ECE 271 – Microcomputer Architecture and Applications Lecture 17

Vince Weaver

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

vincent.weaver@maine.edu

24 March 2022

Announcements

- Read Chapter 15
- Went over Midterm ABI question (5B) again

System Timer

- What are times useful for?
- Exact timing
- If you have an OS, a regular timer tick to keep track of time, also context-switch
- Things that run in background. i.e. heartbeat LED.
- How can you have that run in background? Hook up to timer-tick interrupt

ARM System Timer

- Note, this is part of the ARM processor so not documented in the STM32L4 SoC documentation (rather in the "Cortex-M4 Generic Users Guide")
- 24-bit down counter
- Counts down from N-1 to 0
- Interrupt when it hits zero
- Then reloads value from LOAD register
- Four registers

SysTick_CTRL (Control/Status Register)

- Bit 0 – ENABLE – enable
- Bit 1 – TICKINT – enable interrupts when hit zero
- Bit 2 – CLKSOURCE – 0 = external clock (AHB/8) or 1 (Processor clock, set in RCC_CFGR)
- Bit 16 – COUNTFLAG = special event happened

SysTick_LOAD (Reload Value)

- Bits 23-0 = RELOAD value.
- After counter counts down to zero, reloads SysTick_LOAD
- To interrupt every N cycles, set to N-1
- 24-bit so up to roughly 16M

SysTick_VAL (Current Value)

- Reading gets current value
- Writing any value to it resets to 0 (Setting to LOAD on next tick)

SysTick_CALIB (Calibration)

- Has the value needed to load to get a certain frequency.
- STM doc says this is 0x4000270F which gives 1ms when running the clock at 80MHz/8. This is different than what the textbook says.

Setting up the SystTick Timer

- Full details in text book
- Disable timer and interrupt in SysTick_CTRL
- Set LOAD value
- Clear current count by writing 0 to SysTick_VAL
- Set priority in `SCB->SHP`, we provide a NVIC_SetPriority() function that's smart enough to set SCB for system (negative) interrupt numbers
- Re enable ENABLE and TICK in SysTick_CTRL

Counting Period Example

- $SysTickPeriod = (1 + SysTick\_Load \times \frac{1}{SysTickClockFreq})$
- So if LOAD is 6 and clock is 1MHz, 7us

Lab #8 Notes

- Pulse-Width Modulation (PWM)
- Stream of 0s and 1, but the average voltage is in between which is useful sometimes
- Dimm an LED by pulsing it rapidly
- Control a servo motor by sending it proper stream of pulses
- Can play audio via 1-bit signal

Servo Motor

- A motor that allows exact absolute positioning (i.e. you can tell it to move to 90 degrees)
- Stepper motors you can turn an exact *relative* position (so many steps) but you don't know where you started
- Works via feedback. Various ways to do this (light, tachometer, etc)
- Our motors have a potentiometer connected via gear
- Our motors limited to plus/minus 90 degrees, can't turn whole way around

Controlling the Servo Motor

- Our motors want a very specific pattern
- 20ms (50Hz) repeating pattern
- -90 degrees = 1ms high, 19ms low
- 0 degrees = 1.5ms high, 18.5ms low
- 90 degrees = 2ms high, 18ms low
- This is not inherent in all servo motors, but is popular in ones like this that were originally used for remote control planes?
- You might actually find 1ms/1.5ms/2ms while 1.5ms is

exact, the other two might vary a bit and you might have to trial and error a bit to get it the full 90/-90 swing.

Lab#8

- First part is using PWM to dimm an LED
- Second part is to control a servo motor

Setting up PWM

- Could we do PWM in software?
  Yes, but couldn't do anything else.
  Also trouble if interrupts or something delays us.
  Better to do in hardware.

Advanced Control-Timers

- Note these are *not* the same as the SysTick timer
- Chapter 30 (p1007) in STM manual
- Chapter 15 in textbook
- TIM1/TIM8
- 16-bit auto-reload timers
- 16-bit prescalar
- Unrelated to the other TIM timers
- Each timer has 6 channels
- Key registers

- Counter register `TIM1_CNT`
- Prescaler register `TIM1_PSC`
- Auto-reload register `TIM1_ARR`
- Repetition count register `TIM1_RCR`

- Upcounting mode
  - Counts from 0 to ARR then restarts at 0
  - The repetition counter says how many times to count before triggering an interrupt?
- Downcounting mode
  - Counts starting at ARR down to 0, reloads ARR
- Center-count mode

- Counts up then down then up then down

Various Modes

- Can put counter in various modes. We'll use some of them next lab.
- Like 100 pages on this.
- We are setting PWM mode.
- Period set by ARR register, duty cycle by CCMR register
- Can do complicated things like assymmetric mode (with phase shift) or have different channels with different phases.
- Complicated way of setting phase

- Can generate both signal and inverse, and can add "dead time" to the transition
- Can xor together to generate more complex waveforms.

Where does the output go?

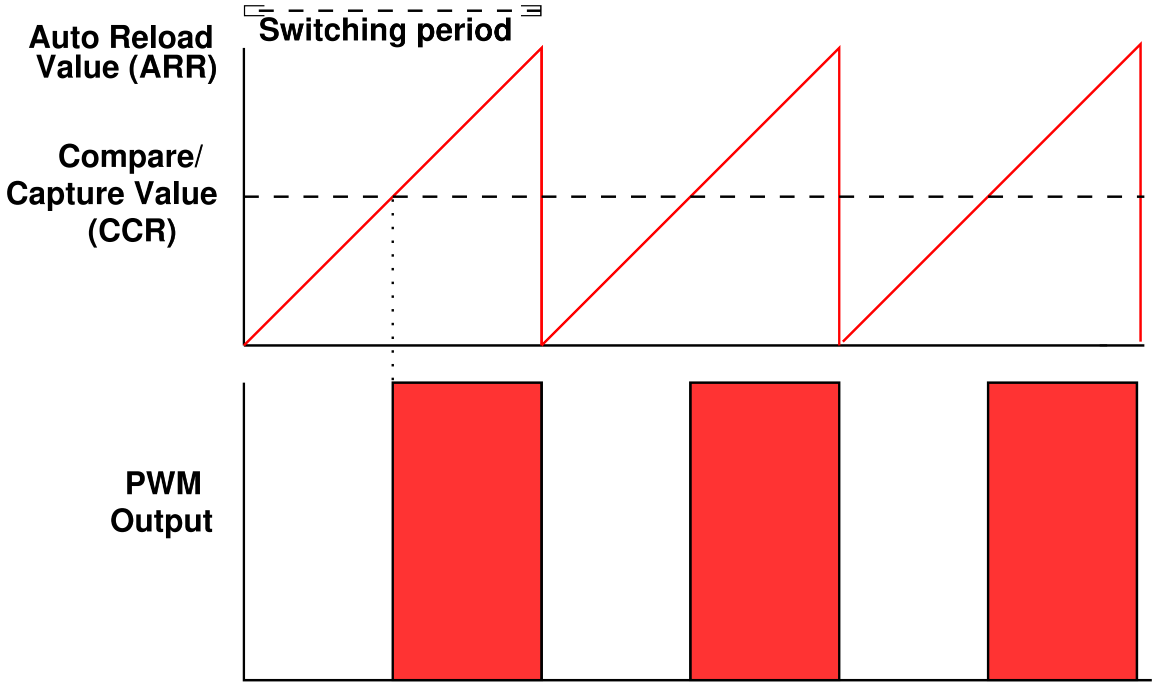
- This is configurable too.
- Can put to some but not all of the GPIO pins. Conveniently PE8 (green LED) gets the TIM1 output, but CH1N (so the inverted output)
- Have to set the AF (alternate function) register to pick. Somehow wasn't able to find where these are listed but it is in Appendix I of the textbook.

PWM Settings

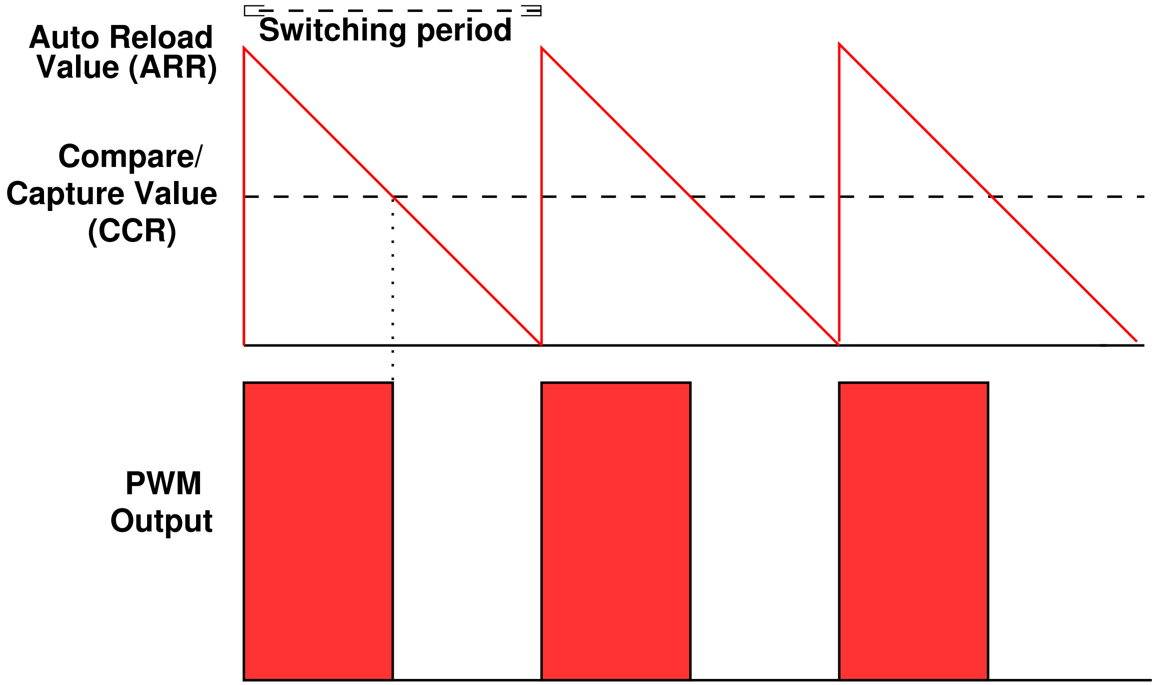
- There is a sawtooth carrier signal (draw graph) and a threshold when it is crossed (CCR) and the value it counts down from (ARR)
- $duty\_cycle = \frac{pulseontime(T_{on})}{pulseswitchingperiod(T_s)} \times 100\%$
  $= \frac{T_{on}}{T_{on}+T_{off}} \times 100\%$
- Three factors matter:
  - comparison between timer counter (CNT) and the reference value (CCR)
  - the PWM output mode

◦ the polarity bit

Up-Counting


Switching period
Auto Reload
Value (ARR)
Compare/
Capture Value
(CCR)
PWM
Output

Down-Counting



Center-Counting