PROJECT 3:
SHARING THE ADC

OVERVIEW

For this project you will share the ADC between the buck converter controller and the touchscreen code, while still maintaining correct timing for the buck converter controller.

CODE MODIFICATIONS AND EXPERIMENTS

STARTER CODE

Start with the code on the github repository at Projects/Project_3_Base. It includes code from Lab 2 for controlling buck converter and flashes the HBLED about every two seconds. However, as soon as you touch the screen, the function LCD_TS_Read (called by Thread_Read_TS) uses the ADC to read the touchscreen. This changes the ADC configuration and stops the control loop and the flashing. Instead, you want a touch to do one of two things (in Thread_Read_TS):

- If touched in upper portion of screen (above the Dim ←→ Bright text), draw a white line between previous and current touch points.
- If touched in lower portion of screen (on or below the Dim ←→ Bright text), set the peak current (**g_peak_set_current**) to between 1 and 120 mA, based on the X position of the touch.

WHY NOT USE A MUTEX FOR THE ADC?

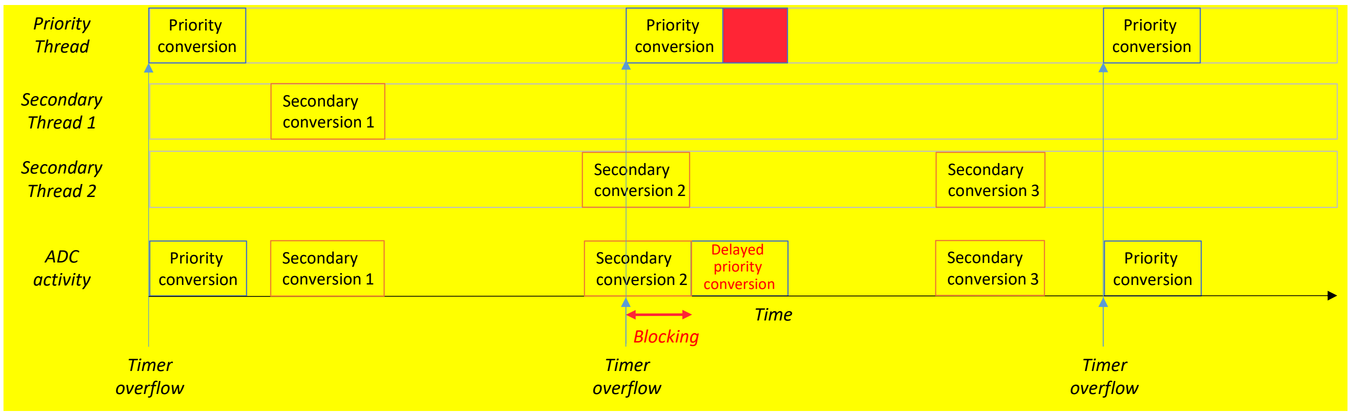


Figure 1. Using a mutex leads to priority inversion which can delay higher priority conversion.

Using a mutex to protect the ADC would introduce excessive timing interference for the HBLED controller because it allows blocking of the highest priority thread, shown in red in Figure 1. This blocking reduces the controller's effectiveness by delaying its sample.

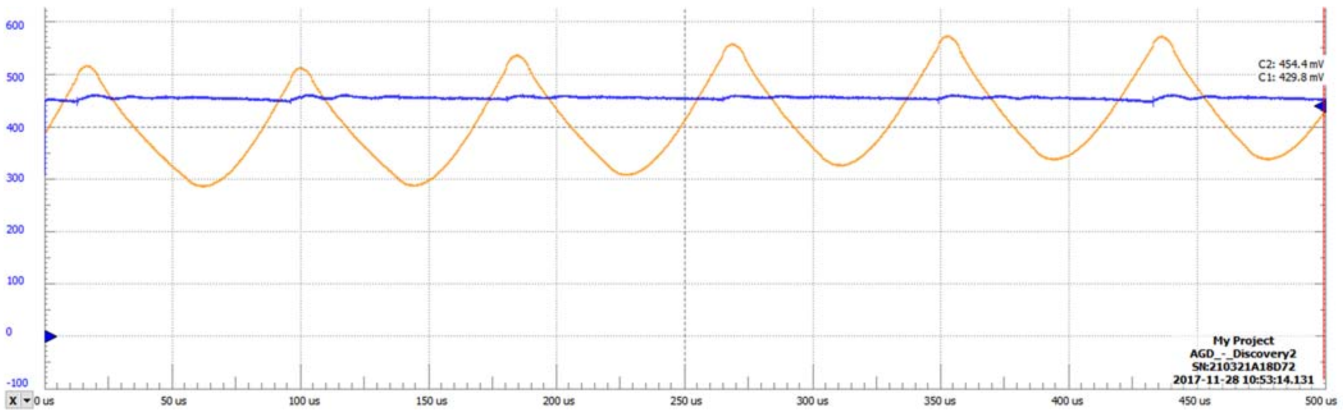


Figure 2. High-frequency switching ripple on V_{sense} makes timing of ADC sample for control loop very important.

For example, Figure 2 shows an example of the sense voltage for a buck converter with an 83 µs switching period. The ripple is 250 mV peak-to-peak. In the worst case, the controller's ADC sample is delayed by half of the switching period, and the voltage error will be 250 mV, leading to a 113 mA error in the current reading.

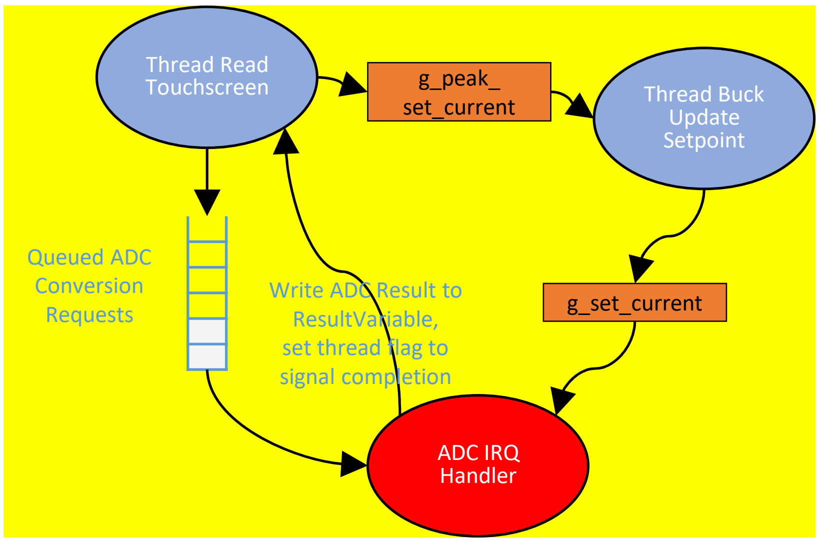
In order to evaluate the impact of smaller timing errors, we can approximate the waveform as a triangle wave, which means that the signal changes at a rate of 250 mV/(83 µs/2) = 6.02 mV/µs. This implies the current changes at a rate of 2.74 mA/µs. So if the ADC sample is delayed by 8 µs, the current reading will have an error of 2.74 * 8 = 21.9 mA.

BUILDING AN ADC SERVER

You will write new code and modify existing code to share the ADC between the HBLED controller and the touchscreen. You will create an ADC server to control conversions, which will prioritize conversions for the HBLED controller over other conversions.

SOFTWARE ARCHITECTURE

The following diagram shows the software architecture for key parts of the system. Note that there are other threads and ISR present as well (e.g. Thread_Read_Accelerometer, Thread_Update_Screen).


Thread Read Touchscreen
g_peak_ set_current
Thread Buck Update Setpoint
Queued ADC Conversion Requests
Write ADC Result to ResultVariable, set thread flag to signal completion
g_set_current
ADC IRQ Handler

Thread_Read_TS calls LCD_TS_Read. LCD_TS_Read uses a digital input to determine **if** the screen is pressed. If it is pressed, it uses the ADC twice to determine **where** the screen is pressed (once for the X axis, once for the Y axis). To complete this project you'll need to modify the LCD_TS_Read to enqueue channel conversion requests and block on the results. LCD_TS_Read returns a 1 indicating the press, and writes the coordinates into the structure pointed to by the function's argument (called position). If the screen is not pressed, LCD_TS_Read does not use the ADC and returns a value of 0.

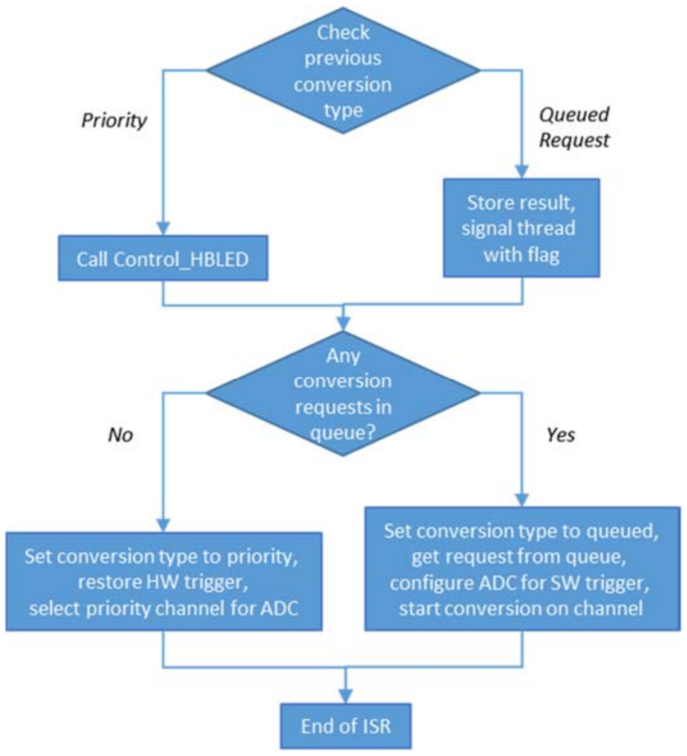
Thread_Read_TS checks to see which part of the screen is pressed. If the upper part is pressed, then it draws a line between the previous and current touch locations. If the lower part is pressed, it updates the global variable **g_peak_set_current** based on the X coordinate. Thread_Buck_Update_Setpoint manages the timing of the flash, and uses **g_peak_set_current** to set the current levels for the flash.

ADC USE AND ADC ISR DESIGN

There are two types of ADC conversion. Conversions for the buck converter are time-critical, so they are triggered by the overflow signal of timer TPM0. Conversions for the queued requests are triggered by software in the ADC ISR. You'll need to add a static state variable to identify the type of conversion. The ADC ISR must use the following structure and behavior, which are summarized in the flowchart:

- What was the type of conversion?
  - If the conversion was a **priority** conversion (for the buck converter), the ISR calls Control_HBLED, which will read and use the result. Note that you will need to modify Control_HBLED as described below.
  - If the conversion was for a **queued request**, the ADC ISR places the result in the location specified in the message (pointed to by ResultPtr) and notifies the client thread by setting FlagNum of thread TID.
- Is there is a conversion request in the queue?
  - If there is, the ISR gets that message, configures the ADC to read that channel with software triggering, updates the state variable (indicating it is servicing a queued ADC conversion request), and starts the conversion with software triggering.
  - If not, the ISR ensures the ADC is configured for hardware triggering from the TPM to read the buck converter voltage.

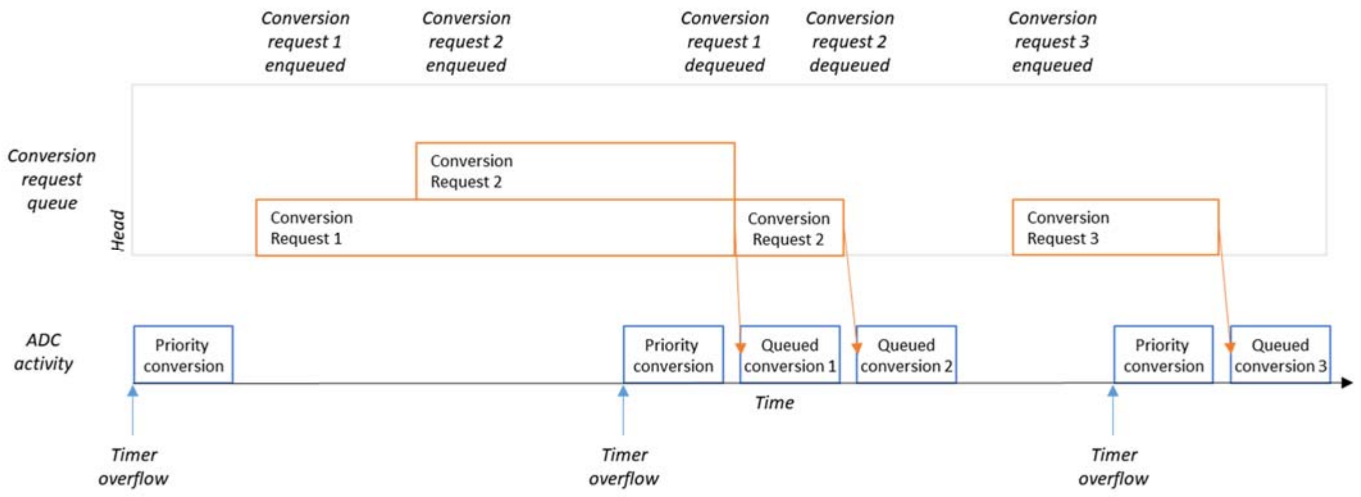
Assume that there will never be more than 4 pending conversion requests, and that all requested conversions will be completed before the next priority hardware trigger.


Check previous conversion type
Priority
Queued Request
Call Control_HBLED
Store result, signal thread with flag
Any conversion requests in queue?
No
Yes
Set conversion type to priority, restore HW trigger, select priority channel for ADC
Set conversion type to queued, get request from queue, configure ADC for SW trigger, start conversion on channel
End of ISR

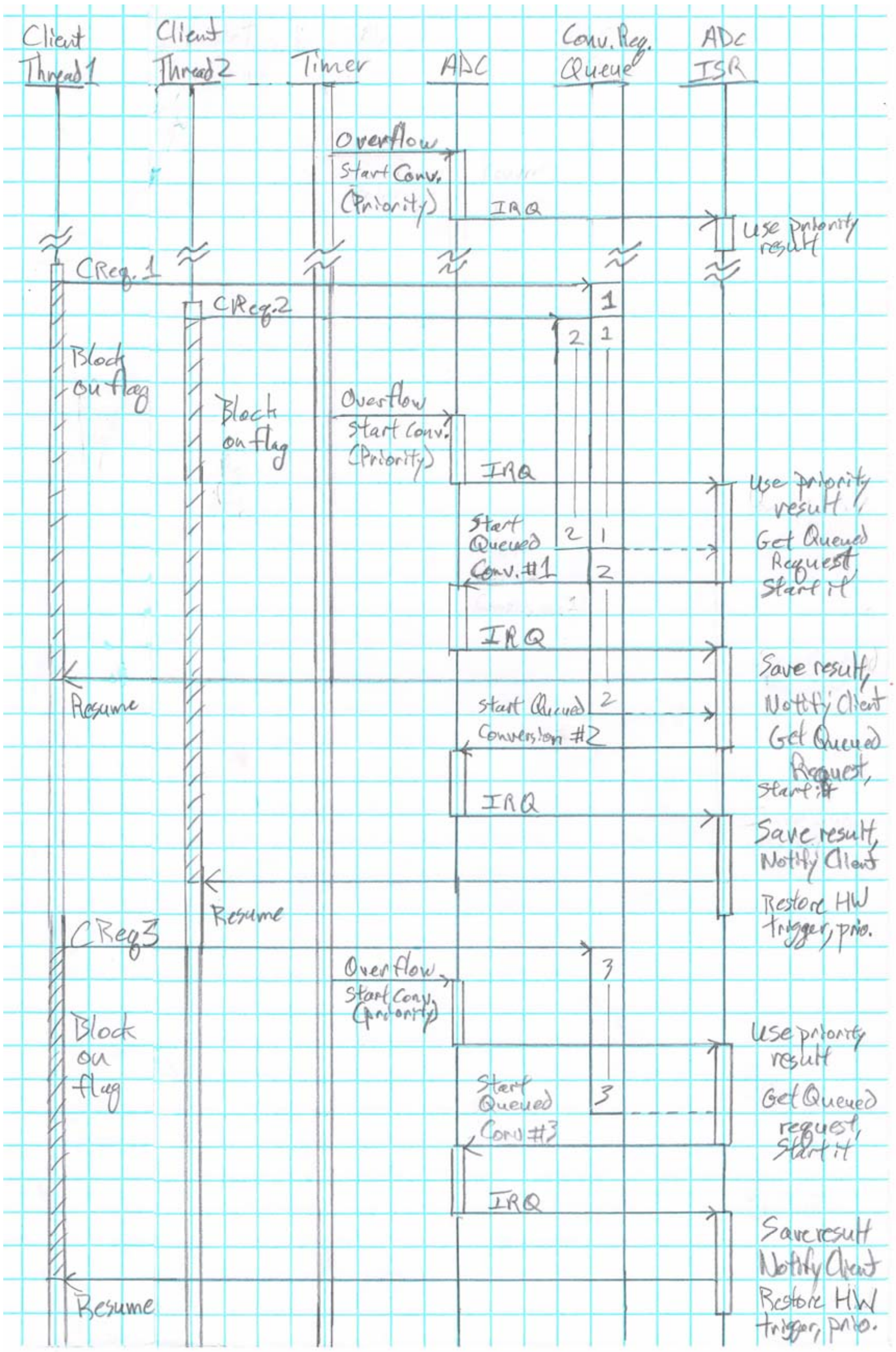
CONVERSION EXAMPLE

The diagram below shows an example of several conversions:

- The priority channel is converted
- Two conversion requests are enqueued
- The priority channel is converted and the two queued requests are converted
- A third conversion request is enqueued
- The priority channel is converted and the third enqueued request is converted



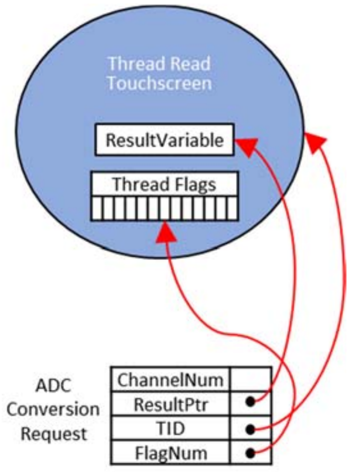
The sequence diagram below shows the transactions in detail.


Client Thread 1
Client Thread 2
Timer
ADC
Conv. Req. Queue
ADC ISR
Overflow
Start Conv. (Priority)
IRQ
Use priority result
CReq. 1
CReq. 2
1
2
Block on flag
Block on flag
Overflow
Start Conv. (Priority)
IRQ
Use priority result
Start Queued Conv. #1
Get Queued Request, Start it
IRQ
Save result, Notify Client
Resume
Start Queued Conversion #2
Get Queued Request, Start it
IRQ
Save result, Notify Client
Resume
Restore HW trigger, prio.
CReq 3
3
Overflow
Start Conv. (Priority)
Block on flag
Use priority result
Start Queued Conv #3
Get Queued request, Start it
IRQ
Save result Notify Client
Resume
Restore HW trigger, prio.

COMMUNICATION BETWEEN THREADS AND ISRS

A client thread (e.g. Thread_Read_TS) must send a message to the ADC ISR through a queue (e.g. ADCRQ). You must declare a data structure type so that each message holds the following information:

- ChannelNum: Number of channel to convert
- ResultPtr: Pointer to location (variable in client thread) to store conversion result
- TID: identifier of client thread
- FlagNum: Number of thread flag to set, which indicates conversion is complete



The diagram shows how this message information is used to indicate where to put the conversion result and how to signal the thread that the conversion is complete. Note that only the field ResultPtr is a pointer.

Refer to the lecture notes and demo code for using queues and thread flags. There is detailed CMSIS-RTOS2 documentation online:

- Overview: https://www.keil.com/pack/doc/CMSIS/RTOS2/html/index.html
- Thread flags (includes code example): https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__ThreadFlagsMgmt.html#details
- Message queues (includes code example): https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__Message.html#details

OTHER CODE CHANGES

Here are some of the necessary changes to make to the existing code. There are likely to be other required changes.

You will need to change the code in Control_HBLED as follows:

- Delete code (if any) to start a conversion or wait for its completion. Since Control_HBLED is called by the ADC IRQ Handler, the conversion has already been completed.
- ~~If you use a queue to change the setpoint (described later), read all messages from that queue and update the setpoint to the value specified in the last message.~~

You will need to change other code as follows:

- LCD_TS_Read() in touchscreen.c needs to send conversion requests to the ADC server through its message queue, and then block on the thread flag indicating conversion completion.
- Thread_Read_TS() in threads.c needs to update **g_peak_set_current** based on the X position of the touch.

ECE 560 ONLY: GUI ENHANCEMENT

Modify the code in **Thread_Update_Screen** to display the numerical value of **g_peak_set_current** on the LCD. Be sure to use the **LCD_mutex** to access the LCD safely.

REPORT

Submit a report containing the following:

- Mixed-signal screenshot(s) showing correct system operation (e.g. similar to the example on page 2 but without queue information). Include any debug signals needed to show that the system is working as specified.
- Text for each screenshot explaining what it shows and why it means your system is working.

OTHER

GENERAL

1. You must work individually on this project.
2. Your code may be examined for possible plagiarism by using MOSS (http://theory.stanford.edu/~aiken/moss/).

DELIVERABLES

Submit the following items online.

- Zipped archive of project.
- Project report.

GRADING

The following factors will be used to determine your grade.

- Correct, complete functionality.
- Code quality. The instructors may deduct points for remarkably bad coding practices (for example functions longer than a page, magic numbers, non-descriptive names). These are described online at http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html. If you have a question about what is acceptable, please post it on the class discussion forum so others can learn as well.