

LAB: Input Capture - Ultrasonic

Date : 2023.10.31

Author/Partner: Duwon Yang

Github : [LINK](#)

Demo Video : [LINK](#)

Introduction

In this lab, We create a simple program that uses input capture mode to measure the distance using an ultrasonic distance sensor. check out distance using Tera Term program.

Requirement

Hardware

- MCU
 - NUCLEO-F411RE
- Actuator/Sensor/Others:
 - HC-SR04
 - breadboard

Software

- Keli uVision, CMSIS, EC_HAL library

Problem 1:Create HAL library

Create HAL library

Declare and Define the following functions in library. and update header files located in the directory `EC\lib\`.

ecTIM.h

```
/* Input Capture*/
// ICn selection according to CHn
#define FIRST 1
#define SECOND 2

// Edge Type
#define IC_RISE 0
#define IC_FALL 1
```

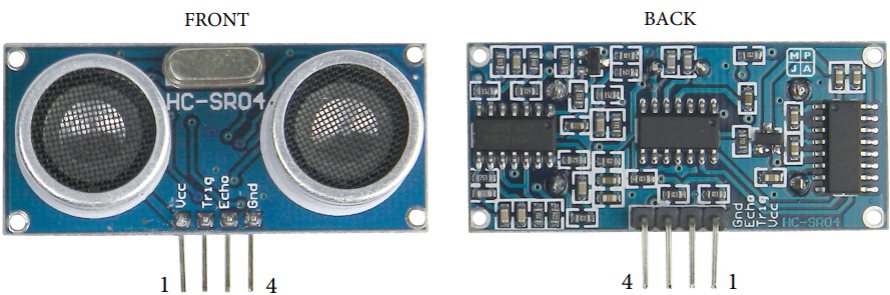
```
#define IC_BOTH 2

// IC Number
#define IC_1 1
#define IC_2 2
#define IC_3 3
#define IC_4 4

void ICAP_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);
void ICAP_init(PinName_t pinName);
void ICAP_setup(PinName_t pinName, int ICn, int edge_type);
void ICAP_counter_us(PinName_t pinName, int usec);
uint32_t ICAP_capture(TIM_TypeDef* TIMx, uint32_t ICn);
```

Problem2: 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.



	Pin Symbol	Pin Function Description
1	VCC	5V power supply
2	Trig	Trigger Input pin
3	Echo	Receiver Output pin
4	GND	Power ground

Electrical Parameters	HC-SR04 Ultrasonic Module
Operating Voltage	5VDC
Operating Current	15mA
Operating Frequency	40KHz
Max. Range	4m
Nearest Range	2cm
Measuring Angle	15 Degrees
Input Trigger Signal	10us min. TTL pulse
Output Echo Signal	TTL level signal, proportional to distance
Board Dimensions	1-13/16" X 13/16" X 5/8"
Board Connections	4 X 0.1" Pitch Right Angle Header Pins

Procedure

- 1. Create a new project under the directory `repos\EC\LAB\LAB_Timer_InputCatpure_Ultrasonic\`
 - The project name is **"LAB_Timer_InputCapture_Ultrasonic"**
 - Create a new source file named as **"LAB_Timer_InputCapture_Ultrasonic.c"**

2. Include updated library in `\repos\EC\lib\` to project.
- `ecGPIO.h, ecGPIO.c`
 - `ecRCC.h, ecRCC.c`
 - `ecTIM.h, ecTIM.c`
 - `ecPWM.h, ecPWM.c`
 - `ecSysTick.h, ecSysTick.c`
 - `ecUART_simple.h, ecUART_simple.c`
3. Connect the HC-SR04 ultrasonic distance sensor to MCU pins(PA6 - trigger, PB6 - echo), VCC and GND

Measurement of Distance

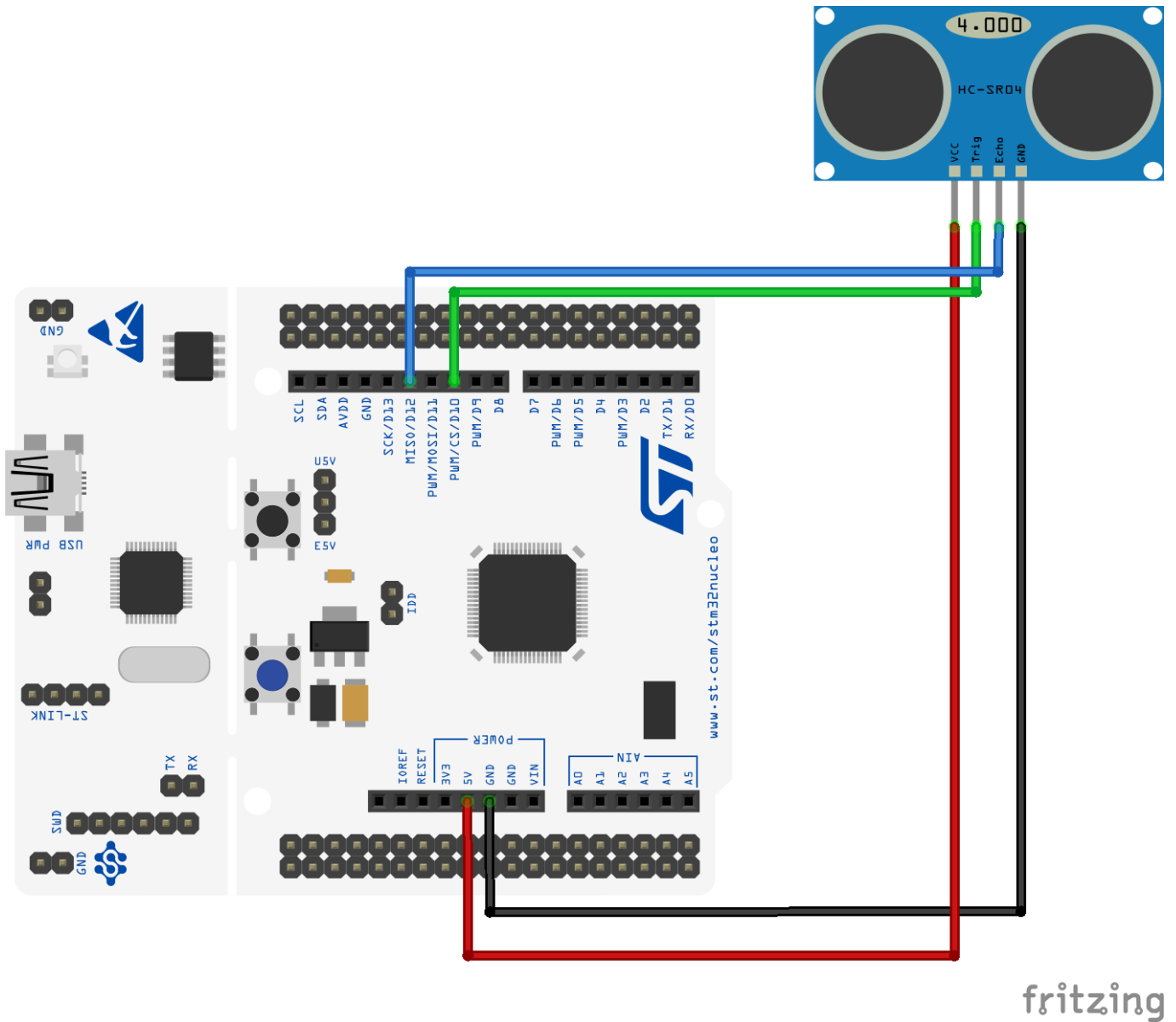
The program works to

- 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

Configuration

System Clock	PWM	Input Capture
PLL (84MHz)	PA6 (TIM3_CH1)	PB6 (TIM4_CH1)
	AF, Push-Pull, No Pull-up Pull-down, Fast	AF, No Pull-up Pull-down
	PWM period: 50msec pulse width: 10usec	Counter Clock : 0.1MHz (10us) TI4 -> IC1 (rising edge) TI4 -> IC2 (falling edge)

Circuit Diagram



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?

over-capture case can often occur when ISR(Interrupt Service routine) is slower than the consecutive pulse event.

if over-capture case is not occurring, time span can be derived from two captures(time1, time2)

time span = ((CCR2-CCR1)+Timer Max Value × Number of Overflows)×Timer Resolution

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

```

period = 1000.000000[msec]
period = 999.000000[msec]
period = 999.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 999.000000[msec]
period = 999.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 999.000000[msec]
period = 999.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 1000.000000[msec]
period = 999.000000[msec]
period = 1000.000000[msec]

```

To evaluate accuracy, I have received 30 data points as follows, and when comparing the average of the measured values to the standard value of 1000 msec, the accuracy was as follows.

$$\left(1 - \frac{1000 - \frac{1000 \times 23 + 999 \times 7}{30}}{1000}\right) \times 100 = 99.976\%$$

Code

This code is composed of main, TIM4_IRQHandler, and setup function.

Using PWM_pulsewidth_us function, PWM signal is generated and export through TRIG pin(PB6). then, ECHO pin(PA6) receives reflected signal using input capture.

Using TIMER4 Interrupt, ovf_cnt increase as overflow occurs, and save ECHO signal's rising and falling edge time using Input Capture. Through these procedures, distance is calculated and print through Tera-Term Program.

```

/**
*****
* @author  SSSLAB
* @Mod      2023-10-31 by DuwonYang
* @brief    Embedded Controller:  LAB - Timer Input Capture
*           - with Ultrasonic Distance Sensor
*
*****
*/

#include "stm32f411xe.h"
#include "math.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecUART_simple.h"
#include "ecSysTick.h"

uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;

#define TRIG PA_6
#define ECHO PB_6
void TIM4_IRQHandler(void);
void setup(void);

int main(void){

    setup();

    while(1){
        distance = (float) timeInterval * 340.0 / 2.0 / 10.0;    // [mm] -> [cm]
        printf("%f cm\r\n", distance);
        delay_ms(500);
    }
}

void TIM4_IRQHandler(void){
    if(is_UIF(TIM4)){                // Update interrupt
        ovf_cnt++;                    // overflow count
        clear_UIF(TIM4);              // clear update interrupt
    }
    if(is_CCIF(TIM4, 1)){             // TIM4_Ch1 (IC1) Capture
        Flag. Rising Edge Detect
        time1 = ICAP_capture(TIM4, IC_1);    //
        Capture TimeStart
        clear_CCIF(TIM4, 1);                // clear capture/compare interrupt
    }
}

```

```

flag
}
else if(is_CCIF(TIM4, 2)){ // TIM4_Ch2 (IC2)
Capture Flag. Falling Edge Detect
    time2 = ICAP_capture(TIM4, IC_2); //
Capture TimeEnd
    timeInterval = (time2 - time1 + ovf_cnt * (TIM4->ARR + 1)) * 10 / 1000; //
(10us * counter pulse -> [msec] unit) Total time of echo pulse
    ovf_cnt = 0; // overflow reset
    clear_CCIF(TIM4, 2); // clear capture/compare
interrupt flag
}
}

void setup(void){

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

// PWM configuration -----
-----
    PWM_init(TRIG); // 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 -----
-----
    ICAP_init(ECHO); // PB_6 as input caputre
    ICAP_counter_us(ECHO, 10); // ICAP counter step time as 10us
    ICAP_setup(ECHO, 1, IC_RISE); // TIM4_CH1 as IC1 , rising edge detect
    ICAP_setup(ECHO, 2, IC_FALL); // TIM4_CH2 as IC2 , falling edge detect
}

```

Results

In this lab, We create a simple program that measures a distance using Ultrasonic sensor through Input Capture function.

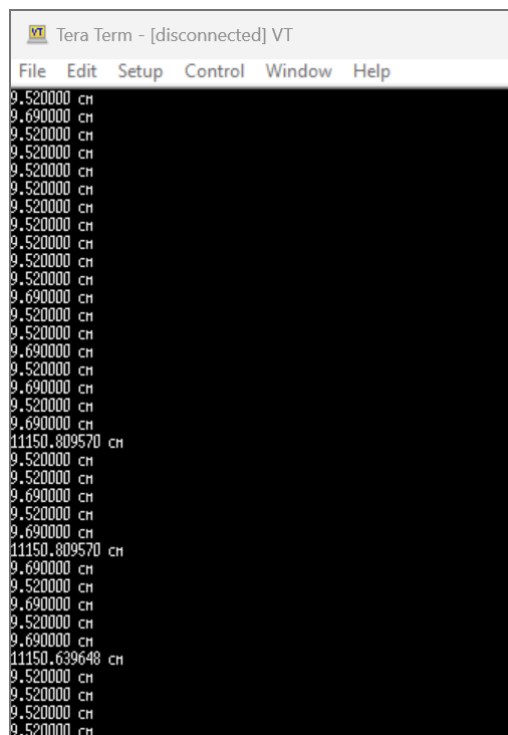
demo video : [LINK](#)

Reference

<https://ykkim.gitbook.io/ec/ec-course/lab/lab-report-template>

<https://ykkim.gitbook.io/ec/ec-course/lab/lab-input-capture-ultrasonic>

Troubleshooting



```
Tera Term - [disconnected] VT
File Edit Setup Control Window Help
9.520000 cm
9.690000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.690000 cm
9.520000 cm
9.520000 cm
9.690000 cm
9.520000 cm
9.690000 cm
9.520000 cm
9.690000 cm
11150.809570 cm
9.520000 cm
9.520000 cm
9.690000 cm
9.520000 cm
9.690000 cm
11150.809570 cm
9.690000 cm
9.520000 cm
9.690000 cm
9.520000 cm
9.690000 cm
11150.639648 cm
9.520000 cm
9.520000 cm
9.520000 cm
9.520000 cm
```

When measuring distances using an ultrasonic sensor, instability and spurious readings were occasionally observed. This phenomenon can be attributed to the uneven surfaces of the objects being measured, leading to incorrect reflections. To address this issue, it would be advisable to employ filtering techniques such as moving average filters or median filters.