ECE 271 – Microcomputer Architecture and Applications Lecture 21

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

7 April 2022

Announcements

- Read Chapter 20
- Midterm Tuesday the 12th
- Course website (including gitlab) will be down for maintenance Friday - Monday
- Office Hours will be cancelled April 18th and 20th due to faculty interview presentations. If you need to meet that week let me know and I can arrange alternate times
- The Student Symposium is April 15th, consider going

Midterm Review

- Shorter than last time.
  No assembly language
  There will be a take-home section (floating point)
- C programming
  - Be sure you know how to set/clear bits in a register
- Interrupts
  - Vectored interrupts – there is a table low in memory (the interrupt vector) which has a lookup table of pointers. When an interrupt happens, the CPU looks

up the address from the appropriate pointer and jumps to it

- Interrupt handlers on Cortex-M are just normal functions (the CPU does “stacking”, saving some registers on the stack automatically for you. A magic value is put in the Link register so the CPU knows you are returning from a handler).
- Once an interrupt happens, you enter the handler. Often you will need to ACK (acknowledge) the interrupt so the CPU knows you are done handling it. How this happens can vary with what hardware you

use (it's often clearing a bit, but somehow it happens automatically).

- Enabling an interrupt is multiple steps. You have to enable it in the device (For example, in the timer registers). Then you have to tell the NVIC (interrupt controller) to enable it. Then you have to enable it on the CPU globally with `__asm("CPIE i")`. Finally you have to enable whatever hardware will be triggering the interrupt.
- Some devices (such as TIM4) can have multiple causes that trigger the same interrupt (i.e., overflow and

capture). You can figure out which one was the cause by checking certain bits to see the source of the interrupt.

- Timers
  - Lots of registers to set.
  - Set what clock source to use system wide.
  - You can then use the prescalar to divide the frequency down.
  - You can set to count up (from 0 to the ARR register value), count down (from ARR to 0) or center count (count up and down).

- PWM
  - pulse width modulation: you set a CCR value that when the counter gets bigger it triggers an output to go high. This way you can generate a regular pulse with an overall frequency based on ARR and a duty cycle based on CCR.
- Input Capture
  - You can use the Timer to measure the length of signals.
  - When the incoming signal has a transition (high to low or low to high) the timer will grab the current timer count and store it in a register. (optionally also an

  interrupt can be triggered).
  - By saving the previous value, and subtracting from the current value, you can calculate how many clock ticks have happened
  - If your signal is wider than the maximum value of the counter, an overflow can happen. In order to measure signals that long you can catch the overflows and count them, and then add that time in to the measured time.
- Fixed/Floating Point: there will be a take-home question where you will convert a decimal value to IEEE-574 floating point and back.

- Be aware of purpose of floating point
- Be aware of the layout of floating point (sign, exponent, fraction)

Analog/Digital Converters (ADC)

- Take an analog signal, quantize it to a set of digital values
- Compare against a reference voltage (result integer is a fraction of the total reference voltage)

ADC Key Terms

- Sampling rate (how many conversions per second). Can be millions or more.
- Number of bits in ADC (resolution)
  - Varies between 6 to 24 bits usually. 12 or 24 bits common.
  - This is how many steps (voltages) can be represented
- Power dissipation – how much power used for conversion.

Have you ever used an ADC?

- Any time you are trying to get analog data into a computer
- Microphone
- Temperature sensor
- Resistive touch screens
- Battery Level

Many, Many types of converters (textbook)

- Sigma-delta (low-speed)
  - low-sampling rate but high resolution (100 k-samples 12-24 bits) which is fine for things like audio
- Successive-approximation (SAR) (low-power)
  - Low-power and moderate sampling rate, 5 million samples/s
- Pipelined (high-speed)
  - oscilloscopes, HDTV, radar, needing grater than 5MSPS

More types of converters (Wikipedia)

- Direct conversion
  - Parallel: Just an array of 2^N comparators. Fast, but need a lot (256 for 8 bit)
  - Counter: count up and stop when gets to it
  - Servo-tracking: if too high, count down, if too low, count up, until it matches
- Integrating
- Apple II
  - RC circuit controls 555 timer. Count how long it takes

for it to trigger.

SAR ADCs (as used on our boards)

- Uses binary search
- Internal DAC (opposite of ADC) set to 1/2 Vref, compares it with Vin
- If Vin is larger, sets MSB (high bit), otherwise 0
- Next it tries either 3/4Vref or 1/4 Vref, does comparison
- Repeats, for each N bits
- Tradeoff between resolution and time

Sampling and Hold Amplifier (SHA)

- Can our signal change while we're trying to measure it?
- Uses a capacitor to grab the voltage and hold it while the ADC happens
- Capacitor takes a while to charge, need to wait
  $V_C(t) = V_{in} \times (1 - e^{-fractT_C})$
- Wait until the sampling time

ADC Sampling Error

- Difference from ideal result

STM32L Hardware implementation

- STM32 board uses SAR with SHA
- Always uses HSI clock no matter the board is using (can use 1, 2, or 4 divider in SMP register)
- Time to convert is sample time + channel conversion time.

STM32L4 Hardware implementation

- Three ADC modules: ADC1, ADC2, ADC3
- ADC1 and ADC2 can run in dual mode (both run at same time)
- Can be 12, 10, 8, or 6 bits
- 16 bits via “over-sampling” (measuring many times then averaging?)
- Provide Vref- and Vref+ externally, internal reference 3.0V
- Clock rate is independent of processor clock

- With 80MHz clock at 12 bits can get 5.33 million samples/second
- Can have a watchdog AWD which watches and if voltage goes above or below certain value, trigger interrupt (why?)

ADC conversion modes

- One input channel
  - Start conversion
  - ADC_DR holds result
  - EOC end of conversion flag set
  - Optionally generate interrupt
  - regular vs injected?
  - can be put into continuous mode
- Multiple input channels
  - Round robin switch between channels

Data Alignment

- Can be aligned in various ways
  - Right aligned, with zeros on left
  - Left aligned, with zeros to right. Note: 6-bit results are aligned within the byte
- Sign extended if injected channel?

ADC Input Channels

- Various GPIO pins can be hooked in
- Single-ended or differential
  - Single ended compares signal against ground
  - Differential takes two input pins and compares the difference

Triggering

- Can trigger with software `ADSTART`
- External trigger
  - Timer outputs
  - External pins
- Should set delay so doesn't retrigger so quickly

Other ADC topics

- Conversion sequence (switching between channels automatically)
- DMA
- Internal reference voltages
- Injected channels (you can have a setup where it regularly samples each channel in turn, but an injected one happens on demand and takes precedence.

Lab #10 – Setup ADC

- 12-bit ADC, single ended
- $ADV\,Result = \frac{V_{input}}{V_{REF}} * 4096$
- $V_{input} = \frac{ADC\,Result}{4096} * V_{REF}$
- Will use pin PA1 connected to `ADC12_IN6`

Lab #10 – Measurement

- First will test by measuring the voltage from a potentiometer voltage divider
- Second we will measure the result from an Infrared LED transmitter/receiver pair
- The voltage on the photo-transistor (IR receiver) will be proportional to the incoming IR light. Roughly proportional to the distance of an object reflecting the transmitter.