# LAB: Input Capture – Ultrasonic Distance Sensor

## I. Introduction

In this lab, you are required to create a simple program that uses input capture mode to measure the distance using an ultrasonic distance sensor. The sensor also needs trigger pulses that can be generated by using the timer output.

You will also create HAL drivers for Timer Input Capture and use these APIs for the lab.

### Hardware

NUCLEO -F411RE

HC-SR04, breadboard

### Software

Keil uVision IDE, CMSIS, EC_HAL

## II. Procedure

### A. Create EC_HAL functions

Specific for given Output Pins

| Include File | Function | Description |
|---|---|---|
| **ecTIM.h, c** | uint32_t is_UIF(TIM_TypeDef *TIMx);<br>void clear_UIF(TIM_TypeDef *TIMx); | Initialize timer counter period of usec.<br>For Timerx= TIM1, TIM2, … |
| | uint32_t is_CCIF(TIM_TypeDef *TIMx, uint32_t ccNum);<br>void clear_CCIF(TIM_TypeDef *TIMx, uint32_t ccNum); | Restrict CCnum= 1~4.<br>Add exception handling |

```
void TIM_INT_enable(TIM_TypeDef* timx);
void TIM_INT_disable(TIM_TypeDef* timx);
```

| | |
|---|---|
| `void ICAP_init(IC_t *ICx, GPIO_TypeDef *port, int pin);` | Initialize input capture mode with given GPIO port and pin and enable each timer clock.<br>// Initial Default Setting<br>Input PSC: 1 Capture Compare & Update Interrupt enabled<br><br>* You can choose other default values |
| `void ICAP_setup(IC_t *ICx, int ICn_type, int edge_type);` | |
| `void ICAP_counter_us(IC_t *ICx, int usec);` | |

You can refer to [example code using mbedOS]

## B. Ultrasonic Distance Sensor (HC-SR04)

The HC-SR04 ultrasonic distance sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

**The HC-SR04 Ultrasonic Range Sensor Features:**

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 5V
- Digital Output: 0V (Low)
- Sensing Angle: 30° Cone
- Angle of Effect: 15° Cone
- Ultrasonic Frequency: 40kHz
- Range: 2cm - 400cm

## C. Configuration

Create a new project named as “**LAB_TIMER_Inputcapture_Ultrasonic”.**

Name the source file as “**LAB_TIMER_Inputcapture_Ultrasonic.c”**

*You MUST write your name in the top of the source file, inside the comment section.*

## Configure Input and Output pins

| System Clock | GPIO |
|---|---|
| PLL 84MHz | PA_6: (PWM, TIM3_CH1)<br>AF, Push-Pull, No Pull-up Pull-down,Fast<br><br>PB_10: (IC, TIM2_CH3)<br>AF, No Pull-up Pull-down |
| **Input Capture** | **PWM** |
| TIM2_CH3: PB_10<br>Counter Clock 0.1MHz(10us)<br>TI3=IC3 (rising edge)<br>TI3=IC4 (falling edge) | TIM3_Ch1: GPIO A, Pin 6<br>PWM period: 50ms<br>PWM duty ratio: 10usec pulse width |

# D. Measurement of Distance

The program needs to

- Generate a trigger pulse as PWM to the sensor.
- Receive echo pulses from the ultrasonic sensor
- Measure the distance by calculating pulse-width of the echo pulse.
- Display measured distance in [cm] on serial monitor of Tera-Term for (a) 10mm, (b) 50mm (c) 100mm

Embedded Controller

TIMER_Inputcapture_example

```c
#include "stm32f411xe.h"
#include "math.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecUART_student.h"
#include "ecSysTIck.h"

uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float timeSt = 0;
float timeEnd= 0;

void setup(void);

int main(void){

  setup();


  while(1){
    distance = (float) timeInterval/58;
    printf("%f [cm]\r\n",distance);
    delay_ms(500);
  }
}

void TIM2_IRQHandler(void){
  if(is_UIF(TIM2)){                     // Update interrupt
    _________________                   // overflow count
    clear_UIF(TIM2);                    // clear update interrupt flag
  }
  if(is_CCIF(TIM2,3)){                  // TIM2_Ch3 (IC3) Capture Flag. Rising Edge Detect
    timeSt = _________________;         // Capture TimeStart from CC3
    clear_CCIF(TIM2,3);                 // clear capture/compare interrupt flag
  }
  else if(_________________ ){          // TIM2_Ch3 (IC4) Capture Flag. Falling Edge Detect
    timeEnd = _________________;        // Capture TimeEnd from CC4
    timeInterval = _________________;   // Total time of echo pulse
    ovf_cnt = 0;                        // overflow reset
    clear_CCIF(TIM2,4);                 // clear capture/compare interrupt flag
  }
}

void setup(){

  RCC_PLL_init();
  SysTick_init();
  UART2_init();

// PWM configuration ---------------------------------------------------------------
  PWM_t trig;                       // PWM1 for trig
  _________________                 // PWM init as PA_6: Ultrasonic trig pulse
  PWM_period_us(&trig,50000);       // PWM of 50ms period. Use period_us()
  PWM_pulsewidth_us(&trig,10);      // PWM pulse width of 10us

// Input Capture configuration -----------------------------------------------------------
  IC_t echo;                        // Input Capture for echo
  _________________                 // ICAP init as PB10 as input caputre
  ICAP_counter_us(&echo, 10);       // ICAP counter step time as 10us
  ICAP_setup(&echo, 3, RISE);       // TIM2_CH3 as IC3 , rising edge detect
  _________________                 // TIM2_CH3 as IC4 , falling edge detect

// Enable TIMx interrupt      ------------------------------------------------------------
  _________________                 // TIM2 Interrupt Enable

}
```

## Discussion

1) There can be an over-capture case, when a new capture interrupt occurs before reading the CCR value. When does it occur and how can you calculate the time span accurately between two captures?

2) In the tutorial, what is the accuracy when measuring the period of 1Hz square wave? Show your result.

# III. Report

You are required to write a consice lab report and submit the program files.

**Lab Report:** See sample report.

- Write Lab Title, Date, Your name, Introduction
- For each Part show only main() source file. Also, need to include the external circuit diagram if necessary.
- Show your whole code **in the appendix,**
- Answer **Discussion questions**
- You can write Troubleshooting section
- Submit in both PDF and original file (*.docx etc)
- No need to print out. Only the On-Line submission.

**Source Code:**

- Write description of your functions in github.
- Upload the final version of your library in github.
- Zip all the necessary source files(main.c, ecRCC.h, ecGPIO.h etc...).
- Only the source code files. Do not submit project files etc.

# Appendix

## Timer GPIO pinout for STM32f411

**Timer PinOut Map**

**Advanced Timer**

| Timer | Channel | Port | Pin |
|---|---|---|---|
| 1 | 1 | A | 8 |
| | 1N | A | 7 |
| | | B | 13 |
| | 2 | A | 9 |
| | 2N | B | 0 |
| | | B | 14 |
| | 3 | A | 10 |
| | 3N | B | 1 |
| | | B | 15 |
| | 4 | | |
| | 4N | | |

**General Purpose Timer**

| Timer | Channel | Port | Pin |
|---|---|---|---|
| 2 | 1 | A | 0 |
| | | A | 5 |
| | | A | 15 |
| | 2 | A | 1 |
| | | B | 3 |
| | 3 | B | 10 |
| | 4 | | |
| 3 | 1 | A | 6 |
| | | B | 4 |
| | | C | 6 |
| | 2 | B | 5 |
| | | C | 7 |
| | 3 | C | 8 |
| | 4 | C | 9 |
| 4 | 1 | B | 6 |
| | 2 | B | 7 |
| | 3 | B | 8 |
| | 4 | B | 9 |

| Process | Sub-Process | Interrupt (Event) | Interrupt (CCR) | PWM | Input Capture |
|---|---|---|---|---|---|
| Clock | System CLK | RCC | | | |
| | TIM CLK Enable | RCC_APB1ENR | | | |
| Counter Setup | Period | PSC, ARR | | | |
| | DIR | DIR | | | |
| | Enable | CEN | | | |
| Interrupt | Flag | UIF | CCxIF | | UIF, CCxOF |
| | Enable | UIE | CCxE | | UIE, CCxIE |
| Compare& Capture | CC value | | CCR | CCR | CCR |
| Output Setup | Pin | | | GPIO AF | |
| | Output Mode | | | OCyM=110 (PWM1) OCyM=111 (PWM2) | |
| | Polarity | | | CCxP | |
| | Enable | | | CCxE | |
| Input Setup | Pin | | | | GPIO AF |
| | IC Select | | | | CCyS |
| | Filter | | | | ICyF |
| | Input Pre-scaler | | | | ICyPSC |
| | Polarity | | | | CCyP |
| | Enable | | | | CCxE |