PROJECT 1: VOLTAGE DATA LOGGER

OVERVIEW

For this project you will program your Freedom board to create a programmable analog voltage data logger. You are encouraged to start with the serial port demonstration code from Chapter 8 of the ESF textbook (found on the book's github repository), incorporating other example code from the textbook as needed.

REQUIREMENTS

Your system must accept commands from UART0 (from a terminal program on a PC) to accept configuration information, acquire the requested data, and send it back to the PC as shown below.

OPENSDA DEBUGGER

You will need to use the PE Micro debugger (rather than ARM's CMSIS-DAP debugger) in order to have the virtual serial port communication on the USB interface. Follow the instructions at https://github.com/alexander-g-dean/ESF/tree/master/Tools.

SYSTEM STATE OVERVIEW

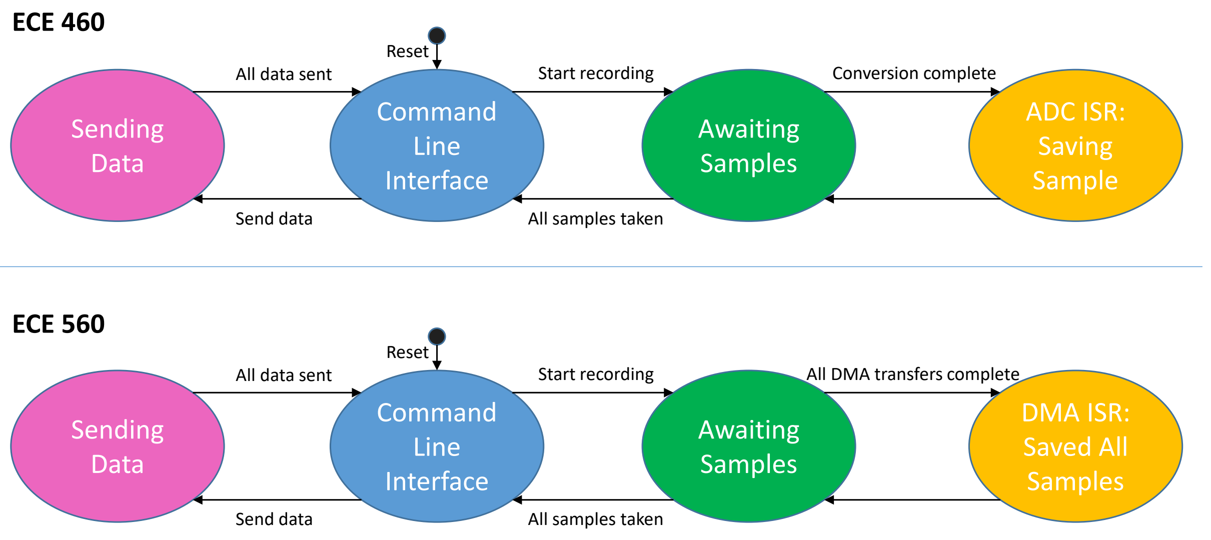

Figure 1. States for data acquisition system.

State	LED Color	Debug Signal
CLI	Blue	PTB0
Awaiting Samples	Green	PTB1
Saving Sample	Yellow (Red & Green)	PTB2
Sending Data	Magenta (Red & Blue)	PTB3

The system can be in one of several states, as shown above and described below. Your code must indicate the current state by lighting the LED and raising the corresponding debug signal to 1 (clearing others to 0).

- CLI: Providing a command line interface with the UART.

- Awaiting Samples: Waiting for more samples.
- **ECE 492 only:** ADC ISR Saving Sample: ADC0_IRQHandler copies ADC result to memory (g_samples).
- **ECE 592 only:** DMA ISR Saved All Samples: DMA0_IRQHandler indicates all sampling has been completed.
- Sending Data: Sending formatted data out the serial port.

Note: ECE 592 students will use the DMA controller to eliminate the need for the ADC IRQ Handler.

Use your logic analyzer to view the debug signals and observe system behavior. **We will use these debug signals when grading your submission.**

GLOBAL DATA

Declare these as global variables, outside of any function.

- Use a 6000-element array of 16-bit unsigned integers (uint16_t) called **g_Samples** to hold the samples.
- Use an unsigned integer (uint16_t) called **g_NumSamplesRemaining** to hold the number of samples remaining to be taken.

SYSTEM STATE DETAILS

COMMAND LINE INTERFACE (CLI) STATE

When in this state, the program must accept and process the following commands as described in this table and related text.

Class	Command	Example	Parameter	Response	Comments	Next State
Both	R: Start recording	R	(none)	“Starting Recording.”	Clear number of samples taken, switch to sampling state	Sampling
Both	S: Send data	S	(none)	“Sending data (3 samples).”	Switch to sending data state	Sending Data
Both	C: Select input channel	C3	Channel number	“Channel x selected.”	Valid channels are 0, 3, 6, 11, 14, 23, 26 and 27.	CLI
592 only	P: Set sampling period	P100	Sampling period, measured in µs	“Sampling period set to 100 us.”	Valid values are integers from 2 to 1,000,000	CLI
592 only	N: Set sample count	N100	Number of samples	“Sample count set to 100.”	Valid values are integers from 1 to 6,000	CLI

- All commands are terminated by the enter key.
- Your code must reply to all valid commands with the responses shown above.
- Commands with illegal parameters are to be ignored. All other inputs are to be ignored.
- 492 students do not need to implement the P or N commands. Instead use a fixed 10 µs sample period and 1000 sample count.

On the transition from CLI to Awaiting Samples, be sure to start the ADC (and other peripherals as needed).

AWAITING SAMPLES

In this state, the program waits for the next sample to be taken and saved. Timing error must be less than 1% to receive full credit.

When **g_NumSamplesRemaining** reaches 0, go back to the CLI state.

ECE 492 ONLY: SAVING SAMPLE

In this state, the program samples the selected input channel and saves it to memory. The program is in this state when the ADC ISR is executing. The ADC ISR must read the conversion result, saving it in the array, and decrement **g_NumSamplesRemaining**.

If there are no samples remaining, disable the timer TPM0 to prevent further ADC conversions.

ECE 592 ONLY: SAVED ALL SAMPLES

The program is in this state when the DMA ISR is executing. This state indicates that all samples have been copied by the DMA controller to memory. The DMA ISR must update **g_NumSamplesRemaining** and disable timer TPM0 to prevent further ADC conversions.

SENDING DATA

Send a header indicating the number of samples. Calculate the voltage of each sample taken (floating-point variable) and then send the sample number and that voltage as text out the serial port. You can assume the ADC reference voltage is 3.3 V. For example:

```
Sending data (3 samples).
1: 3.013 V
2: 3.020 V
3: 2.960 V
```

DATA ACQUISITION SEQUENCE DIAGRAMS

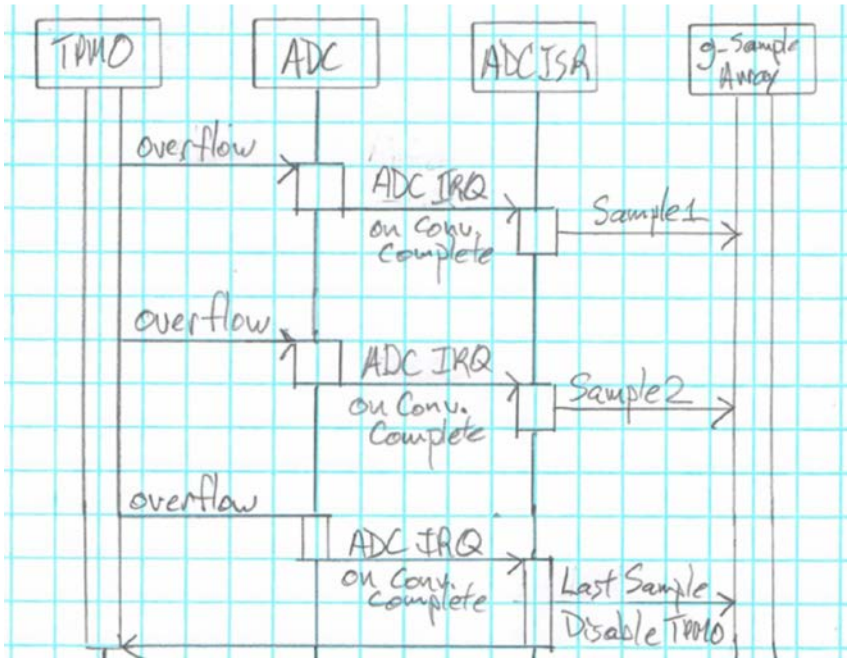

Figure 2. Sequence Diagram for ECE 460/492

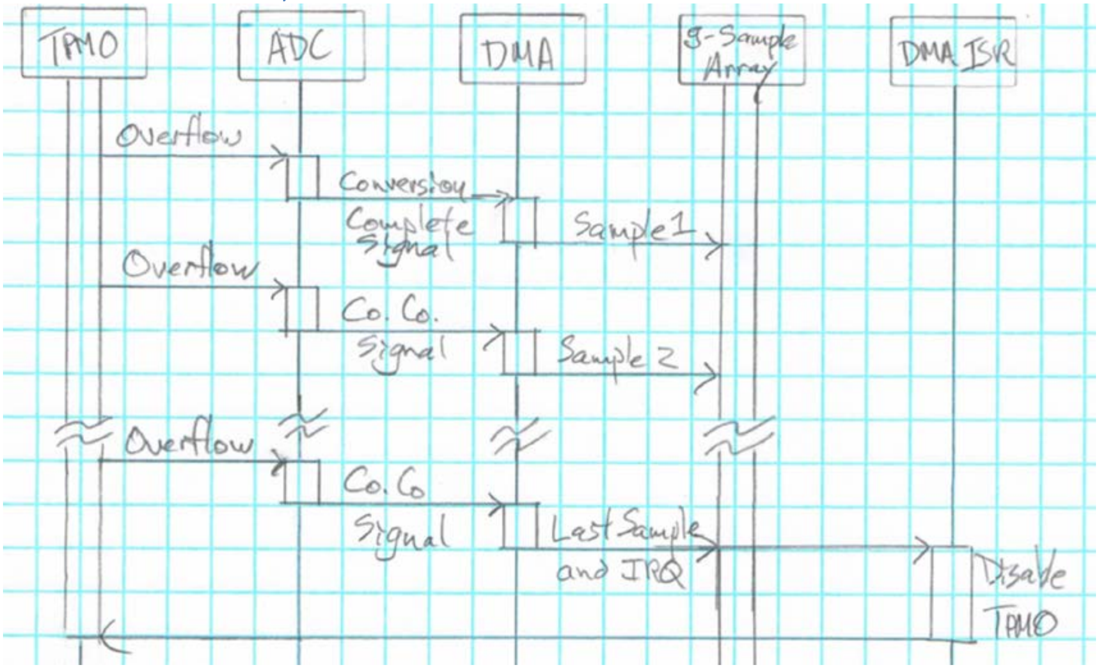

Figure 3. Sequence Diagram for ECE 560/592

PERIPHERALS USED

GPIO

Use GPIO peripherals to create digital output signals which indicate the current system state by:

- Lighting the RGB LEDs.
- Asserting a debug signal on Port B.

Refer to ESF Chapter 2 for details on using GPIO.

UART

Use UART 0 to communicate with a terminal program on the PC through the virtual serial port provided by the Segger OpenSDA debug application. The UART settings are at 115,200 baud, 8 data bits, no parity. Use polling (not interrupts) for the UART.

Refer to ESF Chapter 8 (pp. 210-220, 227-233) for details on using the UART.

TIMER

Use TPM0 to trigger the sampling of the ADC with a hardware signal. Set the period of the timer overflow to match the requested sampling period. Do not use the timer's interrupt or IRQ handler.

Refer to ESF Chapter 7 (pp. 194-199) for details on using this timer.

ANALOG TO DIGITAL CONVERTER

Use the single-ended, 12-bit conversion mode without averaging. This supports a conversion rate of up to 818,330 samples per second (about 1.22 µs per sample). Use hardware triggering of the ADC from Timer TPM0.

ECE 492: Use the ADC's interrupt to trigger the execution of the ADC ISR (ADC0_IRQHandler), which will copy the conversion result into the next free location of **g_Samples**.

ECE 592: Use the ADC's interrupt to trigger the DMA controller to copy the result into the next free location of **g_Samples**.

Refer to ESF Chapter 6 (pp. 164-180) for details on using the ADC. See Table 6.6 on p.171 to find a given ADC channel on the MCU board's connectors and pins.

ECE 592 ONLY: DMA CONTROLLER

Use the DMA controller to copy the requested number of samples from the ADC result register to **g_Samples**. When configuring the DMA controller, be sure to do the following:

- Use ADC0 as the trigger source.
- Set the number of bytes to transfer based on **g_Samples** (two bytes per sample)
- Use cycle stealing, so only one transfer is performed per request
- Set the source address to the ADC0 result register.
- Set the destination address to **g_Samples**, and allow destination incementing.
- Use 16-bit source and destination sizes.

When all the samples have been transferred, the DMA ISR should run and do the following:

- Clear the DMA controller's Done flag.
- Disable TPM0.
- Disable the DMA controller.
- Clear the variable **g_NumSamplesRemaining** to zero.

Refer to ESF Chapter 9 for details on using the DMA controller.

TESTING

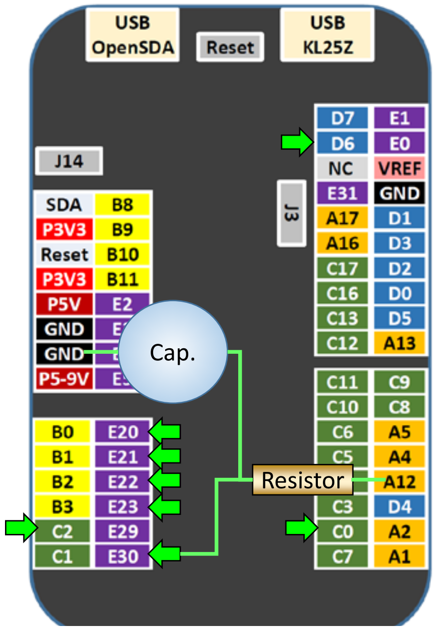

Figure 4. Test circuit connections and ADC inputs to test.

Test your system using a low-pass filtered square-wave generated by one of the timers on the KL25Z. Connect an RC circuit as shown between pin 8 of J1 (marked PTA12) and ground. TPM1 Channel 0 will generate a square wave at a fixed frequency of 500 Hz (high for 1 ms, low for 1 ms). Choose a resistor and capacitor such that the RC time constant is roughly 1 ms. For example, use a 1 μF capacitor and a 1 kΩ resistor, and τ = RC = 1 μF * 1 kΩ. A larger RC time constant will filter more of the square wave's high frequencies.

Use the code in the file tester.c on Github (Project 1) to configure and start the timer, and call it from your main function once. The orange arrows in the diagram indicate the ADC inputs which you will need to use to test the system.

FURTHER INFORMATION

- Embedded Systems Fundamentals, Chapters 6, 7 and 8.
- FRDM-KL25Z Reference Manual.

DELIVERABLES

- Completed Project: Zipped archive of project directory submitted via Moodle.

Please reduce the archive size by cleaning the project targets in MDK-ARM before zipping the directory (Project -> Clean Targets, or Shift-F7).

No project report is required.