CMPE 127 LAB

INFORMATION

Prepared by

Dr. Haluk Ozemek

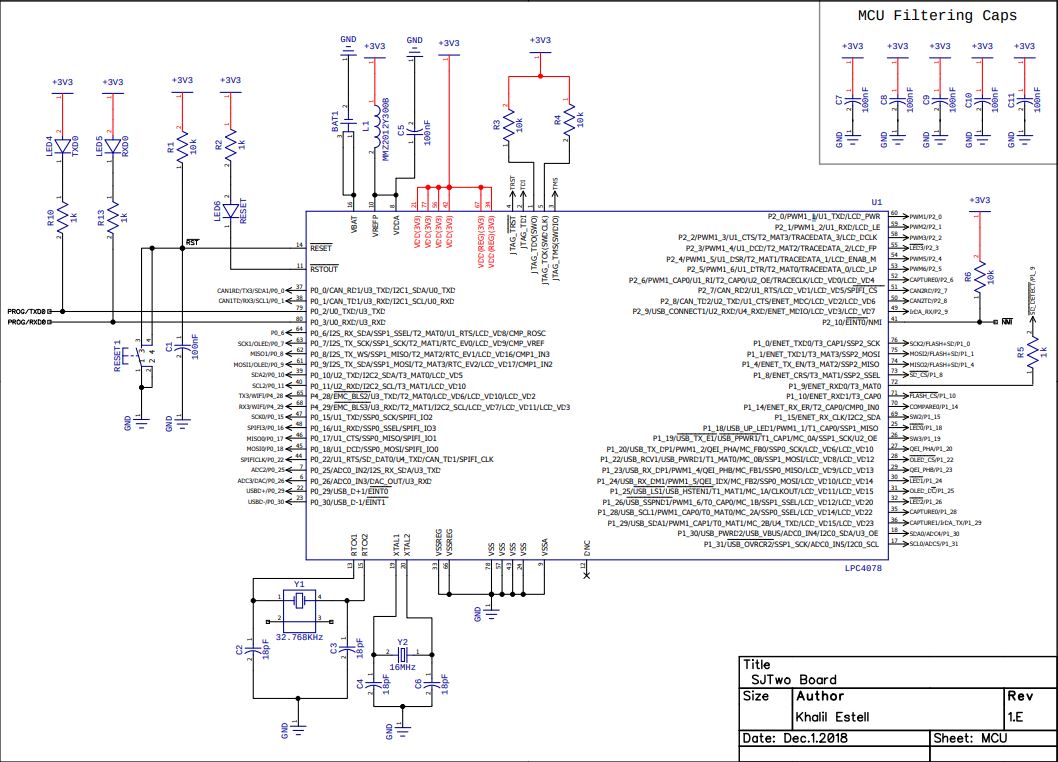
ISA Zackery Plovanic

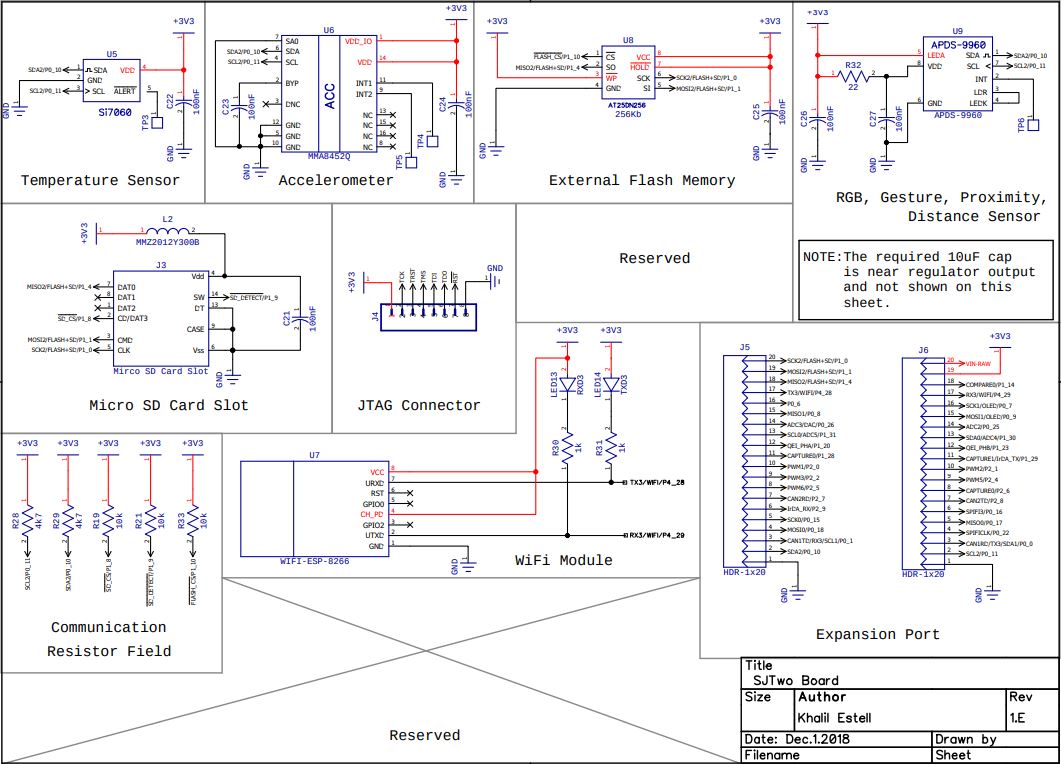
ISA Nikki Ravi

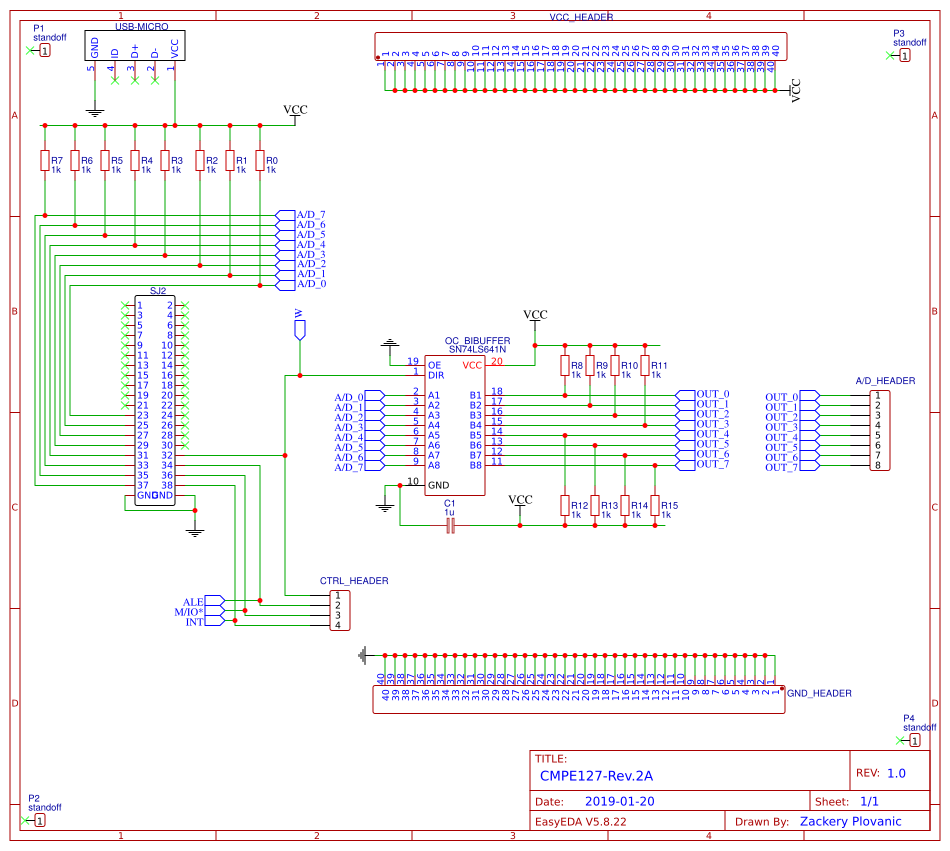
ISA Joseph Abero

ISA Grant Haack

Spring 2019

**SJSU-Dev2 Shematics:** ****

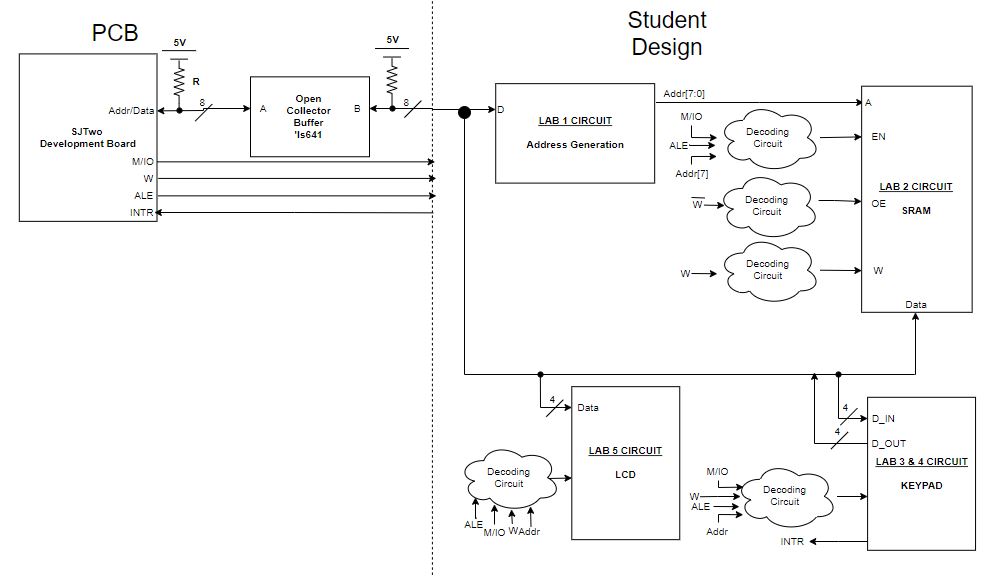
****

**PCB Layout:**

Information about the SJ-Two board can be found here:<https://github.com/kammce/SJSU-Dev2>

Schematic: <https://github.com/kammce/SJSU-Dev2/blob/master/schematic/Schematic-RevE.2.pdf>

An example of how to use the board’s GPIO pins: <https://github.com/kammce/SJSU-Dev2/blob/master/firmware/examples/Gpio/source/main.cpp>

****

**PCB Setup:**

This section is dedicated to a recommended approach to bring up your PCBs. This lays out an efficient and testable approach to ensure that initial errors do not propagate down the road, making it harder discover the mistakes.

1. Solder the ribbon connector onto the PCBs. Once this is done, you can write a small program and test this connection by manipulating your GPIO pins to high/low voltage, and measure this on the ribbon connector. If you are measuring the correct voltages at the ribbon connector you are good to move on. If not, you may have to add more solder to the connector to ensure a proper connection is present. For example, you can write a small program to set Gpio(2,2) to a high voltage. Measure the corresponding ribbon connector pin using a multimeter/logic analyzer and make sure you see a high voltage as well. Try again with a low voltage on the Gpio pin. You should do this with all of the pins that will be used in the this lab, which can be found in the Lab 1 description. In this step, you can also solder on male header pins on the CTRL\_HEADER pins, and try a similar test making sure that the voltage levels there follow the voltage at the corresponding pins.

\*NOTE: For these tests, you should not have the AD bus GPIO pins configured as open drain, since the pull-up resistors are not present at the moment.

1. From there, you can solder your USB-Micro breakout boards to the top left of the PCB, along with the sixteen 1k resistors. Once soldered, you can plug in a USB-Micro cable and grab a multimeter to ensure that the resistors are receiving around 5 Volts on the side where the large PCB traces are.
2. Similar to step two, you can then solder on male header pins onto the VCC\_HEADER and GND\_HEADER on the PCB. A similar test can be done as before, where you ensure that you are measuring around 5 volts on the VCC\_HEADER, and 0 volts at the GND\_HEADER.
3. Lastly, male header pins can be soldered onto the A/D\_HEADER, as well as a 20-pin DIP socket can be soldered onto the OC\_Buffer label on the PCB. To test this portion, place the ‘LS641 IC into its socket, and set the Write control signal, GPIO pin (0, 17), to a high voltage. This selects the proper direction pin, so the values on the SJ-Two’s A/D Gpio pins will be propagated to the A/D\_HEADER. You can now do a similar test to step one, testing different voltage values and seeing if the A/D\_HEADER follows.

\*NOTE: For these tests, **IT IS NECESSARY** to have the AD bus GPIO pins configured as open drain. Failure to do so can damage your SJ-Two board.

**Parts List:**

The parts list for this lab can be found here: <https://docs.google.com/spreadsheets/d/1sy5ossfm_3sAyZz8HF2i3hiS39N2mdsG_aOGJ11-F5g/edit?usp=sharing>. Parts should be bought as soon as possible to ensure you have the proper tools to start on the labs ASAP..

**Tentative Lab Schedule:**

|  |  |
| --- | --- |
| Lab 1: Shared Bus Architecture | Feb 12th |
| Lab 2: SRAM Memory | Feb 26th |
| Lab 3: Polling Keypad | Mar 12th |
| Lab 4: Interrupt Driven Keypad | Mar 26th |
| Lab 5: Parallel LCD | Apr 9th |
| Lab 6: System Integration | Apr 23rd |

**Lab Experiment Reports**: (to be updated as needed)

1st page:

Name:

Lab: CMPE 127 lab

Experiment Number and Title:

Abstract:

Pages to follow:

**Detailed Description of Functionality**: Describe what the circuit does.

**Details on the Design**: Include details on your design - truth tables, K-Maps, equations and logic diagrams. Provide timing diagrams of the interface you are designing.

**Schematics**: Additionally, package designations (also called reference designators) should be labeled using the "U#"-subscript method and pin numbers should be shown to reflect how you actually wired the final experimental circuit. No hand drawing is allowed. LogicWorks is a “good” schematics tool:

**Parts List**: List part name/number and package

**Description of Algorithms**: For each function that is performed on a device, provide detail description of algorithms used to performed accesses to the devices on the interface

**Step-by-Step Description of the Demo**: Describe how you would demo the functionality of your design.

**Source Code:** Paste the code used to demonstrate in a readable format (Code Blocks add-on for Google Docs, etc.)

**Lab 1**

**Shared Bus Architecture**

Goal: Lab 1 is to familiarize yourselves with the many control signals of a CPU. You will need to write software to utilize the ALE pin to store the CPU’s address portion of its shared bus architecture, as well as fill out the “instructions” (functions) that are typically used in assembly.

1. Install a working environment of the SJSU-Dev2’s Framework:

(<https://sjsu-dev2.readthedocs.io/en/latest/index.html>), (<https://github.com/kammce/SJSU-Dev2>)

1. Fill out the member functions of the class below:

|  |
| --- |
| #include <stdint.h>  #include <L1\_Drivers/gpio.hpp>  class Bus  {  public:  enum class ControlType  {  kMemory = 0,  kIO  };  void Init()  {  ad[0].GetPin().SetAsOpenDrain();  ad[1].GetPin().SetAsOpenDrain();  //...  ALE.SetAsOutput();  W.SetAsOutput();  M\_IO.SetAsOutput();  }  void Write(ControlType control, uint8\_t address, uint8\_t data)  {  }  uint8\_t Read(ControlType control, uint8\_t address)  {  }  private:  Gpio ad[8] = {  Gpio(2, 2),  Gpio(2, 5),  Gpio(2, 7),  Gpio(2, 9),  Gpio(0, 15),  Gpio(0, 18),  Gpio(0, 1),  Gpio(0, 10)  };  Gpio Write = Gpio(0, 17);  Gpio ALE = Gpio(0, 22);  Gpio M\_IO = Gpio(0, 0);  Gpio Int = Gpio(0, 11); //This will be used in Lab 4  }; |

\*Note: It is **ABSOLUTELY NECESSARY** that you set your A/D GPIO pins to open drain mode. If this is not set properly you have the potential to break your new SJ-Two boards. This

should be done in your class constructor, and can be done like so:

3. Physically create the circuit to store the CPU’s address since it is only temporarily available. This address, in addition to data presented on the AD bus, will be used to interact with peripherals in future labs.

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

\*Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

\*Note: The SJ2’s GPIO library can be found here:

<https://github.com/kammce/SJSU-Dev2/blob/master/firmware/library/L1_Drivers/gpio.hpp#L32>

Example Code: <https://github.com/kammce/SJSU-Dev2/blob/master/firmware/examples/Gpio/source/main.cpp>

**Lab 1 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | Tues. | TBD |
| 1 | 2/5 |  |
| 2 | 2/12 |  |

Rubric:

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Development Environment Setup complete | 1 |
| Completed function definitions for Bus class | 5 |
| Week 2 | Appropriate Assertions of Control Bus pins | 2 |
| Successful Implementation of Write Operation to the Address Generation Circuit. | 7 |
| Total | | 15 |

**Lab 2**

**Peripheral Interfacing: SRAM Memory**

Goal:

Lab 2 is to interface two SRAMs to our existing architecture. Using the addresses created in the previous lab along with the M/IO and W pins, you will interact with specific locations in the each of the SRAMs. Address decoding will be needed to ensure only one of each is driving, eliminating the possibility of bus contention.

1. Design the address decoding circuitry to access the SRAM components, along with your system’s memory map.
2. Draw the schematics for your design.
3. Implement the wiring of your design (exactly as shown in your schematics).
4. Using the “instructions” that you made in the previous lab, write software to **write** and **read** data to various locations in your SRAMs. The data you read may be displayed on the 7-segment display available on the processor board.
5. Allow user input for any address and data value between 0 and 255.
6. Demo your circuit.

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

**Lab 2 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | T | TBD |
| 1 | 2/19 |  |
| 2 | 2/26 |  |

Rubric

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Schematics displaying appropriate connections including M/IO pin. | 1 |
| Sram circuitry complete including memory decoding portion | 4 |
| Week 2 | Successful Write and Read operations to SRAM0. | 5 |
| Successful Write and Read operations to SRAM1. | 5 |
| Total | | 15 |

**Lab 3**

**Peripheral Interfacing: Polling Keypad**

Goal:

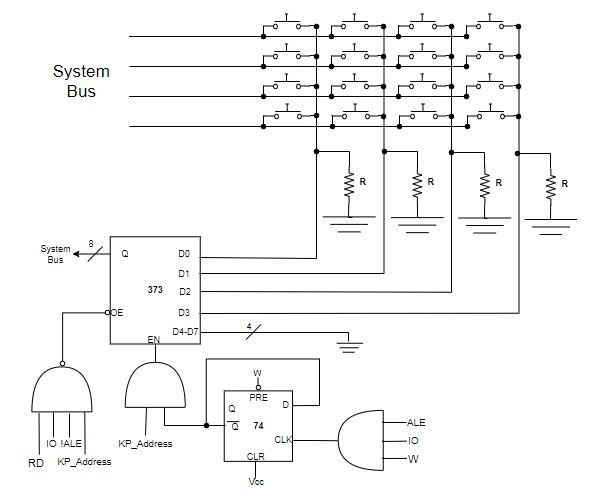
Lab 3 is to interface a keypad to the existing lab architecture.

1. Do not unwire the sram circuitry. Design the keypad interface in addition to the two sram packages on the interface.
2. Design the additional control circuitry and a state machine to interface the keypad. See the diagram on the next page.
3. Update the schematics for your board.
4. Program your design such that you can read the pressed key and display the key on the 7-segment display available on the processor board.
5. Determine the timing of the control signals and implement your state machine.
6. Implement the wiring of your design (exactly as shown in your schematics).
7. Demo your circuit.

\*\*The peripherals address must be fully decoded.\*\*\*

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

****

**Lab 3 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | T | TBD |
| 1 | 3/5 |  |
| 2 | 3/12 |  |

Rubric:

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Appropriate I/O selection of keypad. | 2 |
| Completed Circuit for Keypad | 7 |
| Week 2 | Keypad value displayed on the processor’s 7-segment display | 7 |
| Circuit displays polling functionality | 2 |
| Total | | 18 |

**Lab 4**

**Peripheral Interfacing: Interrupt Driven Keypad**

Goal: Using the Keypad circuit from lab 3, you will extend this design to be interrupt driven. Rather than constantly checking the columns to see if a key has been pressed, the software will utilize an interrupt to determine when a button has been pressed.

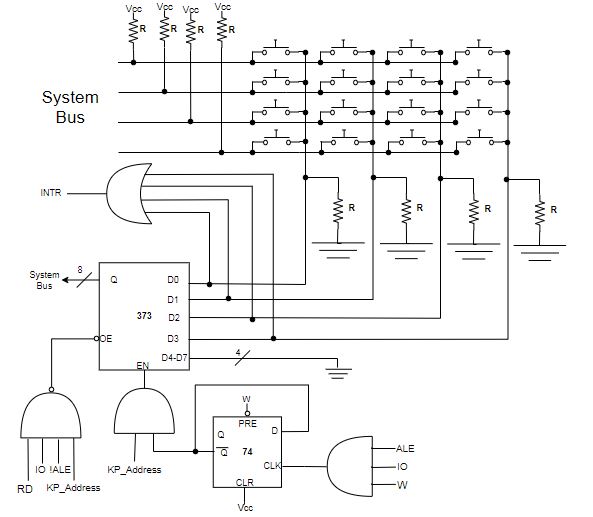
1. Manipulate your previous software:
   1. The row enabling and corresponding column checks of the keypad should be done inside of the ISR function.
   2. When first entering your ISR function, print a message to console showing that the ISR has in fact been called.
   3. While an interrupt is not triggered, i.e. outside of the ISR function, periodically print a message to console showing that a key has not been pressed.

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

Note: Example code for setting up an ISR:

<https://github.com/kammce/SJSU-Dev2/blob/master/firmware/examples/GpioInterrupts/source/main.cpp>

****

**Lab 4 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | T | TBD |
| 1 | 3/19 |  |
| 2 | 3/26 |  |

Rubric:

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Circuit completed for Interrupt | 1 |
| Code for ISR | 3 |
| Week 2 | Keypad value displayed on the processor’s 7-segment display | 2 |
| Interrupt functions properly and include print statement for every interrupt (aka can interrupt processor while it is executing other operations)   * After interrupt is processed, processor must resume previous operation (continuous print statement) | 6 |
| Total | | 12 |

**Lab 5**

**Programmable Peripherals: Parallel LCD**

Goal:

Lab 5 is to interface an LCD to our current architecture. Similar to the previous labs, this peripheral will need its own address. Since we are interfacing with a programmable IO device, a particular order of bytes/instructions must be sent to the LCD before it is able to properly operate.

1. Determine what modes to set your LCD in. This is done by sending your LCD specific commands/bytes, which may be different depending on the LCD you use. Look at your specific LCD’s datasheet.
2. Design the address decoding circuitry to access the LCD, along with your system’s memory map.
3. Draw the schematics for your design.
4. Implement the software and wiring of your design (exactly as shown in your schematics).
5. Demo your circuit.

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

**Lab 5 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | T | TBD |
| 1 | 4/9 |  |

Rubric

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Completed circuit for LCD | 1 |
| Appropriate I/O selection for LCD | 2 |
| Can display characters on two lines   * Code is flexible to allow any kind of words/sentences to be written | 9 |
| Total | | 12 |

**Lab 6**

**Lab Project: System Integration**

Goal:

In Lab 6, you will utilize your systems hardware and integrate them to create a cohesive project. You can design anything you like as long as it uses all the peripherals from the previous labs. You may even interface a new device like a temperature sensor, ROM cartridge, or servo motor. Please approve the project with your lab TA before starting.

An example project would be a simple calculator.

1. A user can press on the numbered keys to create the operands, while the lettered keys can select the specific operation.
2. The system will store the result of the operations in the system’s SRAMs. These results may be read afterwards to do further operations.
3. The operands, operations, and result of this calculator must be displayed on the LCD.

Demo your circuit.

Submit a printed copy of your lab report to instructor as scheduled. Include a copy of your program code.

Note: Late demos and late reports are not accepted (without documented proof of reasons beyond your control).

**Lab 6 Rubric**

Due Dates:

|  |  |  |
| --- | --- | --- |
| Week | T | TBD |
| 1 | 4/23 |  |

|  |  |  |
| --- | --- | --- |
| Requirement | | Points |
| Week 1 | Integrated SRAM(s), keypad, LCD together | 6 |
| Achieves intended functionality  (e.g. For a calculator, should include all operations: +, -, \*, /) | 8 |
| Displays ALL relevant information on LCD (e.g. For a calculator, displays) | 4 |
| Creativity | 2 |
| Total | | 20 |