ELEC 3040/3050
Lab #7

PWM Waveform Generation

STM32L100RC Data Sheet
Goals of this lab exercise

- Begin the primary design project for the semester
  - Speed controller for a D.C. motor
- Generate a pulse-width-modulated (PWM) waveform with keypad-selectable duty cycle
  - Using a programmable timer

The generated waveform will be amplified in the next lab to drive a D.C. motor
Motor Speed Control Project

1. Generate a PWM waveform
2. Amplify the waveform to drive the motor
3. Measure motor speed
4. Measure motor parameters
5. Control speed with a PID or other controller

Diagram:
- 9V Power Supply
- Amplifier
- 12V DC Motor
- Tachometer
- Frequency/Amplitude Measurement
- Computer System
PWM Digital Waveforms

- A pulse-width modulated (PWM) waveform is a periodic signal comprising pulses of varying duration.
- Modulation refers to modifying the pulse width (with period held constant) to achieve a desired effect.
  - “Effect” often an average voltage to control a device.
- PWM signals are often used to drive D.C. motors, commercial lights, etc.
PWM to Drive a Servo Motor

- Servo PWM signal
  - 20 ms period
  - 1 to 2 ms pulse width
PWM Waveform Parameters

T = period of waveform (constant)
T1 = duration of pulse
T2 = T - T1

*Duty Cycle* = $T1/T = T1/(T1+T2)$

$$V_{avg} = V_{max} \times Duty\ Cycle$$

Pulses can also be active-low.
Timer operating modes

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

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

- **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

- **One pulse mode** – Create a single pulse on a pin
  - Setup similar to output compare mode
  - Disable the counter when the event occurs

- **Input capture mode** – Capture time at which an external event occurs
  - 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
  - Use to measure time between events, tachometer signal periods, etc
General-purpose timers TIM10/TIM11

Basic timing function (earlier lab)

* 2.097MHz if default MSI clock used
  (0x0020_0000 cycles/sec)
* 16 MHz if HSI clock used

Capture/Compare Channel 1 – TIMx_CH1 input/output

2 channels in TIM9, 4 channels in TIM2-3-4, no channels in TIM6-7
TIM6-7-10-11 have up counters, TIM2-3-4-9 have up/down counters
Timer capture/compare channels

Input capture:
Copy CNT to CCRx when input event detected

Output compare:
Trigger an event when CNT = CCRx

One-pulse

Pulse-width modulation

OCxREF

CNT

Period Start
CNT=CCRx=3 (toggle OCxREF)
CNT≥ARR=7 (reset CNT and OCxREF)
Capture/Compare Output Stage

Comparator Outputs

Mode = Output Compare or PWM

** Route output OC1 to a GPIO pin as an “alternate function”. (each GPIO pin can connect to one or two timer channels)
# Timer outputs as GPIO pin alternate functions

Each GPIO pin configurable as: INPUT, OUTPUT, ANALOG, **ALTERNATE FUNCTION**
- Select pin modes in **GPIOx->MODER** \((10 = \text{alternate function})\)

From STM32L100RX Data Sheet Table 7. “Pin Definitions” (partial)

<table>
<thead>
<tr>
<th>Pins</th>
<th>Pin name</th>
<th>Type(1)</th>
<th>I / O Level(2)</th>
<th>Main function (after reset)</th>
<th>Alternate functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQFP64</td>
<td>PA5</td>
<td>I/O</td>
<td></td>
<td>PA5</td>
<td>TIM2_CH1_ETR/SPI1_SCK/ADC_IN5/DAC_OUT2/COMP1_INP</td>
</tr>
<tr>
<td></td>
<td>PA6</td>
<td>I/O</td>
<td>FT</td>
<td>PA6</td>
<td>TIM3_CH1/TIM10_CH1/ SPI1_MISO/LCD_SEG3/ADC_IN6/COMP1_INP/OPAMP2_VINP</td>
</tr>
<tr>
<td></td>
<td>PA7</td>
<td>I/O</td>
<td>FT</td>
<td>PA7</td>
<td>TIM3_CH2/TIM11_CH1/SPI1_MOSI/LCD_SEG4/ADC_IN7/COMP1_INP/OPAMP2_VINM</td>
</tr>
</tbody>
</table>

We will use **TIM10_CH1** (Pin PA6)
Selecting an alternate function

**Timers**

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

**AFR[0]**:
AFRLn defines pin n, n=0..7

**GPIOOn->MODER** selects AF mode for pins (10)
**GPIOOn->AFR[0]** selects AFs for pins Pn0-Pn7
**GPIOOn->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
```
Timer System Control Register 1

TIMx_CR1 (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>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

* TIM6-7-10-11 limited to count up:
  - DIR = 0 & CMS = 00 only

*CEN only bit that needs to be changed for simple PWM

See timer overview from earlier lab
Timer Status Register

TIMx_SR (reset value = all 0’s)

7 6 5 4 3 2 1 0

CC4IF CC3IF CC2IF CC1IF UIF

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

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

UART1 interface status register

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)

TIM10 has only CC1IF

See timer overview from earlier lab
**Timer DMA/Interrupt Enable Register**

TIMx_DIER (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>CC4IE</td>
<td>CC3IE</td>
<td>CC2IE</td>
<td>CC1IE</td>
<td>UIE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Update DMA request enable**
  - 1 = enable, 0 = disable

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

- **Update interrupt* enable**
  - 1 = enable, 0 = disable

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

See timer overview from earlier lab

TIM10 has only CC1IE
Capture/Compare Register

- Compared to TIMx_CNT to trigger operations at specified times.
- TIMx_CCRy = TIMx capture/compare register, channel y
  - TIM2-3-4: y=1,2,3,4;  TIM9: y = 1,2;  TIM10-11: y=1
  - CCRy register width same as CNT/ARR registers (16 bits)

- Input capture mode: TIMx_CNT captured in TIMx_CCRy when a designated input signal event is detected
- Output compare mode: TIMx_CCRy compared to TIMx_CNT; each match is signaled on OCy output
- One pulse mode: same as output compare, but disable after match
- PWM mode: TIMx_CCRy compared to TIMx_CNT
  - CNT < CCRy => output active
  - CNT ≥ CCRy => output inactive

TIMx_CNT operates as discussed previously for periodic interrupt generation:
- Signal update event and reset to 0 when CNT = ARR while counting up
- Signal update event and reload ARR when CNT = 0 while counting down
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)

** discussed later

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*)

* Active/inactive levels selected in TIMx_CCER register

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

TIMx_CCER (reset value = all 0’s)

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

**Channel 1**

**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 CCxF = 1 (interrupt flag) in the SR
  - Generate interrupt if configured (CCxE = 1)

![Diagram of TIMx_CNT and TIMx_CCR1 registers with B201h set in CC1R register, indicating a match on CCR1 and an interrupt if enabled.]
Pulse-Width Modulation (PWM) Mode

Output pin

(Duty/Period) x 100%

PWM by comparing TIMx_CNT to both TIMx_CCRy and TIMx_ARR
- Set TIMx_ARR = Period
- Set TIMx_CCRy = Duty

TIMx_CCMRn (capture/compare mode) (n=1 for channels 1-2 / n=2 for channels 3-4):
- Set bits CCyS = 00 to select an output mode for channel y
- Set bits OCyM = 110 (PWM mode 1) – active if CNT < CCRy, inactive otherwise
  OCyM = 111 (PWM Mode 2) - inactive if CNT < CCRy , active otherwise

TIMx_CCER:
- Set bit CCyE = 1 to enable OCy to drive the output pin
- Set bit CCyP = 0/1 to select active level high/low (output polarity) of OCy

Configure GPIO MODER and AF registers to select alt. function TIMx_CHy for the pin
PWM Signal Examples

1. OCXREF active (high) when $\text{TIMx_CNT} < \text{TIMx_CCRx}$
   
   Assumes $\text{OCxM} = 110$ and $\text{CCxP} = 1$

2. OCXREF inactive (low) when $\text{TIMx_CNT} \geq \text{TIMx_CCRx}$

3. Update Event when $\text{TIMx_CNT} = \text{TIMx_ARR}$ (resets $\text{TIMx_CNT}$ to 0)
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:
  - CC1S = 00  (make channel 1 an output)
  - CC1M = 110  (PWM mode 1: active-to-inactive)
- Configure TIM4_CCER bits:
  - CC1E = 1 to enable output OC1 to drive the pin
  - CC1P = 0 to define OC1 as active high
- Configure PB6 as alternate function TIM4_CH1
  - Select AF mode for PB6 in GPIOB->MODER
  - Select TIM4_CH1 (AF2) for PB6 in GPIOB->AFRL

*What if timer clock = 2.097 MHz ?
(0x0020_0000 Hz)
Lab Procedure

- Generate a PWM waveform with timer TIM10
  - Period should be 1 ms (frequency 1 KHz)
  - First, generate a waveform with one duty cycle value
  - Then, verify that you can generate waveforms with each of the 11 specified duty cycles, from 0% to 100%, as selected by keypad keys 0 – A.
    - Measure and record the 11 duty cycle values
    - Plot measured duty cycle vs. selection key #

- Repeat with PWM frequency = 100 Hz
  - What needs to be changed?

  (Time permitting) Repeat with PWM frequency = 10 KHz