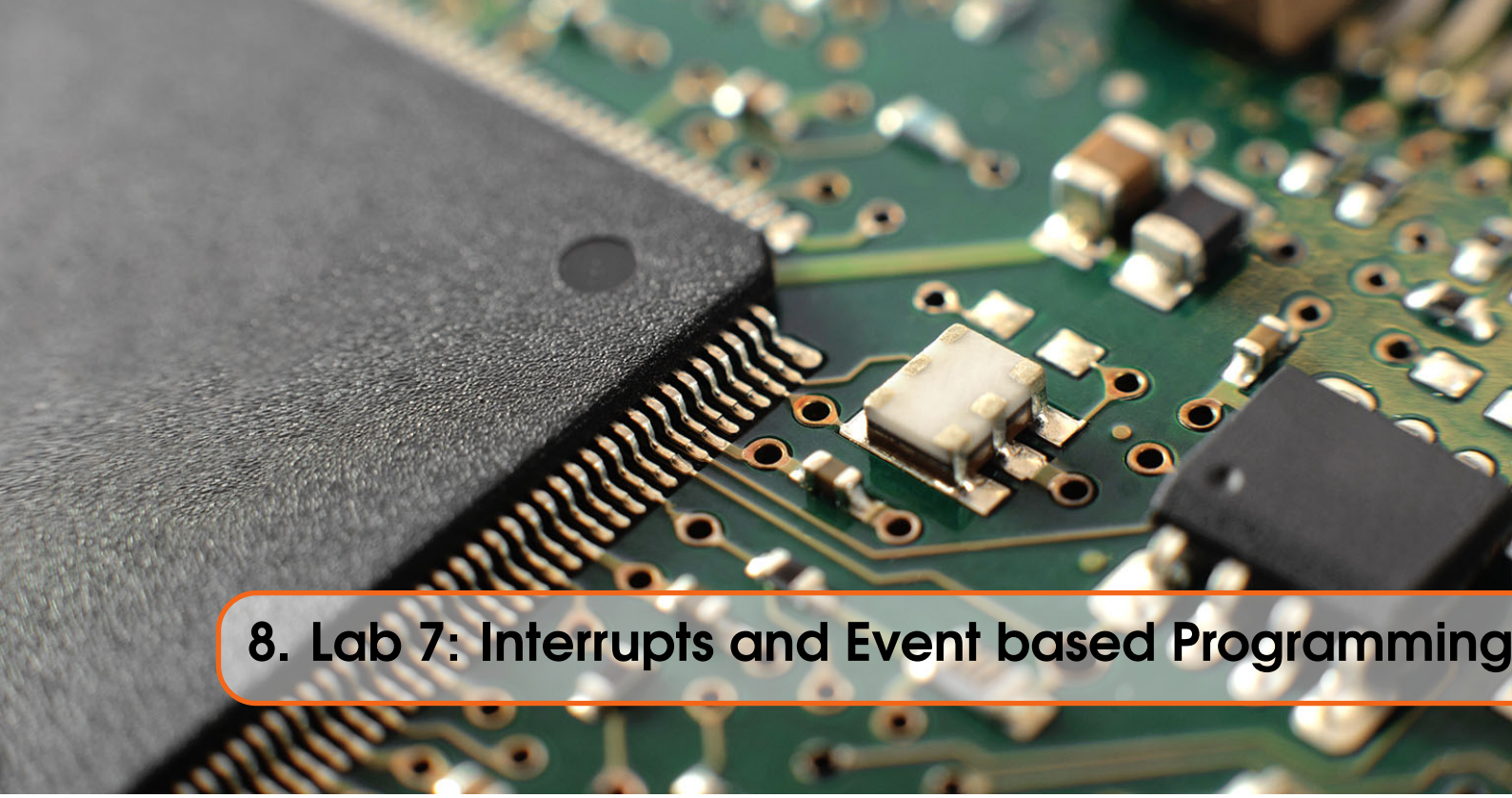
   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

7.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	25	
Milestone 1	30	
Milestone 2	45	
Milestone Bonus 1	10	
Total	**100**	
Deductions		
Final Score		

IV Part Four: Act



8. Lab 7: Interrupts and Event based Programming

8.1 Objective

The objective of this lab is to gain experience using event based programming.

8.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Chapter 9 textbook.
2. Week 7 Studios.

8.2 Pre-Laboratory Preparation

[25 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time.

1. In the context of connecting a button with GPIO, what do negative logic and positive logic mean? (See figure 9.9) [5 marks]
2. When referring to interrupt triggering via GPIO, explain the difference between falling edge, rising edge, low level, and high level. (Hint - see section 9.5) [5 marks]
3. What five conditions must be true for an interrupt to occur? [5 marks]
4. How do you enable interrupts? [5 marks]
5. What are the steps that occur when an interrupt is processed? [5 marks]

8.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

Table 8.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
ToF Sensor	Polo 3415 (VL53L1X Time-of-Flight Distance Sensor)
Software	Realterm or similar

8.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary truth tables and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

Table 8.2: Pins and Ports for Lab 6

Port	Pins	Application
Port B	3:2	I2C

8.4.1 Periodic Interrupt

In Studio you completed periodic interrupt triggering for an onboard LED. Reuse this code to flash an LED and output the same square pulse duration to a GPIO pin of your choice. Using a scope, verify the timing.

Milestone 8.1 Implement a periodic interrupt of 500ms. Write an interrupt service routine that will flash an onboard LED and output a 100ms pulse to a GPIO pin.

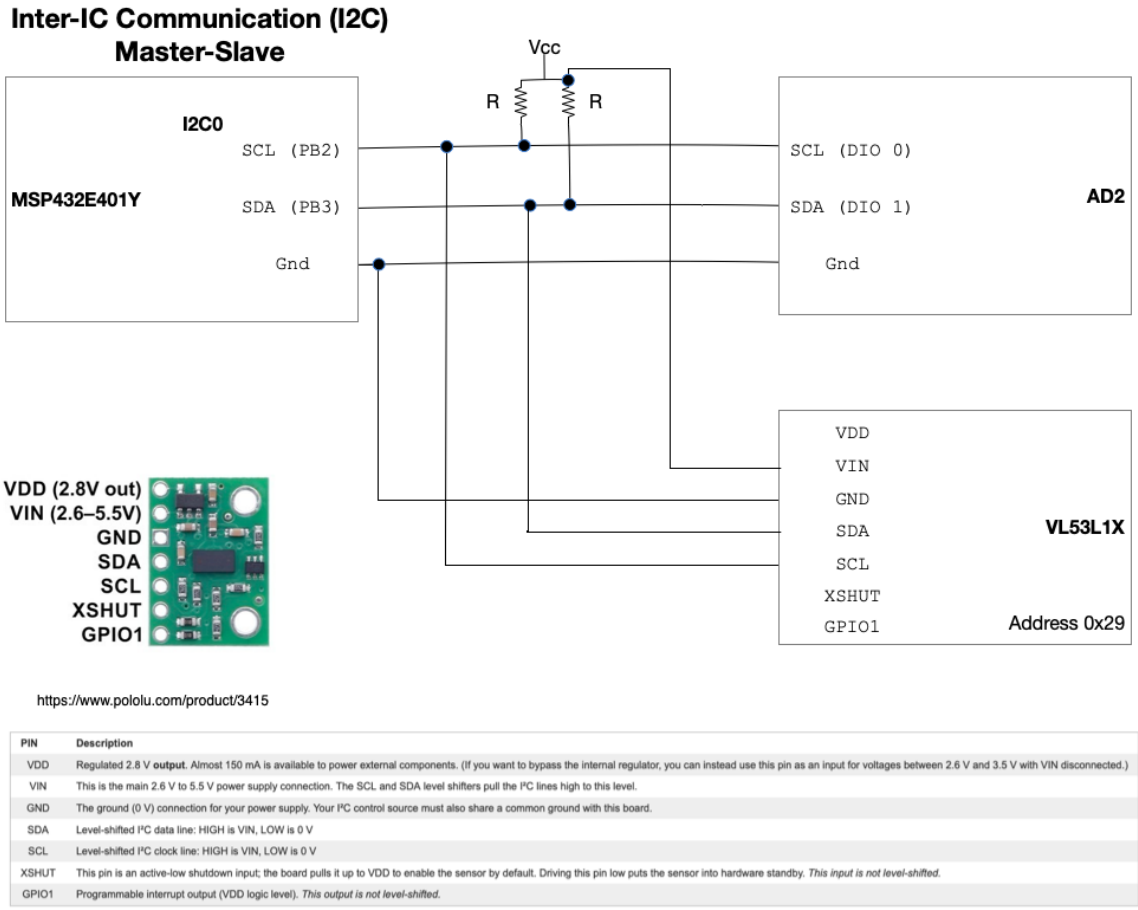
Verify the timing by connecting the GPIO pin to a scope probe. Ensure a photo goes in your lab summary, noting the pulse and interrupt periods.

35 marks ■

8.4.2 GPIO Interrupt

In Studio you completed GPIO push-button interrupt triggering for an onboard LED. For this milestone you will need to connect the Time-of-Flight sensor the same way you did in studio (you should also connect your AD2 for debugging in Logic spy mode).

Reuse studio code to generate an interrupt when the button is pressed. In the interrupt service routine transmit (using I2C) the slave device (Time-of-Flight) address (0x29) and one byte of data (0x00). Using a scope or your AD2, verify correct data transmission (you will need to decode transmission).



PIN	Description
VDD	Regulated 2.8 V **output**. Almost 150 mA is available to power external components. (If you want to bypass the internal regulator, you can instead use this pin as an input for voltages between 2.6 V and 3.5 V with VIN disconnected.)
VIN	This is the main 2.6 V to 5.5 V power supply connection. The SCL and SDA level shifters pull the I²C lines high to this level.
GND	The ground (0 V) connection for your power supply. Your I²C control source must also share a common ground with this board.
SDA	Level-shifted I²C data line: HIGH is VIN, LOW is 0 V
SCL	Level-shifted I²C clock line: HIGH is VIN, LOW is 0 V
XSHUT	This pin is an active-low shutdown input; the board pulls it up to VDD to enable the sensor by default. Driving this pin low puts the sensor into hardware standby. *This input is not level-shifted.*
GPIO1	Programmable interrupt output (VDD logic level). *This output is not level-shifted.*

Figure 8.1: Wiring layout for Lab 7

Milestone 8.2 Implement GPIO interrupt as noted above using the relevant studio code to transmit data to the ToF device and verify the data transmission using your scope or AD2.

40 marks ■

8.4.3 BONUS 1: Interrupt Buttons for transmission to PC

Milestone 8.3 Implement GPIO interrupts for two separate buttons (1 interrupt each). Button 1 transmits your student number via the UART (virtual com port, communication method covered in Studio) to the PC and display using an application (e.g., RealTerm, Python, etc.). Button 2 transmits the four characters: 2DX4.

10 marks ■

8.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone Bonus 1

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 8.3: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

8.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

8.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	25	
Milestone 1	35	
Milestone 2	40	
Milestone Bonus 1	10	
Total	**100**	
Deductions		
Final Score		

9. Lab 8: Collecting Distance Data

9.1 Objective

The objective of this lab is to communicate with, and collect data from, a digital sensor.

9.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. Review the provided code from Studio. It was written to support data collection from VL53L1X using the Ultra Light Driver.
2. I2C based upon MSP432E4 Reference Manual Chapter 19.
3. VL53L1X user manual.
4. Code implementation was based upon format specified in vl531X.pdf pg19-21
5. Prior lecture, labs, studio on I2C, UART, and programming paradigm (polling vs event).
6. Textbook, Chapter 11.6 for examples of system dataflow diagrams and modular design.

9.2 Pre-Laboratory Preparation

[35 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time.

1. Briefly describe how the VL53L1X ToF sensor works to obtain a distance measurement. [10 marks]
2. What units does the VL53L1X ToF sensor return? [5 marks]
3. How many bits is one distance measurement VL53L1X ToF sensor? [5 marks]
4. How many bits can be transmitted at once using UART serial communication? [5 marks]
5. How does the UART transmitter/receiver send/receive data that is bigger than one serial package will allow? [10 marks]

9.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 9.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
ToF Sensor	Polo 3415 (VL53L1X Time-of-Flight Distance Sensor)
Stepper Motor	RB-Vel-133
Software	Realterm or similar

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

9.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary designs (i.e., flow charts) and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

Table 9.2: Pins and Ports for Lab 8

Port	Pins	Application
Port B	3:2	I2C
Port H	3:0	Stepper motor control
Port A	1:0	Virtual com port UART

9.4.1 Read Time-of$_F$*lightSensorIDandModuleType*

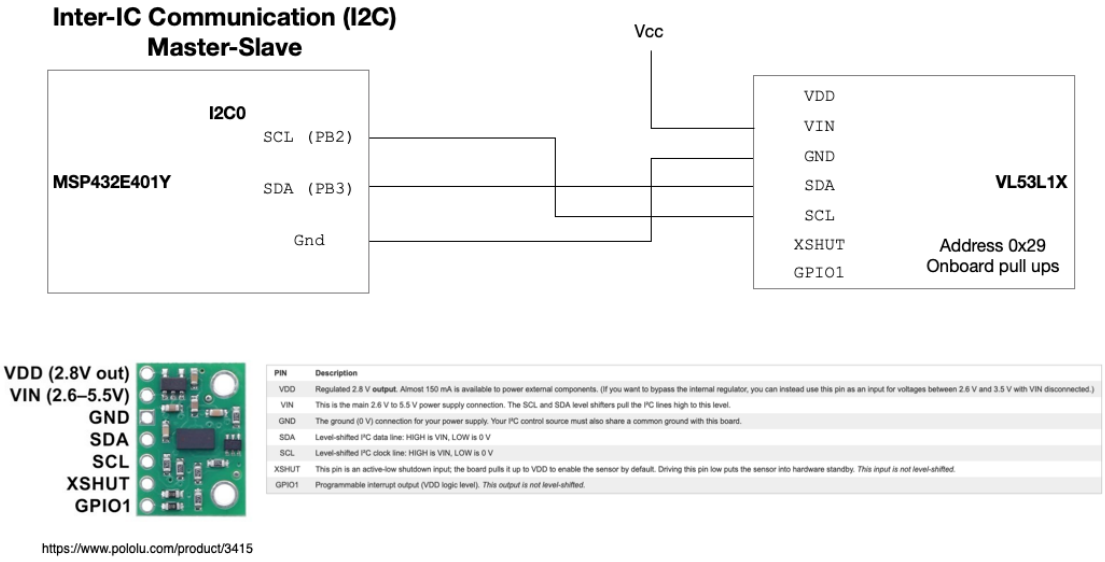
In Studio you communicated with the distance sensor to collect its ID via I2C. This this milestone you will extend that code to communicate the ID to a PC over UART. This is often the first test done with any I2C device to confirm it is on the bus and functioning.

Milestone 9.1 Implement a serial UART communication to transmit the VL53L1X model ID and module type. From the provided code, use the following three API function calls:

```
status = VL53L1_RdByte(dev, 0x010F, &byteData); //for model ID (0xEA)
status = VL53L1_RdByte(dev, 0x0110, &byteData); //for module type (0xCC)
status = VL53L1_RdWord(dev, 0x010F, &wordData); //for both model ID and type
```

The PC should receive 1-byte model ID, 1-byte module type, and both bytes from `wordData`.

Refer to the wiring diagram for this lab. Note the AD2 has been removed and the pull-up resistors are no longer needed because the VL53L1X has pull up resistor on board its PCB. **25 marks** ■



PIN	Description
VDD	Regulated 2.8 V output. Almost 150 mA is available to power external components. (If you want to bypass the internal regulator, you can instead use this pin as an input for voltages between 2.6 V and 3.5 V with VIN disconnected.)
VIN	This is the main 2.6 V to 5.5 V power supply connection. The SCL and SDA level shifters pull the I²C lines high to this level.
GND	The ground (0 V) connection for your power supply. Your I²C control source must also share a common ground with this board.
SDA	Level-shifted I²C data line: HIGH is VIN, LOW is 0 V
SCL	Level-shifted I²C clock line: HIGH is VIN, LOW is 0 V
XSHUT	This pin is an active-low shutdown input; the board pulls it up to VDD to enable the sensor by default. Driving this pin low puts the sensor into hardware standby. *This input is not level-shifted.*
GPIO1	Programmable interrupt output (VDD logic level). *This output is not level-shifted.*

Figure 9.1: Wiring layout for Lab 8

9.4.2 Time-of-Flight Sensor Measurement via I2C

You have been supplied code for communicating with the Time-of-Flight VL53L1X sensor. This is a digital module that we will require the use I2C for communication and control. The code provided is based upon the manufacturer API for use of the ultra light drivers[1].

The API functions provided to you are:

1. `VL53L1X_BootState`
2. `VL53L1X_SensorInit`
3. `VL53L1X_StartRanging`
4. `VL53L1X_CheckForDataReady`
5. `VL53L1X_GetDistance`
6. `VL53L1X_ClearInterrupt`
7. `VL53L1X_Stop`

The Studio sample code provided uses a polling approach to obtain measurement. In Studio you captured data using the ToF VL53L1X sensor. In prior labs you have implemented both UART and stepper motor code.

Milestone 9.2 Combine the ToF VL53L1X sensor with prior stepper motor code to capture 8 (45-degree intervals) or more distance measurements from a single 360-degree horizontal scan.

[1] See: UM2510 A guide to using the VL53L1X ultra lite driver `https://www.st.com/content/ccc/resource/technical/document/user_manual/group1/63/e0/7d/f1/0e/52/4d/cd/DM00562924/files/DM00562924.pdf/jcr:content/translations/en.DM00562924.pdf`

Convert these 8 measurements to an x,y coordinate and plot using a software program like Excel

40 marks ■

9.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 9.3: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

9.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

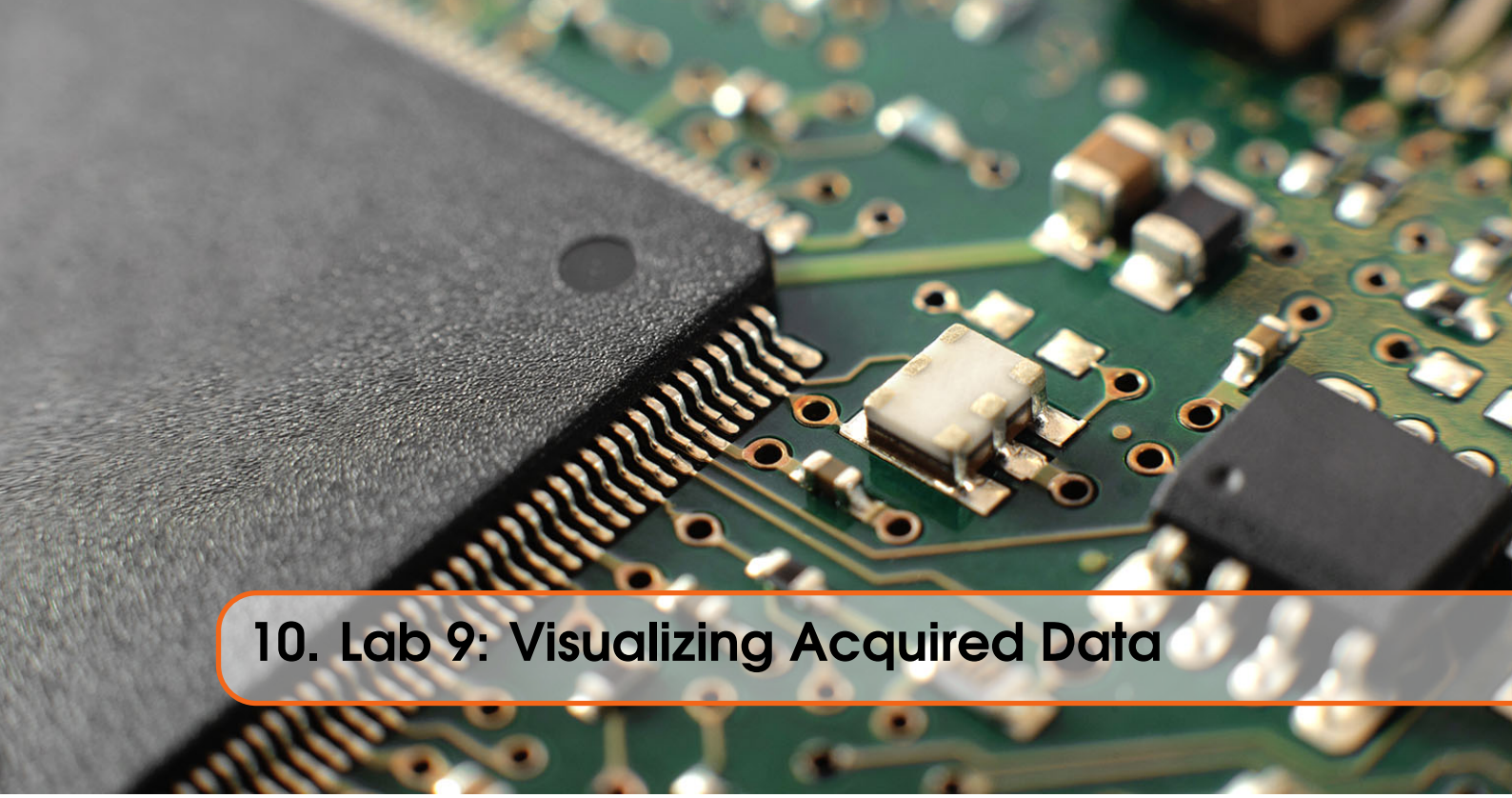
1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

9.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	35	
Milestone 1	25	
Milestone 2	40	
Total	**100**	
Deductions		
Final Score		



10. Lab 9: Visualizing Acquired Data

10.1 Objective

The objective of this lab is to acquire sensor data and then visualize that data.

10.1.1 Resources

You should review the following resources to prepare yourself for this lab and the future labs in this course:

1. pySerial documentation (`https://pypi.org/project/pyserial/`)
2. Open3D documentation (`http://www.open3d.org/docs/release/`)
3. Studio wA.
4. Lecture W9 & WA.
5. Lab 9.

10.2 Pre-Laboratory Preparation

[35 marks total]

All laboratories require a prelab to be submitted at the beginning of the lab. In order to obtain credit for the prelab it must be submitted within the first 15-minutes of lab time.

1. Define the xyz file format used for the Python Open3D library. [10 marks]
2. Draw the algorithm flowchart for taking ToF measurement data and creating an x,y,z co-ordinate. Proper flowchart symbols and correct use of those symbols must be used. [15 marks]
3. Briefly explain how you will synchronize the ToF measurement data with the Python program. (hint: this was discussed in Studio exercises) [10 marks]

10.3 Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory:

Table 10.1: Integrated Circuits and Components for Lab 1

Identification	Description
ARM Cortex-M4F	TI MSP432E401Y Microcontroller
ToF Sensor	Polo 3415 (VL53L1X Time-of-Flight Distance Sensor)
Stepper Motor	RB-Vel-133
Software	Python 3 with PySerial & Open3D

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

10.4 Laboratory Exercises and Milestones

Read the following experiment and study the circuits as shown. Pre-filling your summary with the necessary designs (i.e., flow charts) and structuring your submission such that you only need record experimental output will allow you to focus on the milestones.

ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA – THERE ARE NO MARKS AFTERWARDS.

Table 10.2: Pins and Ports for Lab 9

Port	Pins	Application
Port B	3:2	I2C
Port H	3:0	Stepper motor control
Port A	1:0	Virtual com port UART

10.4.1 Time-of-Flight Sensor Measurement to Python

In Studio you used PySerial to communicate between the microcontroller and the PC. In this lab milestone you will be extending last lab by communicating ToF data to the PC using PySerial.

Recall in the last lab, you combined the ToF VL53L1X sensor with the stepper motor to capture 8 (45-degree intervals) or more distance measurements from a single 360-degree horizontal scan. You then converted these 8 measurements to an x,y coordinate and plot using a software program like Excel.

Milestone 10.1 Implement a serial UART communication to transmit one horizontal scan (8 measurements) of ToF measurement data to the PC via PySerial.

Your Python code should then translate your measurements to an x,y,z coordinate and output to a file called `tof_radar.xyz`. This file should be in the XYZ format (example in W9 lecture).

40 marks ■

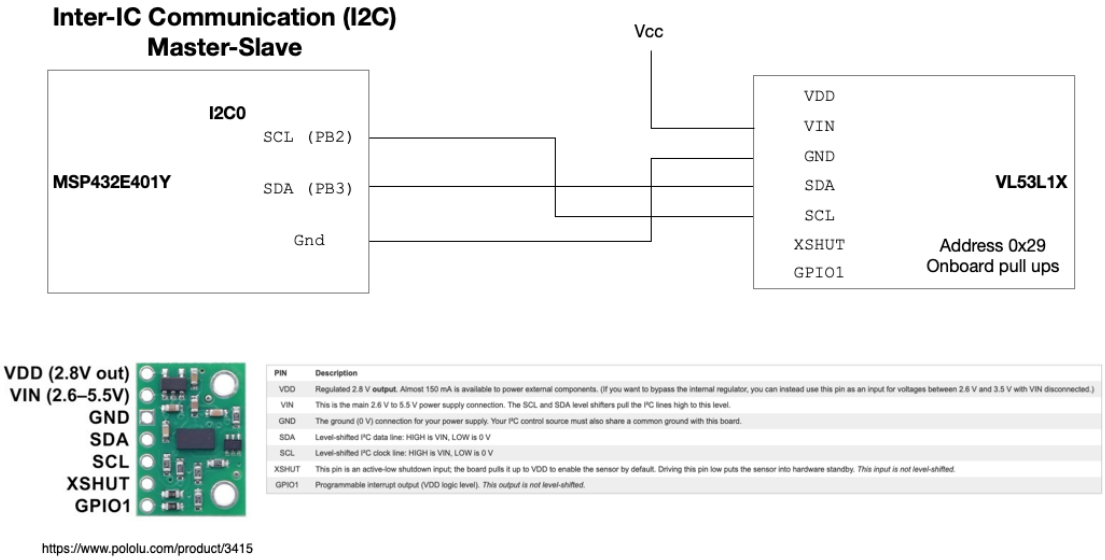


Figure 10.1: Wiring layout for Lab 9

10.4.2 Visualize Time-of-Flight Measurement Data

Once the xyz coordinate file has been written we now have a small example of a point cloud. A point cloud is a spatial sampling of a space or object. Point clouds can be used to generate 3D representations of the space or object. For this milestone we will use the Open3D Python package to convert our coordinates into a horizontal slice of a 3D space. This would be similar to a radar sweep or a lidar scan used in navigation. By using repeated horizontal scans, a much better representation of the space would result.

Milestone 10.2 Employing the Open3D example from W9 lecture, create a plot of the xyz data with lines connecting between each data point.

25 marks ■

10.5 Summary of Milestones & Evaluation Rubric

TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2

Be aware, if the TA does not have a record of the execution then a mark of zero is assessed.

Table 10.3: Evaluation Rubric

Criteria	Mark
Successfully demonstrate and explain to TA a fully correctly working milestone	100%
Demonstrate and explain a coherent attempt at milestone, but incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0

10.6 Submission Requirements

Please ensure you complete the following items *prior* leaving the laboratory:

1. Your lab summary has a cover page clearly indicating the lab title, date, name, and student number.
2. Cover page must also contain the following statement:

   > As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

3. Each exercise milestone has been checked by a TA.
4. Uploaded the lab summary to Avenue.

10.7 Grading Summary Table

Component	Weight	Grade
Lab Questions	35	
Milestone 1	40	
Milestone 2	25	
Total	**100**	
Deductions		
Final Score		

V

Appendix A

11. Getting to Hello

When starting out with any new programming platform, the first thing every developer must to do is install the new development environment and verify that it works. This is where the “Hello World!" code originated in the programming domain. This si done as a quick test to compile and run a basic set of instructions that communicate to the programmer. For embedded development, we do not (yet) have the initial luxury of an existing operating system that fully supports a display, keyboard, mouse, etc. For our first attempt at an embedded “Hello World!", we will assemble pre-tested code that will flash an LED.

Before installing any new software or making any modifications to your computer configuration you must back up all of your data. Backing up your data should be a regular and frequent habit that occurs independent of this course.

11.1 Getting Started...

To start, the development environment needs to be installed. In this course we will be developing using the Keil MDK (Microcontroller Development Kit). Similar to how you would use an IDE for software development, the Keil MDK allows you to create, build, and debug embedded applications.

To obtain Keil MDK 5 use this link `https://www2.keil.com/mdk5` and enter the following information:

- First & Last Name
- McMaster email address
- Company: McMaster Computer Engineering 2DX4
- Device: MSP432E401Y

Keil MDK only runs on Windows and is free for development under 32KB (thus free for this course). If you are running a Windows based computer, skip to section 11.5. If you are running a macOS based computer then you will need to choose either Virtualization (refer to section 11.2) or

Bootcamp (refer to section 11.3) to install Windows.

How do you choose between Virtualization or Bootcamp?

Choose **Virtualization** if your computer has an i5 or better processor, at least 8GB RAM, and 50 GB or more of free disk space.

Choose **Bootcamp** if your computer does not meet the recommended specification for Virtualization. Each virtualization software will have a minimum recommended specification which you can research further if interested; however, the above specifications are recommended from experience.

Choose **Cloud Virtualization** if you're willing to try one of the could based virtualization options. As a McMaster University student you have access to the Microsoft Campus agreement, which offers you a 12-month free Azure cloud virtualization of Windows[1].

11.2 macOS – Virtualization

There are three primary options for virtualization software (listed alphabetically):

Oracle VM VirtualBox `https://www.virtualbox.org`
Parallels Desktop `https://www.parallels.com`
VMware Fusion `https://www.vmware.com/ca/products/fusion.html`

You are free to select any one of these options, but note that only Oracle VM VirtualBox is free. The others may be available through the campus computer store with an educational license for purchase. In all cases, the choice and configuration are the student responsibility. We will help when and where we can on a best effort basis. It is imperative that you sort out your development environment install, configuration, and testing as soon as possible.

We have selected Oracle VM VirtualBox for our 2DX4 development environment[2].

Follow the manufacturer installation procedure for the virtualization software. Once the virtualization software is installed, you will need to configure a virtual computer for a Windows installation. Typically, once you configure the virtualized client you point the application to your Windows install drive/file/ISO and proceed as a normal operating system installation. For our development environment we have chosen a 64-bit Windows 10 Professional installation. Finally, once Windows is installed then install any VM support tools to improve macOS integration (in VirtualBox you will need to install Guest Additions from the Devices menu).

[1] "Free" comes with some implicit risks and financial limits ($260 credit). Accessing this service requires a credit card and if you intentionally/accidentally upgrade your service then it will incur a charge. If you exceed your credit then it will incur a charge. If someone hacks your account you are responsible, and yes that means you will likley incur a charge. From personal experience on a different cloud service provider, it is important to shut down resources you are not actively using and always watch your billing. In all cases cloud, protect you account because this is no different than putting your personal computer on the internet.

[2] VMware Fusion was tested, but failed to connect to the microcontroller board. This was due to either to our incorrect configuration or VMWare's handling of USB connectivity under the newly released macOS Catalina. Cursory research found mention of the issue possibly being related VMware/macOS having some issues around USB 2 devices.

11.3 macOS – Bootcamp

Bootcamp is a boot loader software application. This type of application runs prior to an operating system booting. Depending upon configuration, this can either autoboot to the last OS used, or a menu can load for a choice of which OS the start. This is implemented by the boot loader choosing which disk and/or partition is considered the target. While conceptually logical, the practical consideration is that when booting into the “target", the “other" OS disk/partition becomes unavailable. This can be problematic because files and disk space that are available using one OS are not available using the other (an online drive can alleviate this, or a properly configured shared drive/partition).

In order to install a boot loader, like Bootcamp, a software application will need to make significant changes to your hard drive organization (i.e., modifying partitions, partition tables, and boot records). A power failure, program interruption, reset, etc. will likely result in a non-bootable computer and a **loss of all data.**

The following link describes the method of installing Bootcamp: `https://support.apple.com/en-ca/HT201468`

11.4 Cloud – Virtualization

At this time, this method is experimental and should only be considered for curiosity and possibly simulation.

This option is convenient because you can save time setting up the VM. This option is experimental so it may not work as expected and is obviously limited to the quality of the internet connection. Refer to the following documentation for setting up an Azure cloud VM: `https://docs.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal?toc=/azure/virtual-machines/linux/toc.json`.

One hurdle that remains unsolved – connecting USB. This will depend on the Remote Desktop Client used. To date, using Microsoft Remote Desktop application was not successful. Azure offers Bastion as an option for private RDP. It might also be possible to install VNC client/server. This section will be updated when possible.

11.5 Windows

Install Windows, run all updates, and then plug the MSP432E401Y board. Figure 11.5 notes the correct USB port with which to connect.

Once installed, start Windows and go to `VirtualBox VM -> Devices -> USB` and select the MSP432E401Y to attach the TI board to Windows.

11.5.1 Keil MDK

Steps to installing the MDK:

1. Install Keil MDK 5.28, check for updates of included software packs and update if necessary. Refer to `http://www.ti.com/lit/ug/slau590j/slau590j.pdf` and see step 1.
2. Install the most current Device Family Pack (DFP) for the MSP432E401Y. The file is named `TexasInstruments.MSP432P4xx_DFP.<version>.pack` where the current version is 3.2.6.

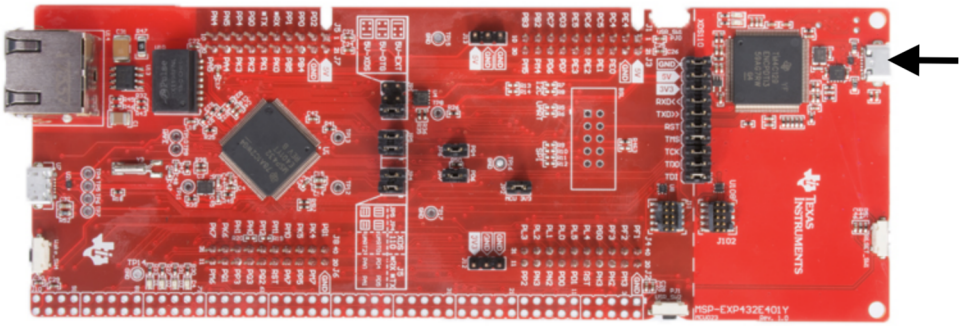


Figure 11.1: Top view of MSP432E401Y board with USB Debug location noted

3. Once Keil and DFP are installed, look under Examples, choose `BlinkLED -> Copy`. This will copy the project from the reference to your local development directory and also give the option to start the Keil MDK.
4. Finally, the correct pronunciation of this application's name is "KYLE".

11.5.2 USB Debug Firmware

To connect the MDK to the MSP432E401Y we will need to update the debug firmware to use the onboard debugging hardware, this is the XDS110 – see `http://www.keil.com/appnotes/files/apnt_309_v1.0.pdf`.

Download and install `ti_emupack_setup_8.3.0.00003_win_64.exe`. This will extract the firmware update application. Read XDS110SupportReadMe.pdf, and do the following (note, by plugging in the USB cable to the board you have already connected the "XDS110 debug probe"):

Updating the Firmware Using xdsdfu

To program the firmware, follow these steps:

1. Plug the XDS110 debug probe into your computer. Make sure that you only have one XDS110 class debug probe plugged in. The xdsdfu program will attempt to flash the first XDS110 debug probe it finds.

2. Run the following two commands from directory with xdsdfu:

   ```
   xdsdfu -m
   xdsdfu -f firmware.bin -r
   ```

   You may need to pause after the first command to give the OS time to recognize that the XDS110 has reconfigured as a different USB device.

Once the second command has completed, the firmware is updated, and the XDS110 should be ready to use.

Figure 11.2: Excerpt from XDS110SupportReadMe.pdf regarding debug firmware update

When the firmware update is complete, please pay special attention to the second last sentence "You may need to pause…" because after the firmware is updated, Windows sees a new USB device and as a result disconnects the "old" device TI board. You will now need to reconnect the board to your VM as demonstarted in figure 11.5.2.

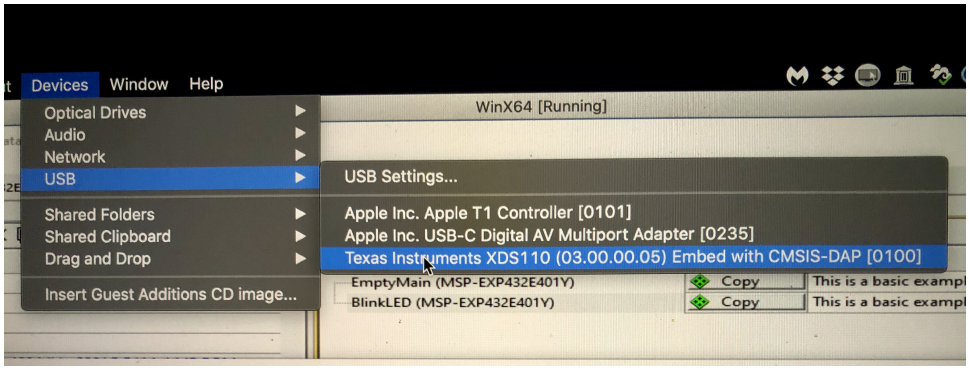


Figure 11.3: Re-Adding the MSP432E401Y's XDS110 as a new device under VirtualBox

11.5.3 Configure MDK Flash Tools

Select the Flash menu, then Configure Flash Tools (we are not using the ULink). We are using the MSP432E401Y's onboard XDS110, which requires the selection of CMSIS-DAP Debugger – see figure 11.5.3.

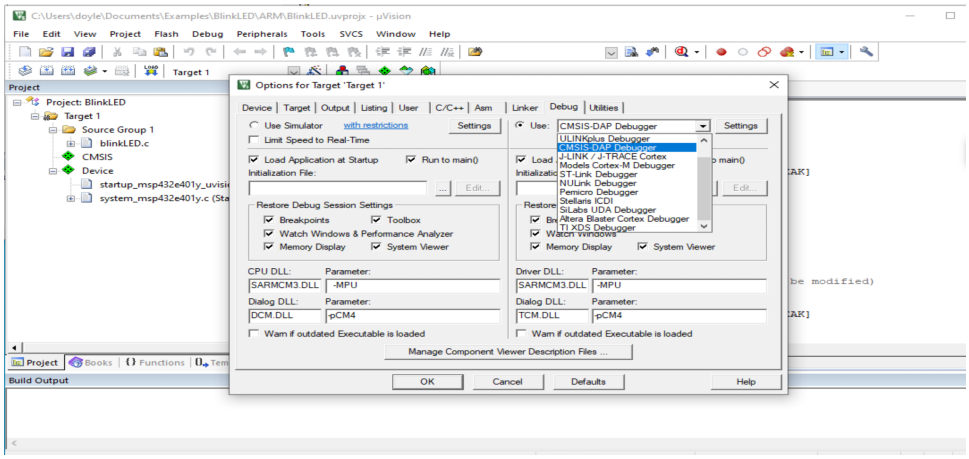


Figure 11.4: Configure Flash Tools configuration

Select Settings as shown in figure 11.5.3 and select OK.

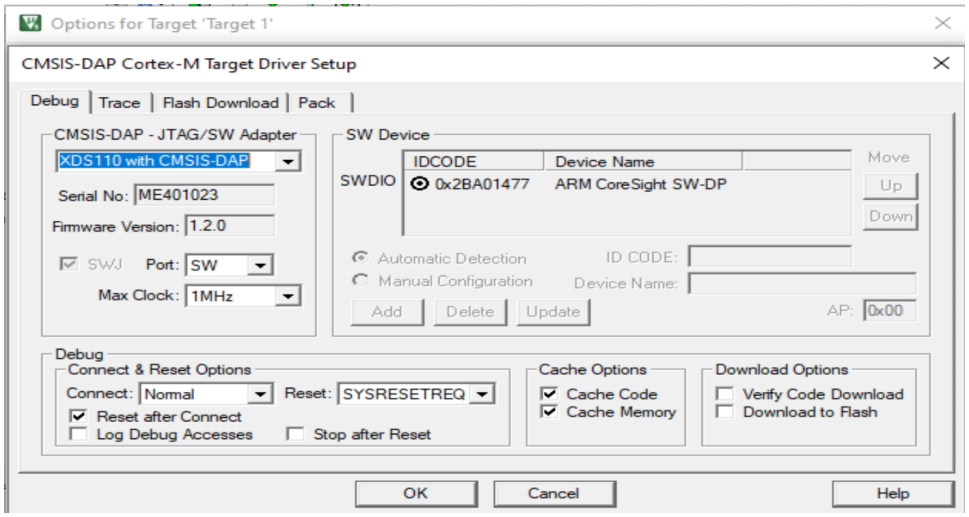


Figure 11.5: Configure Flash Tools settings

11.6 Say Hello

Return to Keil MDK and build the blinkLED.c code that you opened prior. Assuming no errors (there should be none), now select Load. The Build Output should indicate the Flash Load finished.

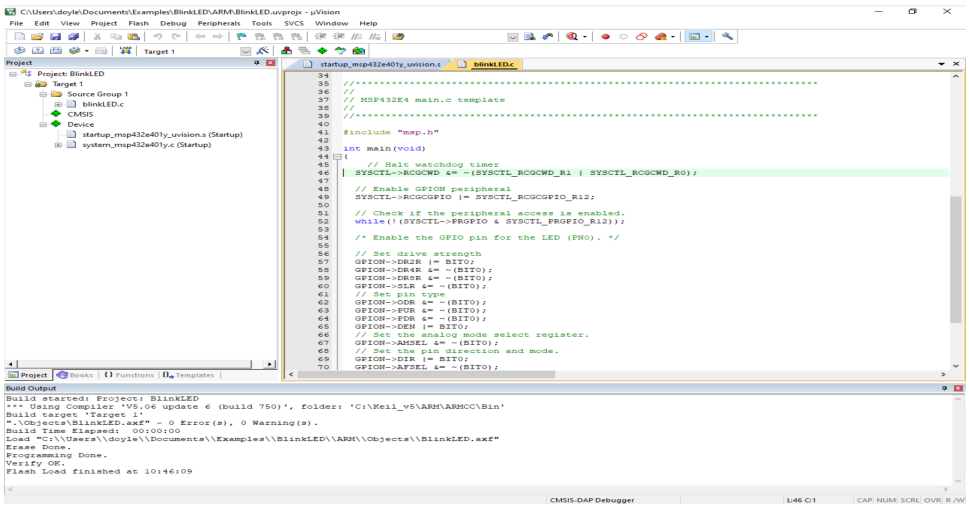


Figure 11.6: Successful build, load, and flash of blinkLED onto MSP432E401Y

Press `Ctrl+F5` or select from the menu `Debug -> Start/Stop Debug Session`. The code can now be stepped/run and memory/registers inspected (`F5` to run).

Running the code should flash (a.k.a say Hello with) the onboard LED. See figure 11.8 for a video showing the expected result[3].

R Finally, to be consistent with the textbook we will modify J4 and J5 jumpers. Before modifying the jumpers, ensure the USB cable has been unplugged from the MSP432E401Y.

[3]This media file requires Adobe Reader PDF viewer to run. To stop Adobe from repeatedly asking for permission to run media content change Adobe Reader's preference to allow `Enable playing of 3D content"`

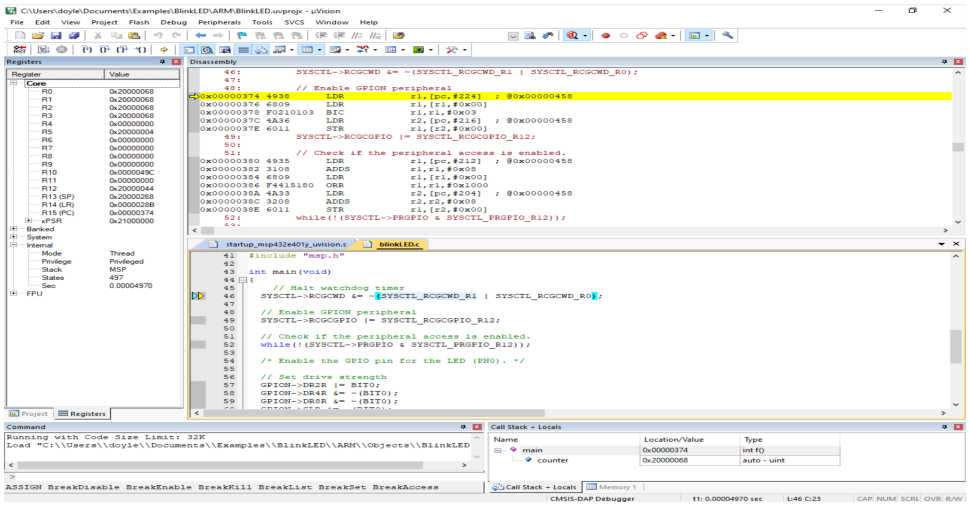


Figure 11.7: Keil MDK in hardware debug environment for blinkLED on MSP432E401Y

Now, place J4 and J5 in the UART2 position so PA1 and PA0 are connected to the PC as a virtual com port. Reconnect the USB cable.

Figure 11.8: Flashing LED from correctly assembled and loaded “blinkLED.c"

11.7 Software Version Summary

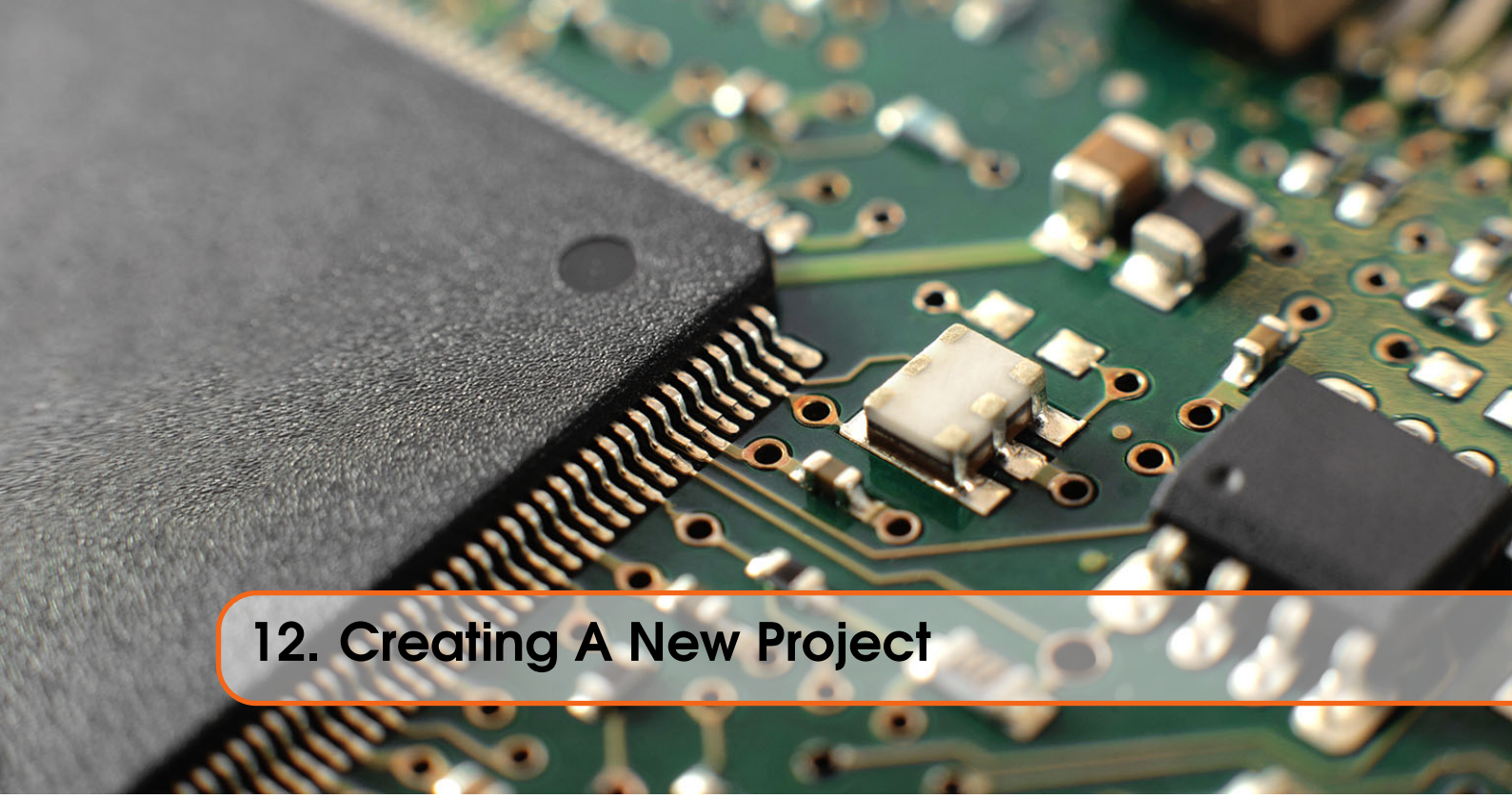
For these instructions we used the following software and version:

- Keil MDK 5.28 with `MSP432E4_DFP 3.2.6`
- macOS 10.15
- Microsoft Windows 10 Professional

- Oracle VM VirtualBox 6.0.14 (with Guest Additions installed after Windows)
- VMware Fusion 11.5 (with VMware Tools installed after Windows)
- MSP432E401Y firmware for XDS110 8.3.0.00003 (`ti_emupack_setup_8.3.0.00003_win_64.exe`)
- Keil sample project “blinkLED.c"

VI

Appendix B



12. Creating A New Project

Once you have confirmed your development environment is working, the next step is to create your own project.

12.1 Create a New Project from Scratch

12.1.1 C Language

To describe the steps of creating a new Keil project in the C language, please follow the link below.
`https://youtu.be/ZLzT955KfGY`

12.1.2 Assembly Language

To create a new Keil project follow these steps, similar to the C language:

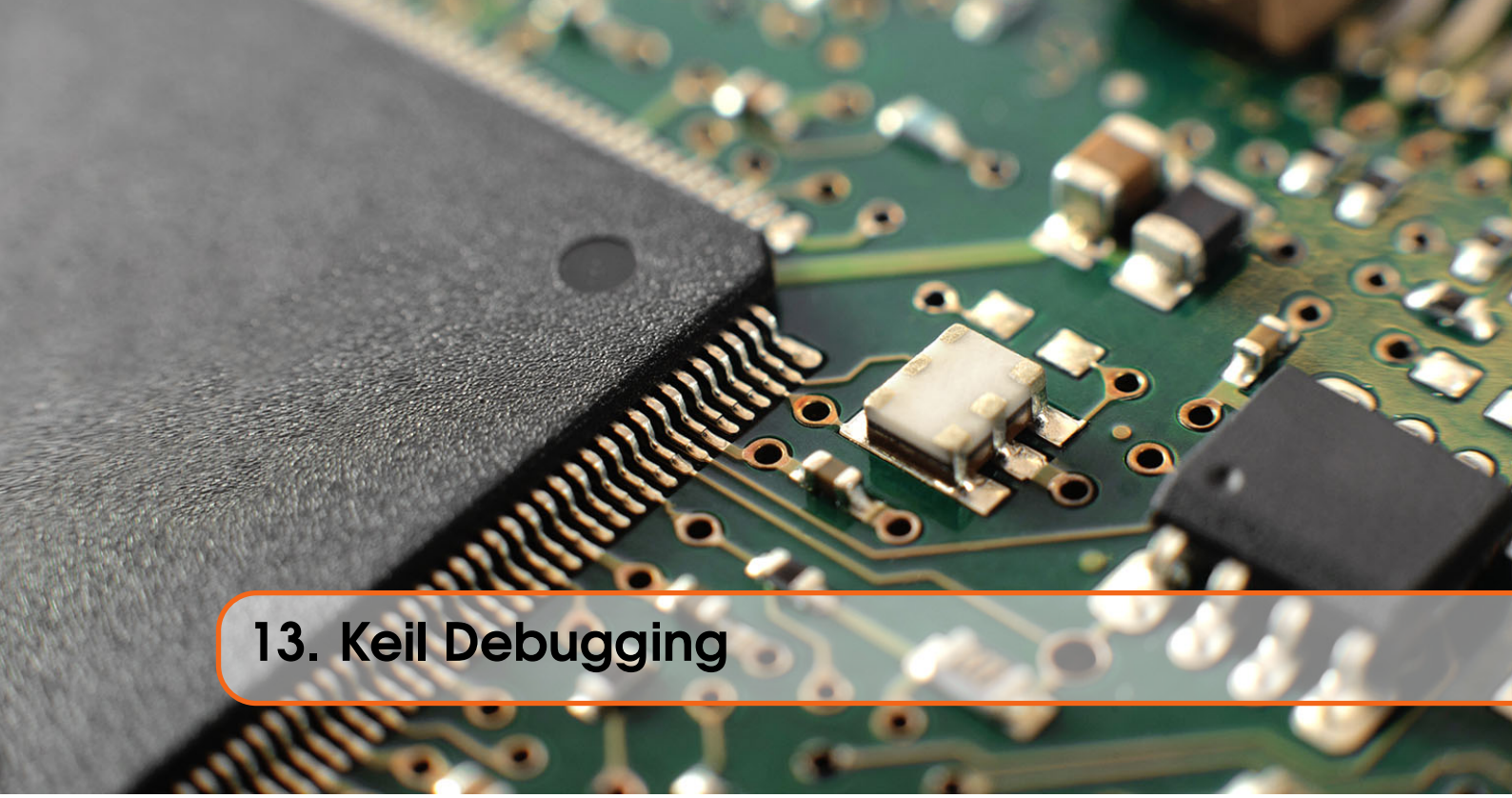
1. Project > New uVision Project When prompts for target device select MSP432E401Y
2. In the next window go to CMSIS check core and close window
3. In the project window, click plus beside Target to reveal source group 1, right click source group 1,in the menu that opens click add existing files to source group 1
4. Add `Startup.s` and add `gpio.s` and close (in this example, `gpio.s` is your “main")
5. Build target
6. Flash to the board

The assembly files `Startup.s` and add `gpio.s` can be downloaded directly from `https://www.ece.mcmaster.ca/~doyle/2DX4_Code/L0/`.

A short video demonstrating the steps to create a new Keil project in assembly language can be found here: `https://youtu.be/wxs56Hwxxcw`.

VII

Appendix C



13. Keil Debugging

The Keil IDE offers a complete set of debugging tools. The following are steps for specific debugging tasks in 2DX4 labs.

13.1 Exporting Data From the Keil IDE

One of the challenges when recording large amounts of data into microcontroller memory is verification. This relates directly to functional and performance dubugging.

The Keil Debug IDE offers numerous tools for inspection and the ability to customize inspection through setting Watches on variable and data structures, but also by using the Command Line. The following steps will allow you to export debug data:

1. Connect the microcontroller to computer
2. Load project
3. Configure Options for Target (proper debug hardware configuration)
4. Ensure the variable you wish to inspect is going to be in memory upon completion of your program. The easiest way to do this is to define the variable as global scope.
5. Translate, Build, Load.
6. Start Debug Session.
7. Before pressing Run, set Watch for variable(s) of interest. To do this, using your mouse hover over the variable in the source code and right click, then select "Add [variable] to...", then choose Watch 1. If this is an array and you want inspect all elements then modify the Watch 1 window to remove the [index]. Press + to see all array elements.
8. At the top, select View menu and then select Periodic Window Update to have the IDE update output windows (such as Watch).
9. Run.
10. Upon conclusion you will see your variable(s) of interest in the Watch window for inspection.
11. To export the variable(s) we can look into the Command window operations, like LOG, EVAL, D, and FUNC to allow easy inspection and capture of the data. Try the following:

```
Command Line:              Comment:
LOG > demo.txt             Starts a log file called demo.txt
EVAL func_debug           Evaluates variable - in this case this is a
                               pointer so just provides address of index [0]
EVAL func_debug[0]         Presents contents as hex and decimal
EVAL func_debug[1]            ... helpful but not for large quantity of data
D func_debug[0],func_debug[99]  Displays memory contents which are
                                      stored little endian
LOG OFF                    Ends a log file
```

12. These are helpful, but we can make better use of the debug capabilities. Looking at FUNC and .ini files we can more easily extract meaningful data. FUNC will permit us to write ANSI C style functions and .ini files permit us to save as a script. To write a script, at the top select the Debug menu, and then select Function Editor. An example is shown below of a FUNC that logs 100 array elements.

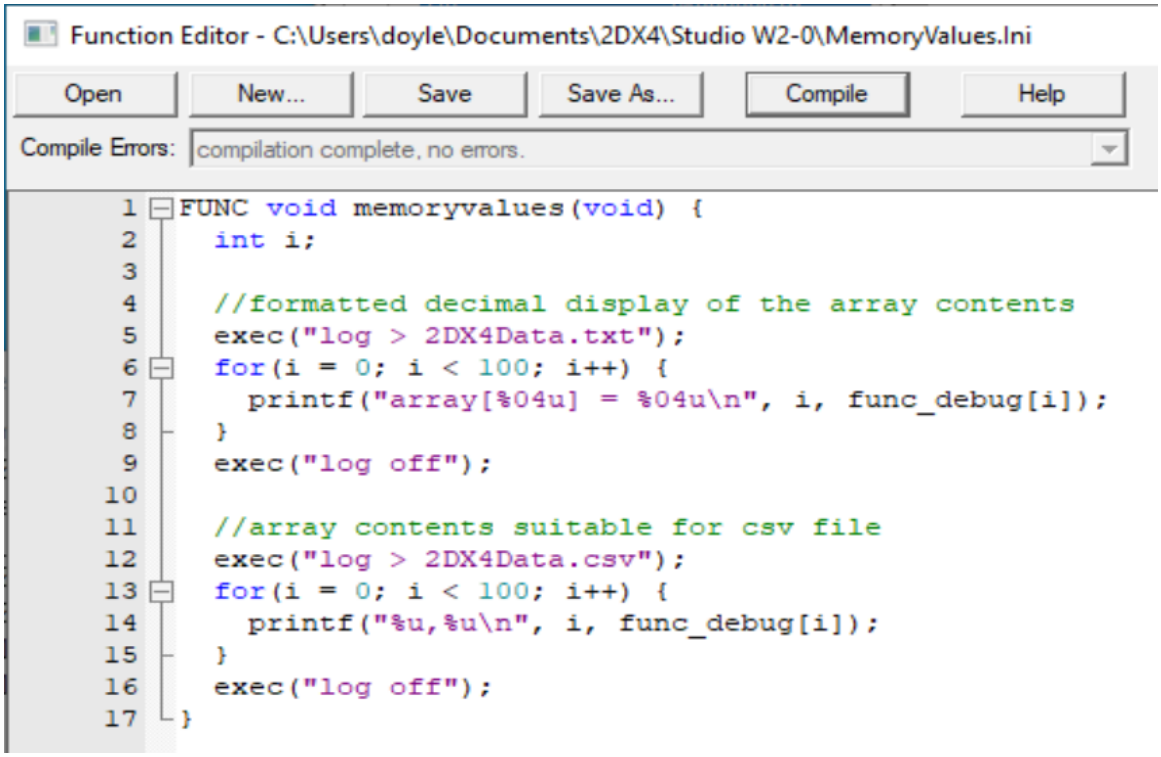


Figure 13.1: Sample .ini script illustrating ANSI C style function for Keil debugging

13. Once the program is complete and compiled, the function can be invoked in the command line:

```
memoryvalues()
```

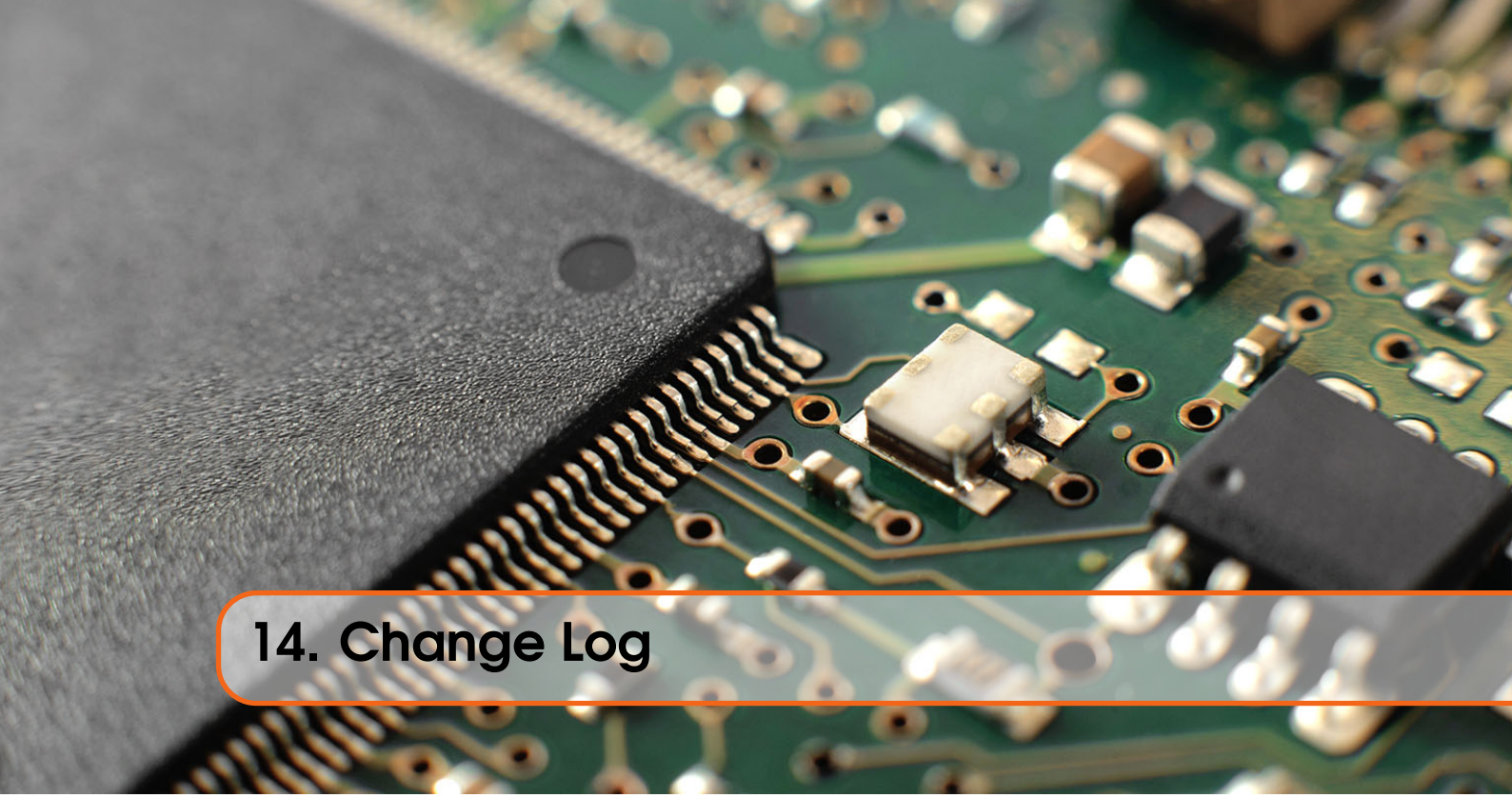
For more information, refer to the Keil Command Line resource here: `http://www.keil.com/support/man/docs/uv4/uv4_debug_commands.htm`.

More details on the commands listed above:

D Display: `http://www.keil.com/support/man/docs/uv4/uv4_cm_display.htm`
LOG LOG: `http://www.keil.com/support/man/docs/uv4/uv4_cm_log.htm`
EVAL EVALuate: `http://www.keil.com/support/man/docs/uv4/uv4_cm_evaluate.htm`
FUNC FUNC: `http://www.keil.com/support/man/docs/uv4/uv4_cm_func.htm`

VIII

Appendix D



14. Change Log

This appendix will record the changes made to this document in chronological order.

January 5, 2021 Lab manual issued with Labs 1-9.
January 25, 2021 Lab 1, milestone 2 clarified to be done in assembly.
February 5, 2021 Lab 2, Milestone 1 changed to use a 3 bit code instead of 4.
February 5, 2021 Lab 2, Milestone 1/2 use the onboard LEDs as indicators of loads.
February 5, 2021 Lab 2, Milestone 1/2 onboard LEDs are assigned by student number not TAs
February 6, 2021 Lab 2, Milestone 1/2 correction made regarding LED assignment.
February 8, 2021 Lab 2 Milestone 1 corrected to specify a 3 bit combinational code unlock.
March 15, 2021 Lab 6, Bonus milestone modified.
March 15, 2021 Lab 6, Reference to LCD in part list removed.
March 30, 2021 Lab 7, milestone 2 clarified.
March 30, 2021 Lab 7, Bonus milestone modified.