Programmable timing functions
Part 2: Timer operating modes

Textbook: Chapter 8, Section 8.6 (pulse-width modulation)
          Chapter 9, Sections 9.6, 9.7 (SysTick and Timer interrupts)

STM32F4xx Technical Reference Manual:
          Chapter 17  – Basic timers (TIM6)
          Chapter 15  – General-purpose timers (TIM4)
          Chapter 10  – Interrupt vectors (for TIM4/TIM6 interrupts)
Timer operating modes

Timer capture/compare channels provide operating modes other than periodic interrupts

- **Input capture mode**
  - Connect a GPIO pin to timer input TIMx_CHy
  - Capture CNT value in Capture/Compare Register CCRy at time of an event on the pin
    - Captures time at which the external event occurred
    - Use to measure time between events, tachometer signal periods, etc.

- **Output compare mode**
  - Connect timer output TIMx_CHy to a GPIO pin
  - Compare CNT to value in Capture/Compare Register CCRy
  - Change output pin when CNT = CCRy
    - Create a signal change/waveform/pulse/etc.

- **One pulse mode**
  - Setup similar to output compare mode
  - Disable the counter when the event occurs

- **Pulse-Width Modulated (PWM) waveform generation mode**
  - Setup similar to output compare mode
  - Force output pin active while CNT < CCRy
  - Force output pin inactive while CCRy ≤ CNT ≤ ARR
  - ARR sets PWM period, CCRy determines PWM duty cycle
Timer capture/compare channels

**Input capture:**
Copy CNT to CCRx when input event detected

**Output compare:**
Trigger an event when CNT = CCRx

---

**One-pulse**

- **OCxREF**

**Pulse-width modulation**

- **OCxREF**
  - active
  - inactive

**CNT**

- Period Start
  - CNT = CCRx = 3
  - (toggle OCxREF)
- CNT ≥ ARR = 7
  - (reset CNT and OCxREF)
General-purpose timers TIM2 – TIM5

Basic timer, plus:
Capture/compare support,
PWM generation,
Triggering options,

One “channel” – general-purpose timers have 1, 2, or 4 channels
Capture/Compare Channels

- Different channels but same blocks
  - Capture mode can be used to measure the pulse width or frequency
  - Input stage includes digital filter, multiplexing and prescaler
  - Output stage includes comparator and output control
  - A capture register (with shadow register)

Input Stage Example
- Input signal $\rightarrow$ filter $\rightarrow$ edge detector $\rightarrow$ slave mode controller or capture command

Diagram:
- From input pin $\rightarrow$ filter downcounter $\rightarrow$ Edge Detector $\rightarrow$ slave mode controller or capture command $\rightarrow$ To capture register
Capture/Compare Channels

- Main Circuit

The block is made of one preload register and a shadow register.

- In capture mode, captures are done in shadow register than copied into preload register
- In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter
**Route output OC1 to a GPIO pin as an “alternate function”. (each GPIO pin can connect to one or two timer channels)**
Configure the GPIO - AF

- Refer to the user manual to determine which pin is able to connect to TIMx channels (table of pin functions)
- Configure the GPIO pin as AF mode, be careful with the pull up or down setting since it should match the setting of edge detection
- Configure the GPIO AF register to select the TIMx channel for the pin
Alternate functions for pins PD12-13-14-15

From STM32F407 Data Sheet – Table 6

TIM4 can connect to GPIO pins (alt. function):
PD12 = TIM4_CH1
PD13 = TIM4_CH2
PD14 = TIM4_CH3
PD15 = TIM4_CH4

TIM4 inputs for input capture mode.
TIM4 outputs for output capture/PWM/one-pulse mode.

Discovery board LEDs driven by PD12-PD15.
Selecting an alternate function

- **TIMERS**
  - Only a subset of AF’s available at each pin, as listed in data sheet. (see previous slide)

- **GPIOn->MODER** selects AF mode for pins (10)
- **GPIOn->AFR[0]** selects AFs for pins Pn0-Pn7
- **GPIOn->AFR[1]** selects AFs for pins Pn8-Pn15

Example: Configure PA6 as TIM3_CH1 (AF2)

- `GPIOA->MODER &= ~0x00003000;` //clear PA6 mode
- `GPIOA->MODER |= 0x00002000;` //PA6 = AF mode
- `GPIOA->AFR[0] &= ~0x0F000000;` //clear AFRL6
- `GPIOA->AFR[0] |= 0x02000000;` //PA6 = AF2
TIMx capture/compare registers

- **TIMx_CCRy** = TIMx capture/compare register, channel y
  - TIMx_CCR1 – address offset 0x34
  - TIMx_CCR2 – address offset 0x38
  - TIMx_CCR3 – address offset 0x3C
  - TIMx_CCR4 – address offset 0x40
  - Register width (16/32 bits) same as CNT/ARR registers
  - TIMx may have 0, 1, 2, or 4 channels (see manual)

- **Output compare mode**: TIMx_CCRy compared to CNT, with match signaled on OCy output

- **Input capture mode**: CNT captured in TIMx_CCRy when designated input signal event occurs
Timer System Control Register 1

TIMx_CR1 address offset 0x00  (default = all 0’s)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARPE</td>
<td>CMS*</td>
<td>DIR*</td>
<td>OPM</td>
<td>URS</td>
<td>UDIS</td>
<td>CEN</td>
<td></td>
</tr>
</tbody>
</table>

**Center mode select**
- 00 = edge-aligned
  - count in one direction
- Others: center aligned
  - count in both directions

**Direction**
- 0 = count up
- 1 = count down

**Counter Enable**
- 0 = disable
- 1 = enable

**One Pulse Mode**
- 1 = counter stops at update event
- 0 = counter continues at UE

Advanced Options:
- ARPE = 1 enables ARR buffer (transferred to ARR on update event)
- URS = 0 allows multiple events to generate update interrupt
  - 1 restricts update interrupt to counter overflow/underflow
- UDIS = 0 enables update event to be generated

* Not in TIM6/TIM7
Timer Status Register

TIMx_SR address offset 0x10  (reset value = all 0’s)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC4IF</td>
<td>CC3IF</td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UIF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Capture/compare interrupt flags
1 = update interrupt pending
0 = no update occurred

Set by hardware on capture/comp event
Cleared by software
(reset CCxIF bit to 0)

Update interrupt flag
1 = update interrupt pending
0 = no update occurred

Set by hardware on update event
Cleared by software
(reset UIF bit to 0)
Timer Status Register

TIMx_SR address offset 0x10 (reset value = all 0's)

Capture/compare interrupt flags
1 = capture/compare interrupt pending pending
0 = no capture/compare event occurred

Set by hardware on capture/comp event
Cleared by software
(reset CCxIF bit to 0)

Update interrupt flag
1 = update interrupt pending pending
0 = no update occurred

Set by hardware on update event
Cleared by software
(reset UIF bit to 0)
Timer DMA/Interrupt Control Register

TIMx_DIER address offset 0x0C  (default = all 0’s)

8  7  6  5  4  3  2  1  0

UDE  CC4IE  CC3IE  CC2IE  CC1IE  UIE

Update DMA request enable
1 = enable
0 = disable

Capture/Compare interrupt enable
1 = CCx interrupt enabled
0 = disabled
TIMx interrupt on capture/compare event

Update interrupt enable
1 = enable
0 = disable
Timer DMA/Interrupt Enable Register

TIMx_DIER address offset 0x0C (reset value = all 0’s)

<table>
<thead>
<tr>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **UDE** (Update DMA request enable)
  - 1 = enable, 0 = disable

- **UIE** (Update interrupt* enable)
  - 1 = enable, 0 = disable

- **CC4IE**
  - Capture/Compare interrupt* enable
    - TIMx interrupt on capture/compare event
    - 1 = CCx interrupt enabled, 0 = disabled

* Capture/compare and update events generate the **same IRQn signal**, and use the **same interrupt handler**. Handler reads status register flags to determine source.
Capture/Compare Mode Registers

TIMx_CCMR1 : bits 7:0 configure channel 1; bits 15:8/channel 2
TIMx_CCMR2 (TIM2-3-4): bits 7:0/channel 3; bits 15:8/channel 4
(reset values = all 0’s)

Output mode ->
Input mode** ->
** discussed later

Output Compare 1 Mode
000 = frozen (no events)
001 = Set CH1 active* on match
010 = Set CH1 inactive* on match
011 = Toggle CH1 on match
100 = Force CH1 to inactive* (immediate)
101 = Force CH1 to active* (immediate)
110 = PWM mode 1 (active* to inactive*)
111 = PWM mode 2 (inactive* to active*)

Capture/Compare 1 Select
00 = output
01 = input**: IC1 = TI1
10 = input**: IC1 = TI2
11 = input**: IC1 = TRC

* Active/inactive levels selected in TIMx_CCER register
### Capture/compare mode register \( \frac{1}{2} \)

**Input capture mode**

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-6</td>
<td>OC1CE</td>
</tr>
<tr>
<td>5-4</td>
<td>OC1M[2:0]</td>
</tr>
<tr>
<td>3-2</td>
<td>OC1PE</td>
</tr>
<tr>
<td>1-0</td>
<td>OC1FE</td>
</tr>
</tbody>
</table>

#### Input Capture 1 Filter
- Defines frequency used to sample TI1 input and length of digital filter applied to TI1

#### Input Capture 1 Prescaler
- 00: capture on every event
- 01: capture on every 2\(^{nd}\) event
- 10: capture on every 4\(^{th}\) event
- 11: capture on every 8\(^{th}\) event

#### Capture/Compare 1 Select
- 00 = output
- 01 = input: IC1 = TI1
- 10 = input: IC1 = TI2
- 11 = input: IC1 = TRC

---

**OC1xxx** = function if CC1S selects “output”

**IC1xxx** = functions if CC1S selects “input”

Active/inactive level selected in TIMx_CCER register

CCMR2 configures Channels 3/4
Capture/Compare Enable Register

TIMx_CCER address offset 0x20 (reset value = all 0’s)

<table>
<thead>
<tr>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 - 12</td>
</tr>
<tr>
<td>11 - 8</td>
</tr>
<tr>
<td>7 - 4</td>
</tr>
<tr>
<td>CC4</td>
</tr>
<tr>
<td>CC3</td>
</tr>
<tr>
<td>CC2</td>
</tr>
</tbody>
</table>

**CC1 Polarity**
If CC1 = **output**, CC1P selects:
- 0 = OC1 active high
- 1 = OC1 active low

If CC1 = **input**:
CC1NP/CC1P select capture trigger:
- 00: falling edge of input
- 01: rising edge of input
- 11: both edges of input

**CC1 Enable**
If CC1 = **output**:
- 1 = OC1 drives output pin
- 0 = OC1 does not drive output

If CC1 = **input**:
- 1 = Capture enabled
- 0 = Capture disabled
Output Compare Mode

- Change output pin state or indicate when a period of time has elapsed
- When a match occurs (CCRx = CNT):
  - Generate specified output on corresponding pin
  - Set CCxIF = 1 (interrupt flag) in the SR
  - Generate interrupt if configured (CCxIE = 1)
Input capture mode

TIMx_CCRx latches TIMx_CNT value when transition detected
- CCxIF flag sets and interrupt generated, if enabled
- Signal edge programmable (rising, falling, both)

Example:
PWM input mode:
2 ICx signals of opposite polarity
Wind Speed Indicator (Anemometer)

- Rotational speed (and pulse frequency) is proportional to wind velocity
- Two measurement options:
  - Frequency (best for high speeds)
  - Width (best for low speeds)
- Can solve for wind velocity $v$
- How can we use the Timer for this?
  - Use Input Capture Mode to measure period of input signal

Diagram showing the relationship between $T$, $T_1$, and $T_2$.
Input Capture Mode for Anemometer

- **Operation: Repeat**
  - First capture - on rising edge
    - Reconfigure channel for input capture on falling edge
    - Clear counter, start new counting
  - Second Capture - on falling edge
    - Read capture value, save for later use in wind speed calculation
    - Reconfigure channel for input capture on rising edge
    - Clear counter, start new counting

- **Solve the wind speed**
  - \( V_{\text{wind}} = K \div (C_{\text{falling}} - C_{\text{rising}}) \times Freq \)
Set up for Anemometer measurement

- Apply *Anem_Out* signal to pin PD15
  - TIM4_CH4 is an alternate function for PD15 (from data sheet)
  - Configure PD15 as alternate function in GPIOD_MODER
  - Select alternate function TIM4_CH4 for PD15 in GPIOD_AFRH
- Configure **TIM4_PSC** and **TIM4_ARR** for TIM4 counting period
  - Best if counting period > time to be measured
  - Reset **TIM4_CNT** after each capture
- **TIM4_CCMR2** Capture/Compare mode register 2 (Channels 3 and 4)
  - Set CC4S to map IC4 on TI4
  - Set IC4F, IC4PSC to defaults (no filter or prescale)
- **TIM4_CCRER** Capture/compare enable register
  - Set CC4E to select “input” mode
  - Set CC4N:CC4P = 00 to select rising-edge (01 for falling edge)
- **TIMx_DIER** DMA/interrupt enable register
  - Set CC4IE to enable interrupt on input capture event *(to read captured value)*
- **TIM4_CR1** Control register: Set CEN to enable the counter
- **TIM4_SR** Status register: CC1IF indicates input event occurred *(clear by software)*
- **TIM4_CCR4** Capture/Compare register = captured value of TIM4_CNT
Output Compare Mode

- Control an output waveform or indicating when a period of time has elapsed
- When a Match occurs (CCRx=CNT)
  - Generate specific output on corresponding pin
  - Set the CCxIF(Interrupt status) bit in the SR
- Generate Interrupt if configured
- Generate DMA request if configured

**Configure steps**
- Select the counter clock
- Write the desired data in ARR and CCR registers
- Enable Interrupt or DMA request if needed
- Select the output mode
- Enable the counter
Pulse-Width Modulation

- **Uses of PWM**
  - **Digital power amplifiers** are more efficient and less expensive than analog power amplifiers
    - Applications: motor speed control, light dimmer, switch-mode power conversion
    - Load (motor, light, etc.) responds slowly, averages PWM signal
  - **Digital communication** is less sensitive to noise than analog methods
    - PWM provides a *digital encoding* of an *analog* value
    - Much less vulnerable to noise

- **PWM signal characteristics**
  - Modulation frequency – how many pulses occur per second (fixed)
  - Period – \(1/(\text{modulation frequency})\)
  - On-time – amount of time that each pulse is on (asserted)
  - Duty-cycle – on-time/period
  - Adjust *on-time* (hence *duty cycle*) to represent the analog value
PWM to Drive Servo Motor

- Servo PWM signal
  - 20 ms period
  - 1 to 2 ms pulse width
Pulse-Width Modulation (PWM) Mode

- PWM produced by comparing TIMx_CNT to both TIMx_CCRy and TIMx_ARR
  - Set TIMx_ARR = Period
  - Set TIMx_CCRy = Duty
- TIMx_CCMRn (capture/compare mode)
  - Set bit CCxE = 1 to configure the channel as output
  - Set bits OCxM = 110 (PWM mode 1) – active if CNT < CCRy, inactive otherwise
    OCxM = 111 (PWM Mode 2) - inactive if CNT < CCRy , active otherwise
- TIMx_CCER:
  - Set bit CCxP = 0/1 to select active level high/low (output polarity) of OCx
  - Set bit CCxE = 1 to enable OCx to drive the output pin
- Configure GPIO MODER and AF registers to select alt. function TIMx_CHn for the pin

Duty cycle = \((\text{Duty/Period}) \times 100\%\)
PWM Signal Examples

1. OCXREF active (high) when TIMx_CNT < TIMx_CCRx
   
   Assumes OCxM = 110 and CCxP = 1

2. OCXREF inactive (low) when TIMx_CNT ≥ TIMx_CCRx

3. Update Event when TIMx_CNT = TIMx_ARR (resets TIMx_CNT to 0)
Edge-Aligned

- CMS bits in TIMx_CR1 are 00
- As long as TIMx_CNT<TIMx_CCRx then the OCXREF is high
- Select down-counting or up-counting using DIR bit in TIMx_CR1
Example:
20KHz PWM signal with 10% duty cycle on pin PB6

- Configure TIM4, Channel 1
  - Since TIM4_CH1 = AF2 for pin PB6
- Assume timer clock = 16MHz and prescale = 1
  - PWM Period = 16MHz/20KHz = 800 = TIM4_ARR
  - PWM Duty = 800 x 10% = 80 = TIM4_CCR1
- Configure TIM4_CCMR1 bits:
  - CC1E = 0 (make channel 1 an output)
  - CC1M = 110 (PWM mode 1: active-to-inactive)
- Configure TIM4_CCER bits:
  - CC1P = 0 to define OC1 as active high
  - CC1E = 1 to enable output OC1 to drive the pin
- Configure PB6 as alternate function TIM4_CH1
  - Select AF mode for PB6 in GPIOB->MODER
  - Select TIM4_CH1 (AF2) for PB6 in GPIOB->AFRL