ECE-412 INTRO TO EMBEDDED SYSTEMS

LAB 3

UART, ADC, EEPROM, GPIO, LCD Module

Eugene Rockey,

Yellow highlight points out lab related action required from the student.

Green highlight points out report related action required from the student.

Cyan highlight emphasizes certain terms and information.

Lab 3, in part, is an introduction to a variety of embedded peripherals including example supporting system software written in C and Assembly. All applicable Xplained Mini Mega328P data sheets/user manuals are recommended for this lab. The user manual and or data sheet matching the issued LCD module is recommended. This lab folder employs the ACM1602K LCD module.

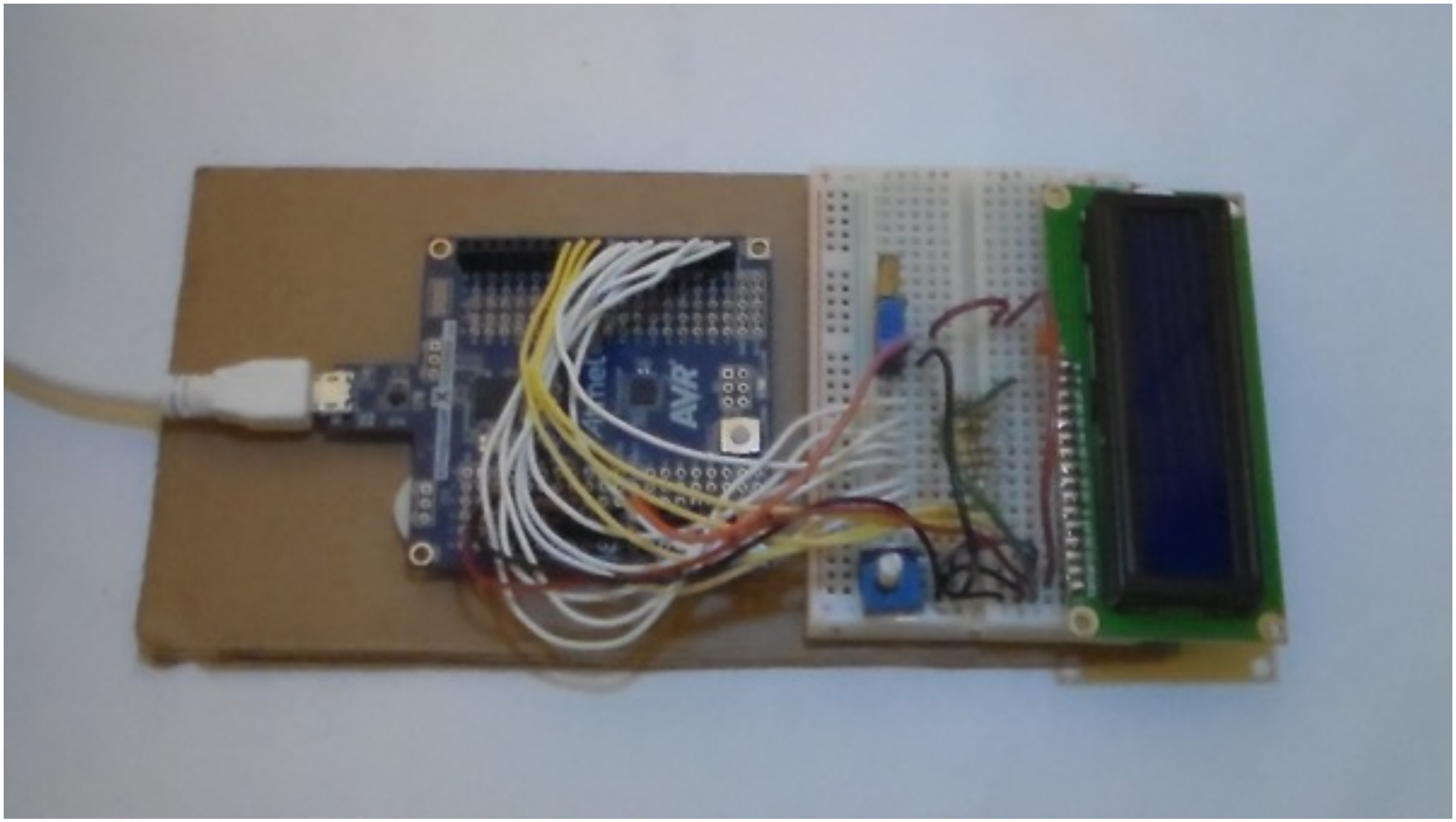
Communication inside an MCU chip is between the processor, memory, and peripherals using internal address, data, and control lines. External communication can involve a variety of peripherals and interfaces and also other MCU's or processors. For example, the Mega328P's internal Universal Synchronous Asynchronous Receiver Transmitter (USART) is an I/O Peripheral device that allows for serial I/O communication. Serial communication involves transmitting and or receiving data one bit at a time. Synchronous serial transmission means that the serial data is accompanied by a clock signal. Asynchronous communication means that the serial data is not clocked but rather framed using control bits. In lab 3, the asynchronous mode is investigated by interfacing the Mega328P board to a PC running a terminal program such as TeraTerm. In UART mode, and running Assembly-based routines, the Mega328P will communicate with TeraTerm via the mEDBG's virtual communication port and the board's USB cable.

The Mega328P Analog to Digital Converter (ADC) is a 10-bit successive approximation type. It has either 6 or 8 time-multiplexed input channels depending on the chip's part number. This peripheral is investigated to determine how to configure and utilize such a device for interfacing the Mega328P to external analog signals. Analog signals converted to digital data can then be readily processed by user code for some practical purpose.

The internal EEPROM can be written to during programming of the chip if an .eep file was generated during the assembly process as one of the output files. Subsequently, data can be read from the EEPROM during run time. Note the .eep file will be generated by Atmel Studio if the .eep option is enabled in the toolchain and if EEPROM is referenced in the code such as with the .eseg directive or by the Atmel Studio Framework (ASF). The EEPROM can be written to and read from during runtime by special user code via the EEPROM control, data, and address registers. In this lab, the EEPROM is investigated as a Non-Volatile secondary storage device during runtime.

The ACM1602K Liquid Crystal Display (LCD) module is designed to receive commands and alphanumeric data using its external data bus and control lines. The Mega328P board connects to the LCD module's data and control lines via its GPIO ports. And the GPIO ports can be configured to drive, under Mega328P control, the on-board SPLC780D1 LCD MCU, which in turn drives the liquid crystal display. The SPLC780D1 implements an industry standard set of LCD commands and alphanumeric data. Google and use the ACM1602K data sheet or the data sheet that matches your issued LCD module.

The preliminary steps necessary for successfully completing this project are soldering headers onto the Xplained Mini evaluation board. These headers allow the MCU to be connected to an external circuit. The external circuit is shown below and constructed on a breadboard. A schematic is provided in the lab 3 folder. It is advised to securely Velcro the Xplained Mini board and breadboard with circuit onto a piece of cardboard for support OR in some such manner. Have the instructor/TA check the circuit before connecting to power.


AVR

Part 1: In this part, COPY and PASTE and BUILD the given TinyOS project in Atmel Studio. This project consists of an expandable operating system designed to run and exploit the features of the Xplained Mini board and its Mega328P MCU. Two files are given, one is the main.c and the other is the Assembly1.s file. Start a new GCC C executable project in Atmel Studio and name it TinyOS. Now, add an Assembly file to the project. The Assembly file type should be .s for use with the GCC C based solution and can be added to the solution by right-clicking in the Solution Explorer window where main.c exists. The Assembly1.s file will be created under Solution Items. Click on and drag Assembly1.s down from Solution Items and drop it on top of main.c where it should now show up. Go back to Solution Items and remove the Assembly1.s file from there, by right clicking on it and then clicking on the REMOVE menu item.

Open the new main.c in Atmel Studio editor window, copy and paste the TinyOS code from the given main.c file in the lab 3 folder to the new main.c, overwriting any existing template code.

Open the new Assembly1.s in Atmel Studio editor window then copy and paste the TinyOS code from the given Assembly1.s file in the lab 3 folder to the new Assembly1.s, overwriting any existing template code.

Build the TinyOS then program the Mega328P MCU.

Open a serial communication terminal such as TeraTerm. A CDC COM port to the mEDBG should be detected and used. Set the serial port settings to 9600 baud, 8 data bits, no parity, 1 stop bit, and no hardware flow control. Study the Mega328P data sheet section 24, covering the USART.

Click the Start without Debugging. The terminal should display the TinyOS banner and command line. Click in the Terminal window. Explore the commands: LCD, ADC, and EEPROM. Refer to main.c and Assembly1.s listings when testing the commands. Atmel Studio is not necessary to run the TinyOS.

In your report, discuss the TinyOS Assembly instructions that initialize the USART via its registers. Discuss all the USART registers. Discuss all the TinyOS USART related Assembly and C code that make the command line possible. Discuss how the USART could be configured to implement a different baud rate, different # of data bits, different parity, different # of stop bits, and what is hardware flow control? Discuss the interrupt driven Tx and Rx capabilities of the USART according to section 24.

Part 2: In this part, examine and test the 'ADC to voltage' lite demo. The TinyOS command line should display a relatively accurate voltage reading. The ADC Channel 0 or PC0 is connected to the wiper of the precision multi-turn potentiometer for the analog input signal. Repeat the (A)DC menu selection as the potentiometer is turned and observe the voltage reading change. A DMM may be used to verify the accuracy of the voltage reading. In your report, discuss how the ADC is gathering analog input and converting it to a voltage reading, include details about the ADC registers, successive approximation circuitry, and the related TinyOS C and Assembly codes. Discuss the free running mode versus the single conversion mode. Discuss the differential input versus the integral input mode. Discuss the clock pre-scaler. Discuss the 10-bit resolution.

Part 3: In this part, examine and test the 'EEPROM' lite demo. The TinyOS should display the character 'F'. The two assembly driver functions EEPROM_Write() and EEPROM_Read() simply write an 'F' to a memory location in EEPROM and then read it back and echo it to the command line. All this happens during runtime which is uniquely different than generating an .eep file and having the programmer write the data to EEPROM before runtime.

After running the (E)EPROM routine once, unplug the Xplained Mini board. Go to the EEPROM C code in Atmel Studio and comment out the EEPROM_Write() line. BUILD the TinyOS. Plug the board back into the computer. Program the Mega328P chip with the modified TinyOS. Bring up the serial terminal program and the TinyOS command line. Run the (E)EPROM command and notice the 'F' is still in EEPROM memory.

In your report, discuss the EEPROM data, address, and control registers. Discuss the code for writing to and reading from the EEPROM during runtime. Refer to section 12.4 of the AT Mega328P data sheet.

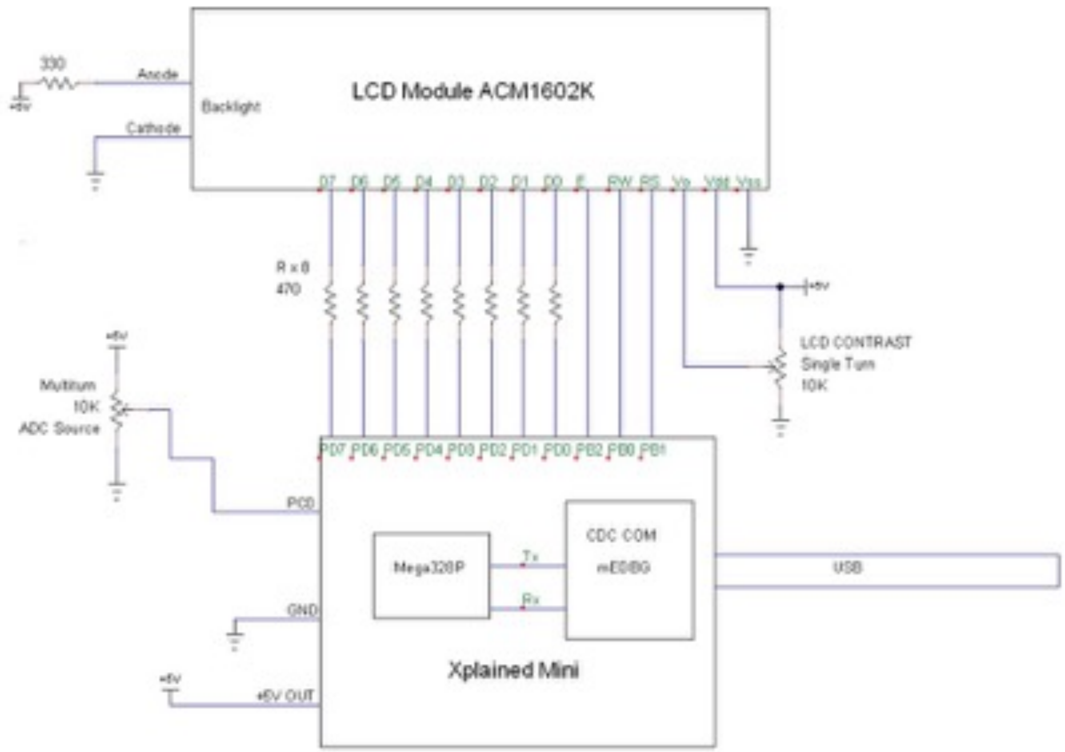
Part 4: In this part, examine and test the 'LCD' lite demo. The LCD module can display alphanumeric graphics and text using a set of commands and data provided by the manufacturer. The Mega328P GPIO can be configured to send commands as well as send and receive data to and from the LCD's MCU. Note that the Virtual Com Port used to communicate with the Mega328P and its TinyOS uses PD0(Rx) and PD1(Tx). The LCD module uses all of Port D in its 8-bit parallel communication with the Mega328P MCU. This could be a conflict but fortunately the GPIO ports can be reconfigured during runtime. And the UASRT and LCD can be enabled and disabled during runtime. Basically, the LCD operates while the USART is disabled and the USART operates while the LCD is disabled. It is possible for multiple devices to use the same port in this way.

In your report, discuss the LCD module commands and alphanumeric data and refer to the manufacturer's data sheet. Discuss how the LCD is wired to the Xplained Mini board. Discuss the back light and contrast control. Discuss the GPIO ports and how they can be configured as input or outputs, as interrupt inputs, and as pulled high. Discuss the multiple purposes of the GPIO pins/ports such as TWI, SPI, TC, UART, and AC. Refer to section 18 of the Mega328P data sheet.

In your report, list both main.c and Assembly1.s under the SOFTWARE heading. Create and enter all the necessary comments where it states "Student Comments Here". Try to make the comments highly descriptive such as the purpose of each line with respect to the program and not the instruction's defined operation.

The Lab 3 schematic, shown below, lists: Xplained Mini Board, micro USB cable, ACM1602K LCD Module, single turn 10K potentiometer, multi-turn 10K potentiometer, one 330 Ohm resistor, and eight 470 Ohm resistors.

IMPORTANT: Use red wire for +5 volt, black wire for ground (GND), white wire for PD0-PD7, yellow wire for PB0-PB2, orange wire for PC0, and blue wire for LCD contrast. A wire color scheme is essential for identifying signals and power in a circuit.



Title ECE412 Lab 3 Circuit		
Author Eugene Rockey		
File C:\Users\Donkey\Desktop\test.dsn		Document
Revision 1.0	Date 2/16/2018	Sheets 1 of 1