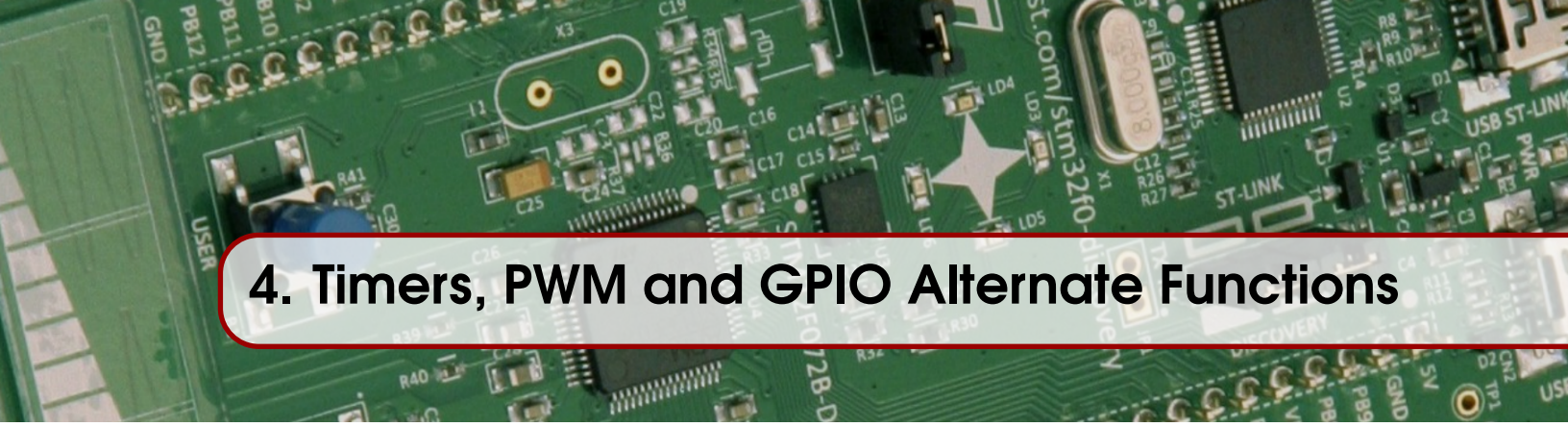

4. Timers, PWM and GPIO Alternate Functions

4.1 Introduction to Timers

Why have timer circuits? SysTick timer last lab modified its interrupt simple down-counter, generates ISR when hit zero used to create fixed-time periodic interrupts Hard and inefficient to measure time with only instruction count Have hardware to count processor or other clock source in the chip Means you can measure time to schedule operations

4.1.1 Systick vs Peripheral Timers

The SysTick is very limited, mainly used for just a timebase (Delay libraries, RTOS scheduler etc...) The peripheral timers of the STM32F0 have more advanced capabilities than generating simple interrupts when they reload their value.
(Block diagram of timer 2 & 3)
Here are a few basic features: Selectable and prescalable clock sources The timers have built-in frequency dividers (prescaler) to reduce the input clock signal so they can count at more arbitrary rates. Generate interrupts at multiple conditions Can generate interrupts on top, bottom or even arbitrary counter values. These are selected by the different modes of operation that the timer can be placed into. Ability to directly modify some GPIO pin outputs without requiring an interrupt. Used mainly for waveform generation. (capture-compare, PWM) Can directly read some GPIO input pins to measure the time between changes. (input capture)

4.1.2 Timers in the STM32F072

(Show table of timers in the STM32F072)
Discuss basic features: (BRIEFLY!) Timer Type Counter resolution/size Counter Type Prescaler DMA Request Capture/compare channels Complimentary outputs

4.2 Using the Timer Documentation

The timer peripherals in the STM32F0 are complex, it isn't possible to document them in the lab manual MUST get familiar with searching the peripheral reference manual For these sections will be using the documentation on timers 2&3 They're not super simple timers, but not the advanced one either Their documentation is 65 pages long... (and dense)

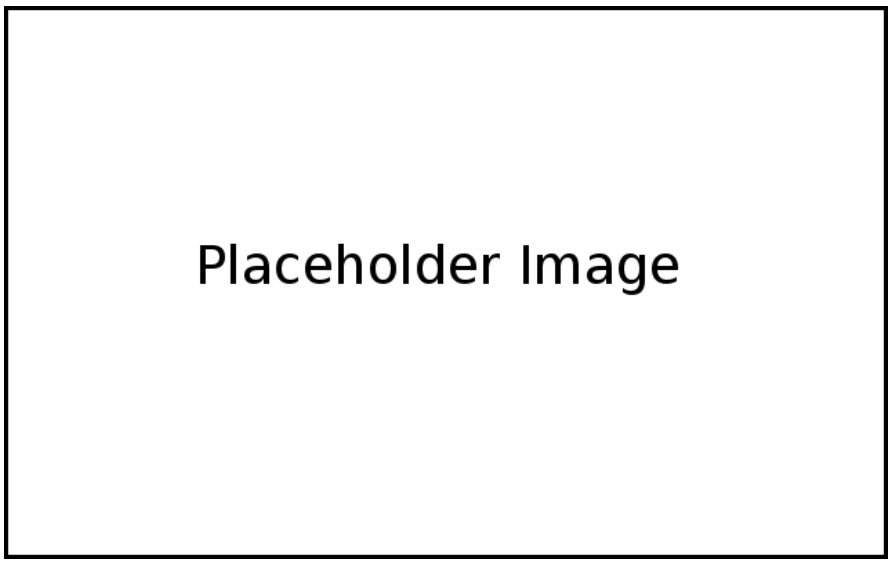

Figure 4.1: Block diagram of timers 2 & 3

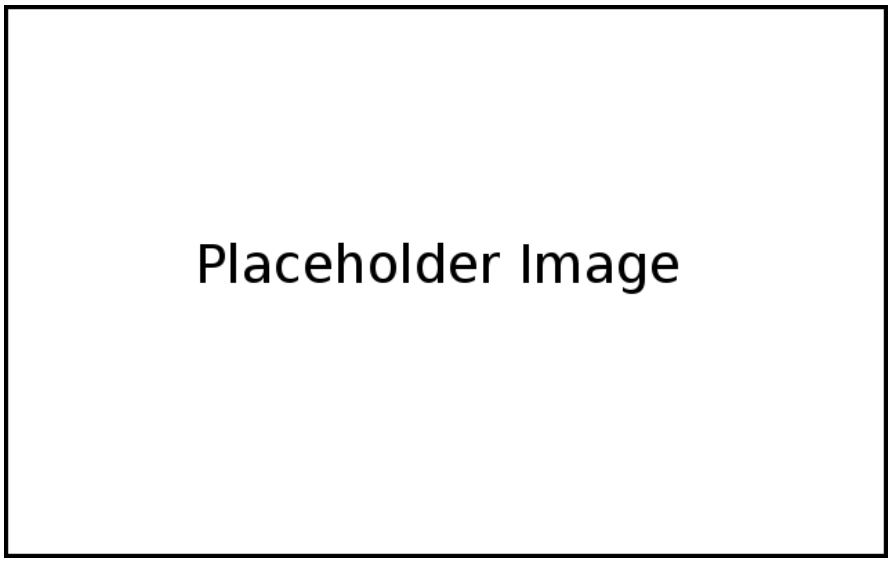

Figure 4.2: STM32F072RB hardware timers

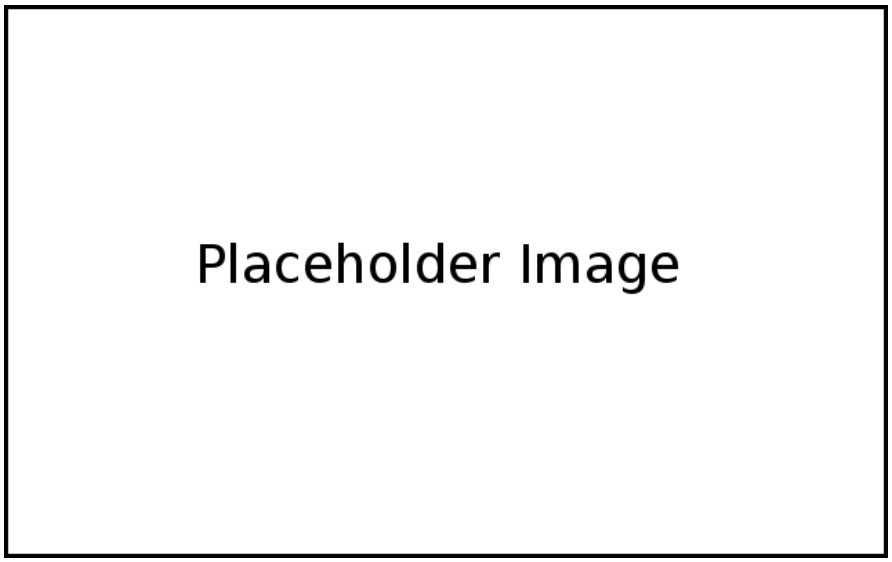

Figure 4.3: Timer counting modes and update interrupts

Reference manual documentation goes in this format: Basic information on what peripheral does Details and architectural block diagrams Documentation of the different modes These usually have some specific background info Summarized information on how to configure the mode Gives generic steps, will need to carefully read the register documentation Summarized information on data produced describe where produced data is written Includes figures depicting operation
This manual will document the purpose of the base registers as they are used in each mode May also indicate groups of control bits that pertain to the configuration or operation

4.3 Using Timers to Generate Interrupts and Events

These timer modes make use of the basic time-base unit of the timer. This portion does what you'd expect from the timer (it counts)

4.3.1 Basic Timer Operation

Basic registers used in these modes (not including control/setup registers) timer counter prescaler auto-reload
up, down, up/down (center-aligned) counting modes
(combination figure showing sawtooth/triangle waves for counter value)

Basic Timer Interrupts

UEV - Update event, occurs when the counter value is reset by hardware this can happen at counter overflow/underflow whenever the timer is reset to 0 because it reached the ARR value whenever the timer changes direction (center aligned mode) Other interrupts are generated by the capture/compare system

4.3.2 Selecting Timer Frequency and Interrupt Period

choosing a frequency the timer runs at (keep BRIEF) selecting a clock source internal clock, selected by the clocking system when configuring the project (most used) external clock pin for timer works on a configurable edge like the EXTI external interrupts Internal trigger inputs These signals can connect multiple peripherals together Used to chain timers together, or count events generated by other types of peripherals

Using the Prescaler

remember that it's 0-biased so subtract 1 from the desired divider (prescaler of 1 divides frequency by half)

Calculating Register Values

calculating the timer ARR and prescaler values from a desired interrupt period

■ **Example 4.1 — Calculating ARR and PSC Values.** Example: Do a calculation for an arbitrary period, perhaps multiple ways to do the same. ■

4.4 Using Timers to Generate or Measure Signals

These modes all use the capture/compare circuity of the timer Brief overview of the capture/compare unit
The mode of operation of the capture/compare unit is set by the 3 capture/compare mode registers. (CCMR1-3)

4.4.1 Input Capture Mode

Keep brief, not focus of lab Basic theory and graphical example Simple use-cases: stopwatch, tracking motor speed using a click-wheel will be using a fancy version of input capture mode (quadrature encoder mode) to track speed and direction of the motor in later labs. Counter value saved in the capture/compare register (CRR)

4.4.2 Output Compare Mode

Output compare mode uses an extra control register CCR The timer operates like normally with the counter and ARR but a capture/compare interrupt/event occurs whenever the timer counter reaches the value in the CCR.
The capture/compare unit can directly modify the output of a pin on this compare event What happens to the pin (set-high, set-low, toggle) depends on the mode in the control register.
(figure of edge-aligned toggle on match)
The capture compare mode register can be changed on-the-fly, moving the event point. This is often used in PWM mode.

4.4.3 Pulse-Width Modulation (PWM) Mode

What is PWM? What is it used for? How does it approximate an analog signal? low-pass filter electronic, mechanical (motor or speaker), others (human eye)
(figure of PWM representing sinewave)
How the STM32F0 generates PWM (fancier form of output compare) Perhaps in list mode where the registers involved are listed with their function? (figure of edge-aligned mode/sawtooth with appropriate markings)

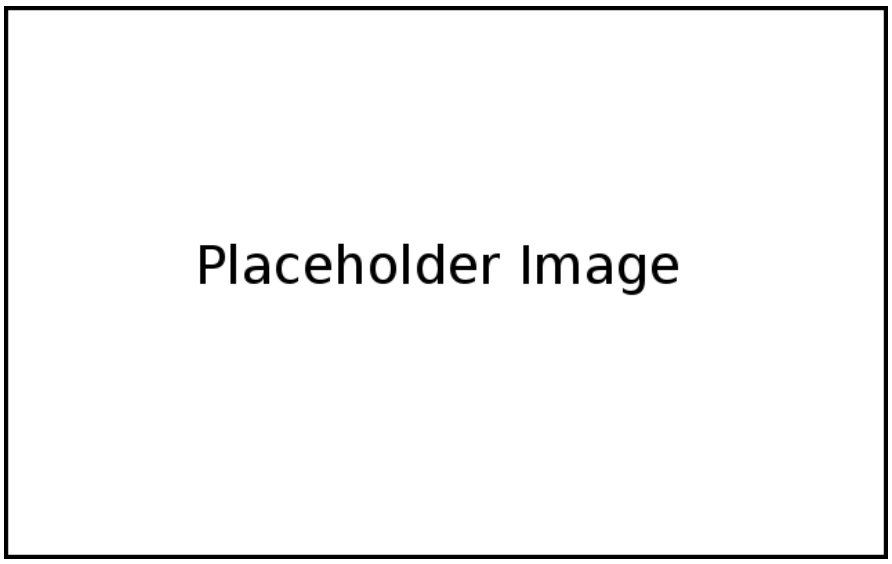

Figure 4.4: Output Compare mode with pin toggle on compare match.

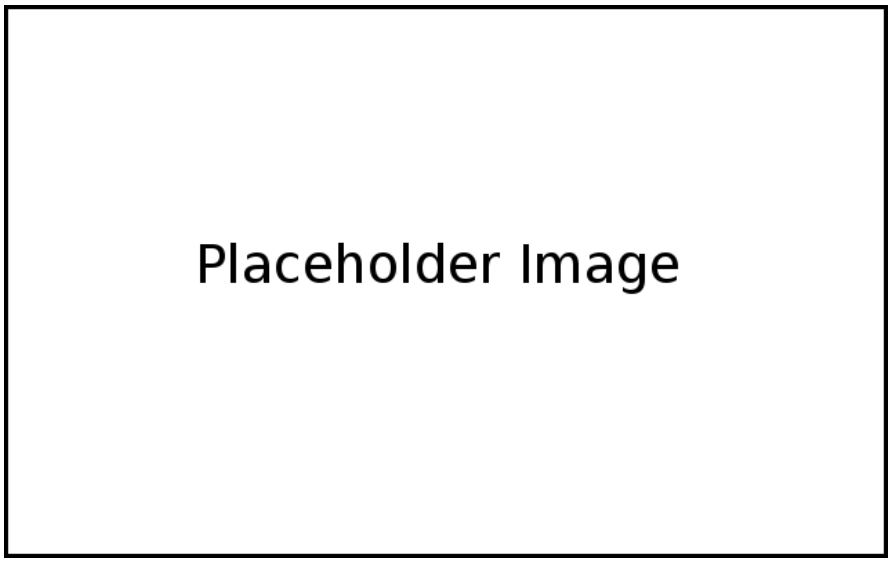

Figure 4.5: PWM approximation of an analog sine-wave.

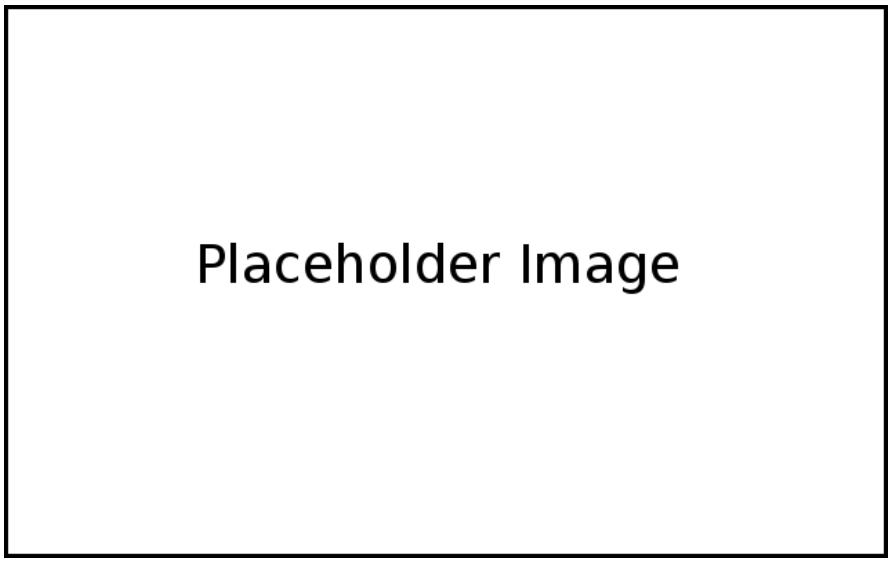

Figure 4.6: Edge-aligned PWM mode and output pin state.

4.5 Configuring GPIO pins to Alternate Function Mode

What is Alternate Function Mode? Need to tell the GPIO hardware to allow the timer to directly control the pin output state Connecting a pin to an internal peripheral of the device is called "Alternate Function Mode" In the STM32F0s most peripherals are directly connected to specific sets of pins. While you can't assign an internal signal to an arbitrary pin, they do give you a few pin options to choose. Most GPIO pins have multiple possible alternate functions, need to look up.
Use chip datasheet (Pinouts and pin descriptions section)

4.5.1 Finding Available Pins on a Device Package

Using a 64-pin LQFP (LQFP64) Table 13. STM32F072x8/xB pin definitions Only have pins that have physical pin numbers under the LQFP64 column. Table lists pin assignments for all chip packages the STM32F072 device is produced in. Alternate functions listed, will be talking about other columns of table in later labs.
(marked Figure of table 13, package column highlighted, pin name and possible alternate functions)

■ **Example 4.2 — Finding Pins With an Alternate Function.** Example: Chose arbitrary alternate function for a peripheral and walk through the process of finding pins ■

(add in section on how to graphically do this with stmcube later)

4.5.2 Selecting an Alternate Function

Demonstrate the AFR registers in the GPIO peripheral Once have pins that can use the AF, need to find what specific AF number it is for that pin. Use tables 14-19 in the chip datasheet
(marked figure of table 14-19, pick a GPIO and an alternate function)

■ **Example 4.3 — Determining an Alternate Function's Number.** Example: Find AF number and select for pin in previous example. ■

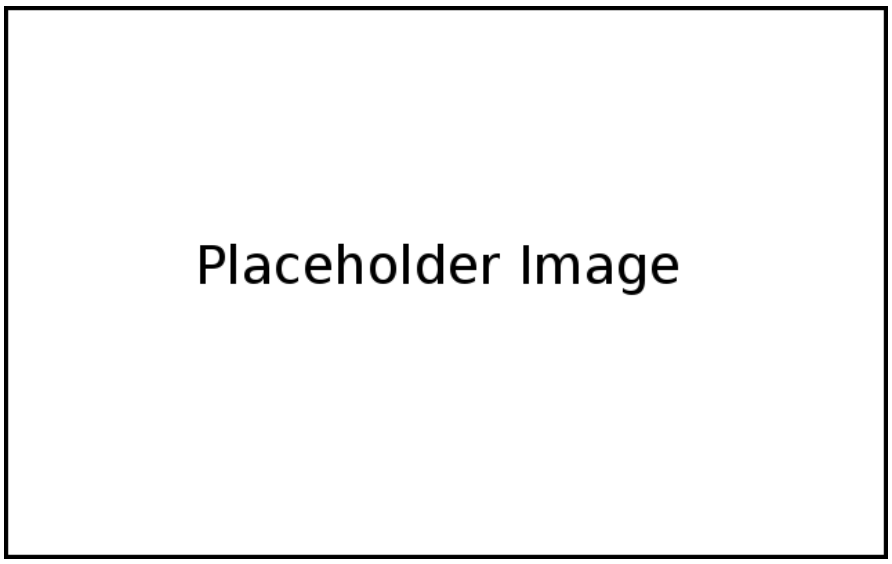

Figure 4.7: Subset of table 13 - STM32F072x8/xB Datasheet

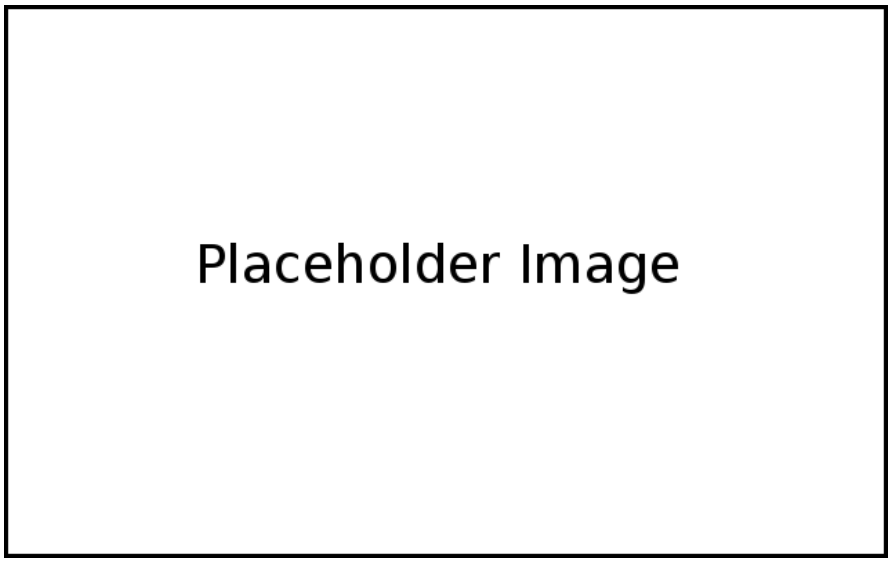

Figure 4.8: Subset of table 14 - STM32F072x8/xB Datasheet

4.6 Lab Assignment: