BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 – Microprocessor and Embedded Systems Laboratory
January 2023

Experiment 04

Arm Cortex M Interrupts and Timers II: SysTick and Timer Input Capture

---

Evaluation Form:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
1	LED Toggling using Systick (Section 4)	15	
2	Distance Measurement using Ultrasonic Sensor (Section 5)	15	
3	Code Printout(Section 6)	20	
	TOTAL	50	

Signature of Evaluator: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. ***You will not receive credit for this lab experiment unless this statement is signed in the presence of your lab instructor.***

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Last Name (Printed):_____ **Lab Group:**_____ **Date:** _____

E-mail:_____**@**eee.buet.ac.bd **Signature:** _____

Contents

1 Introduction

Lab overview

At the end of this lab, you will be able to:

- Control an LED using Systick
- Measure distance by interfacing an ultrasonic sensor

2 Pre-lab Study

Before attempting this lab, please do the following:

1. Make sure you have completed tasks for Experiment 06 and 07
2. Study:
    - Read [Zhu] Textbook **Chapter 11 Interrupts (Section 11.1-11.5).**
    - Read [Zhu] Textbook **Chapter 15 General Purpose Timers (Section 15.1-15.3).**
    - Watch YouTube Tutorials on Interrupts (https://youtu.be/uFBNf7F3l60)
    - Watch YouTube Tutorials on SysTick (https://youtu.be/aLCUDv_fgoU)

3 Introduction to Interrupt

Recall from Experiment 3 that an interrupt is a signal sent by the hardware or software processes calling for the immediate attention of the CPU. Once the CPU receives this signal, it stops whatever it is doing and takes care of the request. When the interrupt signal is activated, the microcontroller branches to a program called interrupt service routine. The microcontroller then executes the subroutine. After the completion of ISR, the microcontroller returns back to the main program it was executing earlier.

Interrupts can be **Internal** or **External.** Internal interrupts can be come from Timers and External interrupts can come from external devices for example a push button or a keypad. In the previous experiment, we looked into external interrupts and how to configure them. In this experiment, we'll look into an internally generated interrupt Systick and also we'll apply the Timers in Input capture mode to investigate how to determine the interval between two events.

4 SysTick

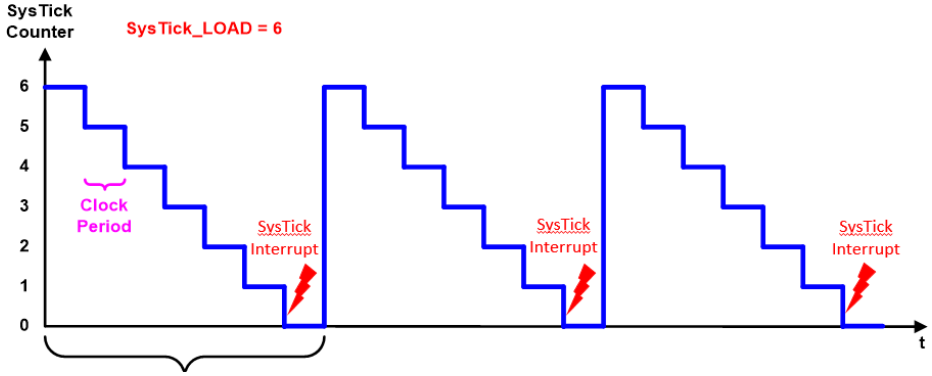
The SysTick exception is generated periodically by a dedicated timer. The system tick timer (SysTick) is a simple **24-bit down counter** to produce a small, fixed time quantum. Software uses SysTick to create time delays or generate periodic interrupts to execute a task repeatedly.

- The timer counts down from N-1 to 0, and the processor generates a SysTick interrupt once the counter reaches zero.
- After reaching zero, the SysTick counter loads the value held in a spedal register named the SysTick Reload register and counts down again.
- The SysTick timer does not stop counting down when the processor is halted. The processor still generates SysTick interrupts during the process of debugging.

Application:

SysTick Generates interrupts at a fixed time interval, which can be used for –

- Measuring time elapsed, such as time delay function
- Executing tasks periodically, such as periodic polling, and OS CPU scheduling



SysTick is Just a Timer!! But while other Timers are peripheral chosen by chip vendors, SysTick is defined by ARM. It has the same address on all Cortex M4 chip.

There are four 32-bit registers for configuring system timers.

I. **SysTick_CTRL:** SysTick control and status register
CLKSOURCE: 0 → external clock (AHB_clk/8), 1 → Processor clock
TICKINT: 0 → no interrupt generated, 1 → counting down to 0 asserts SysTick Interrupt

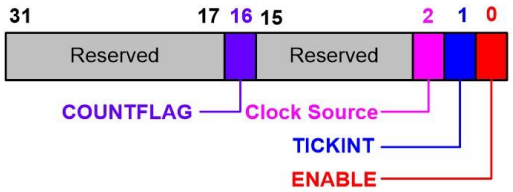
request

ENABLE: 0 → counter disabled, 1 → enabled

COUNTFLAG: 1 → Counter has transitioned from 1 to 0 since the last read of SysTick_CTRL,

0 → COUNTFLAG is cleared by reading SysTick_CTRL or by writing to SysTick_VAL.

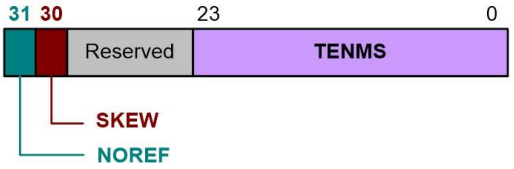
SysTick control and status register (SysTick_CTRL)



SysTick reload value register (SysTick_LOAD)

SysTick current value register (SysTick_VAL)

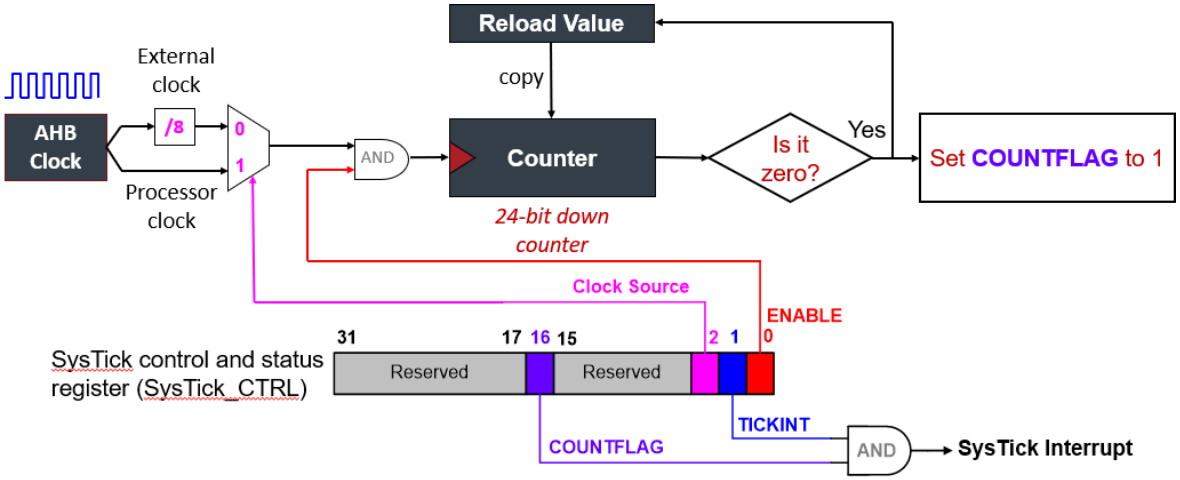
SysTick calibration register (SysTick_CALIB)



II. **SysTick_LOAD:** SysTick reload value register (like ARR)

III. **SysTick_VAL:** SysTick current value register (like CNT)

IV. **SysTick_CALIB:** SysTick calibration register, read only. Stores the required preload value for 10ms delay.



To enable SysTick interrupt, the program needs to set up the following bits:

- Set bit TICKINT in SysTick_CTRL to enable SysTick interrupt.
- Enable SysTick interrupt in the NVIC vector. Note that SysTick interrupt is enabled by default in the NVIC vector. Also set priority.
- Set bit ENABLE in SysTick_CTRL to enable the SysTick timer.
- Finally, write an ISR called SysTick_Handler(void), that performs the task you want to do periodically

4.1 Example Code:

```
void SysTick_Initialize (uint32_t ticks) {
SysTick->CTRL = 0;          // Disable SysTick SysTick->LOAD = ticks - 1; // Set
reload register
// Set interrupt priority of SysTick to least urgency (i.e., largest priority value)
NVIC_SetPriority (SysTick_IRQn, (1<< NVIC_PRIO_BITS) - 1); SysTick->VAL = 0;
        // Reset the SysTick counter value
// Select processor clock: 1 = processor clock; 0 = external clock SysTick->CTRL |=
SysTick_CTRL_CLKSOURCE;
// Enables SysTick interrupt, 1 = Enable, 0 = Disable SysTick->CTRL |=
SysTick_CTRL_TICKINT;
// Enable SysTick
SysTick->CTRL |= SysTick_CTRL_ENABLE;
}


void SysTick_Handler (void) { // SysTick interrupt service routine
// write code here
}


void main(void) {
// other codes SysTick_Initialize();
}
```

4.2 Lab Exercise

- Toggle an LED every 1s using SysTick
- Implement a function called mydelay() that takes time in ms as input and creates that delay.

5 Distance Measurement using Ultrasonic Sensor and Timer Input Capture:

5.1 What is Timer Input Capture?

In experiment 7, we have learnt to use timer in output compare mode and generated PWM signals with required duty cycle and period. Timers can also be configured to use as input capture mode. In this mode, timers act like stopwatches and measure the time spent between two events.

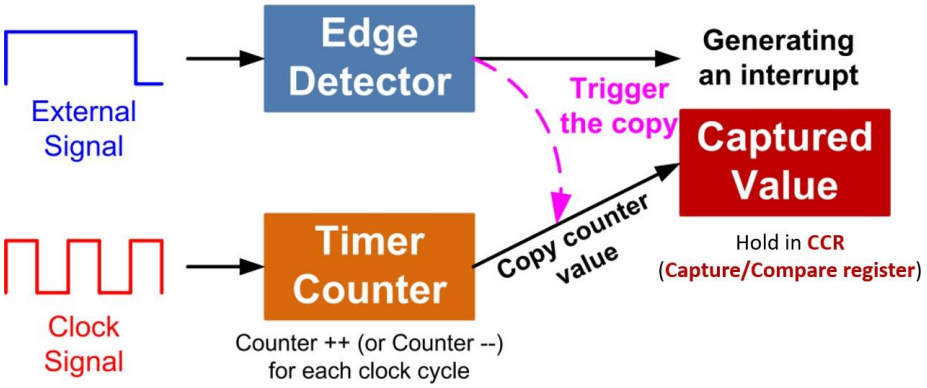


Fig 2: Timer used in input capture mode

Let us consider that we need to find out the time span for which the external signal (shown in above figure) is High. Let us denote the rising edge of this signal as event 1 and falling edge this signal as event 2. Also let the time period of Clock signal (in figure) be denoted as T_{Clk}. When the edge detector detects event 1, it copies the timer counter value (CNT1) into CCR register. Similarly, for event 2, counter value (CNT2) is copied into CCR register again. If the timer counts in up counting mode and no overflow is occurred during two events (i.e counter value did not reach ARR value), then we can find the time span between two events from:

$$\text{Time span between event 1 \&2} = \boldsymbol{TClk} \times (\boldsymbol{CNT2} - \boldsymbol{CNT1})$$

We will use this knowledge to measure the distance of an object for this lab. But it can be used anywhere we need to find time difference between two sensor measurements. For example, if we install a sensor in a wheel which measures the time span for a full rotation of the wheel, we can use the time span to calculate the velocity of the vehicle. For comprehensive reading, refer to chapter 15 of Yifeng Zhu's Embedded Systems book.

5.2 How does an Ultrasonic sensor work?

Ultrasonic sensors are used to measure distance of an object using sound waves. Their operation can be visualized from the picture below:

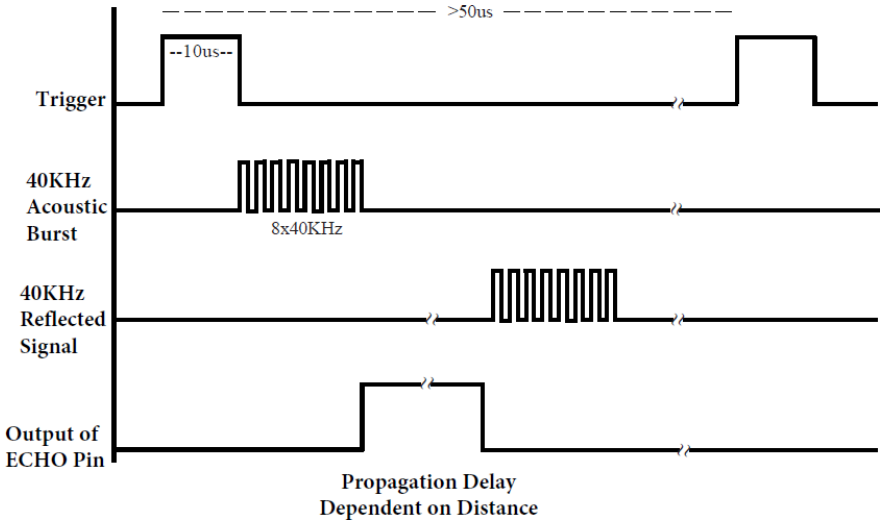


Fig 3: Operation of HC-SR04Ultrasonic Sensor

Ultrasonic sensors have 4 pins(Vcc, Trig, Echo and Gnd). If a trigger of at least10μs pulse width is applied to the Trig pin, the transmitter sends out 8 bursts of directional 40kHz ultrasonic wave and Echo pin is set High after all the bursts are sent. When the reflected signal starts being received by the receiver, echo pin is reset to Low. The sensor can measure a

distance between 2 *cm* and 400 *cm*, with a resolution of 0.3 *cm* and the corresponding echo pulse width is between 150 μs and 25 *ms*. When the sensor detects no object, the echo pulse width is 38 *ms*. To calculate the distance:

$$\text{Distance(in inches)} = \text{time } (\mu s)/148 \text{ or distance(in cm)} = \text{time } (\mu s)/58$$

Here 'time' implies the time span during which Echo pin was kept High. If we need continuous measurement of object distance, we need to continuously give signal of 10μs pulse width to Trig pin. Minimum time between two such signals is 50 μs. For more details, read datasheet and user manual of HC-SR04 Ultrasonic Sensor.

echo pin er rising edge and falling edge er difference mapbo

5.3 Circuit Connection:

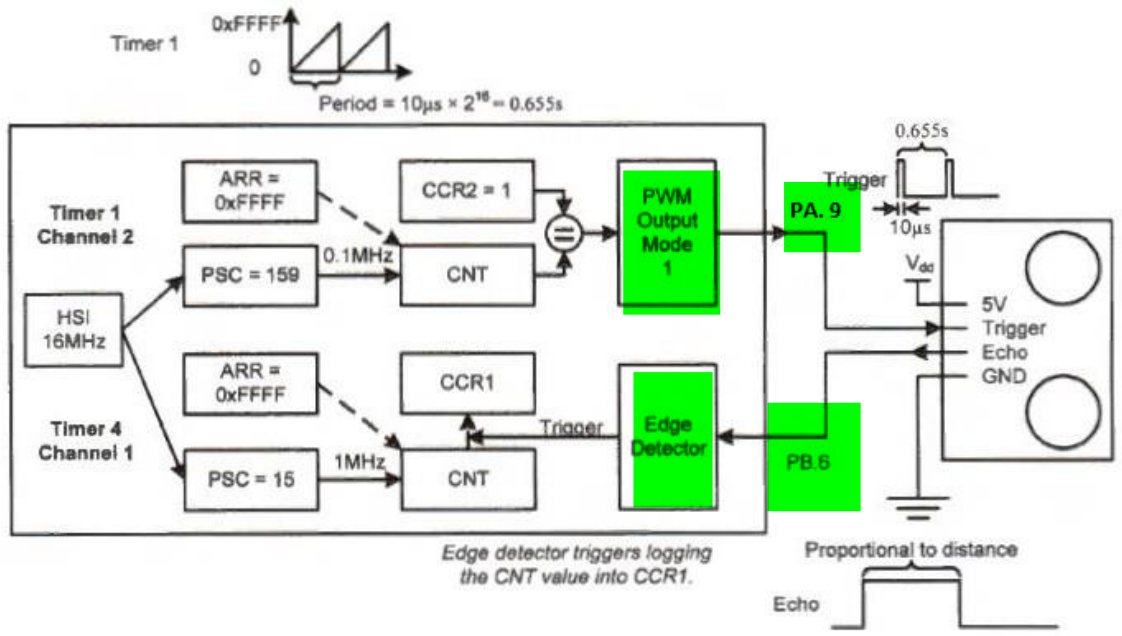


Fig 4: Timer configuration and Circuit connections for interfacing Ultrasonic Distance sensor

Here we will configure Port A pin 9 in output compare mode to generate the pulse required for Trigger pin of the sensor. On the other hand, Port B pin 6 will be configured in input capture mode to measure the time during which Echo signal remained High.

Trigger	PA9 (AF mode)	TIM1_CH2 (Output, PWM mode 1)	AF1
Echo	PB6 (AF mode)	TIM4_CH1 (Input Capture on rising and falling edge)	AF2

Which pins on your STM32 board can be configured into which functionalities can be found in its datasheet. For STM32F446RE, you can download the datasheet from https://www.st.com/resource/en/datasheet/stm32f446re.pdf .

5.4 Flowchart for configuring a pin in Input Capture mode:

To configure PB. 6(Timer4_Channel1) in input capture mode, we need to follow this flowchart (See Zhu's book for details):

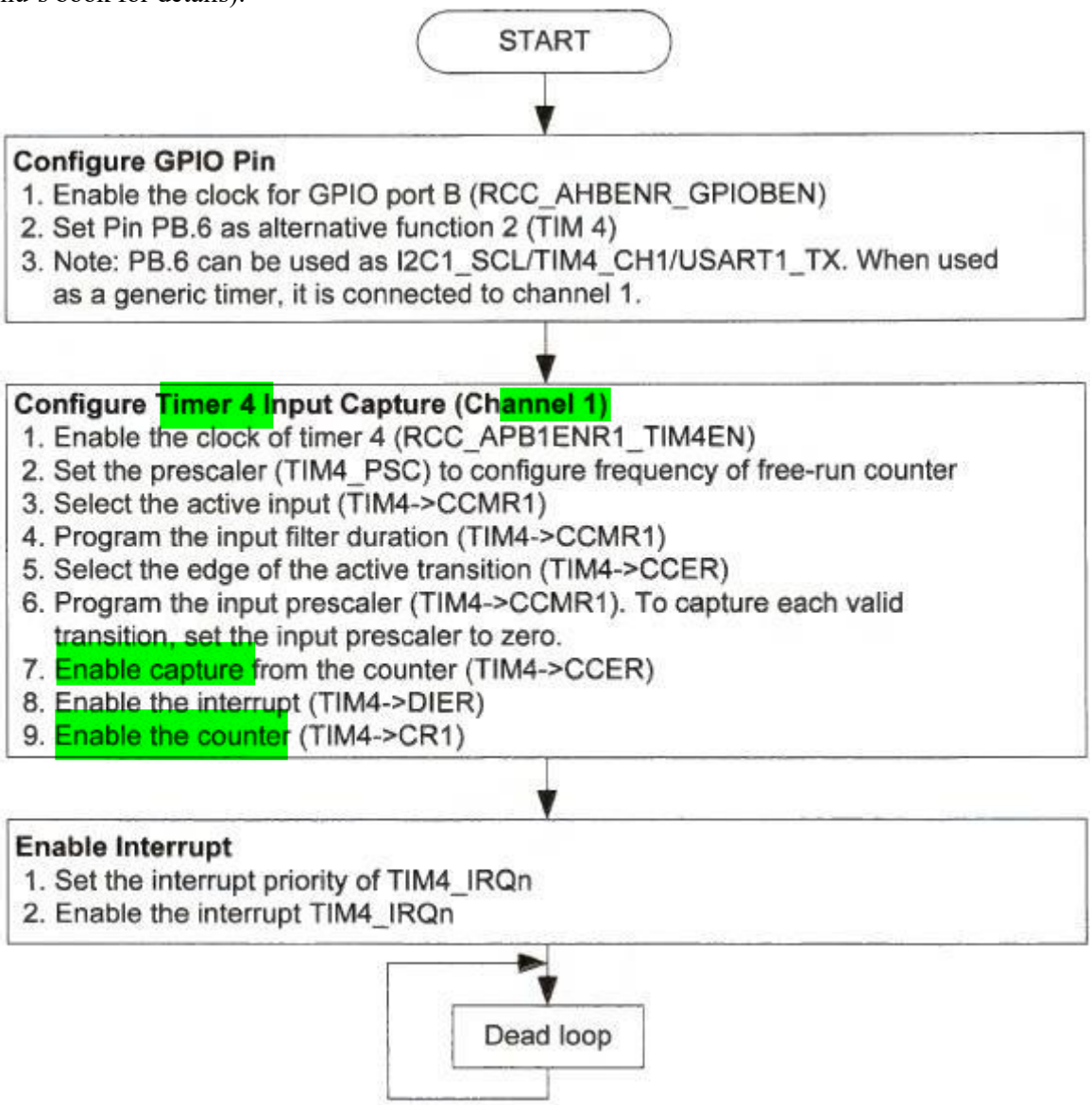


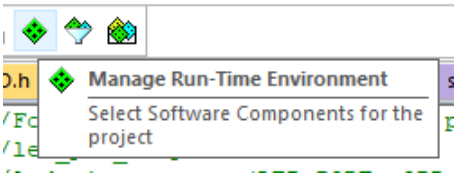
Fig 5: Configuring PB.6 in input capture mode

Two steps are missing in the above flowchart. After we set the pre-scalar (step 2 in 'Configure Input Capture'), we also need to set the ARR and counting direction of timer counter by accessing TIM4->ARR and TIM4->CR1 respectively.

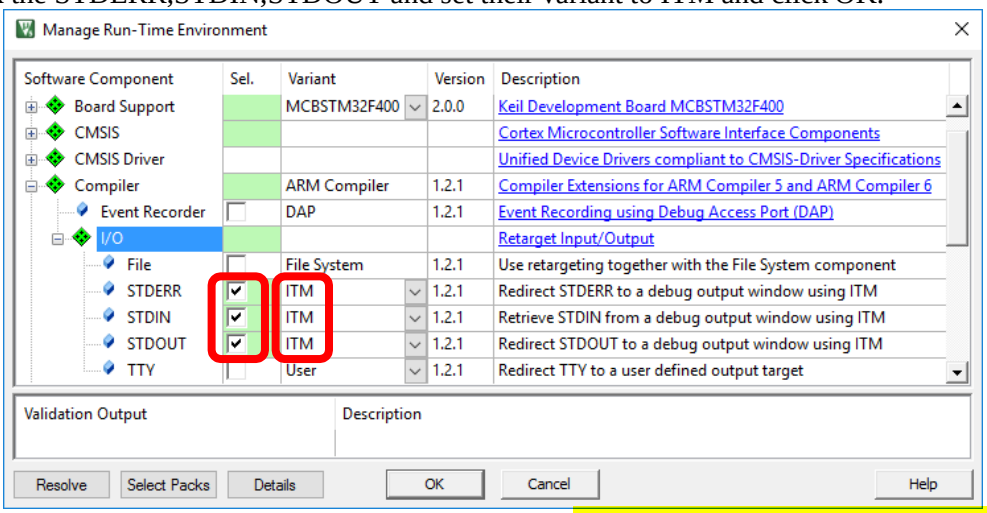
5.5 Setting Up Debug (printf) window:

Keil has several visualization and debugging tools to understand why a code is not working properly. One of them is printf window. We can print anything using printf command in our C code. We will use this window to print our measured distance. To use printf window, we need to first follow these steps(taken from µVision User's Guide (arm.com)):

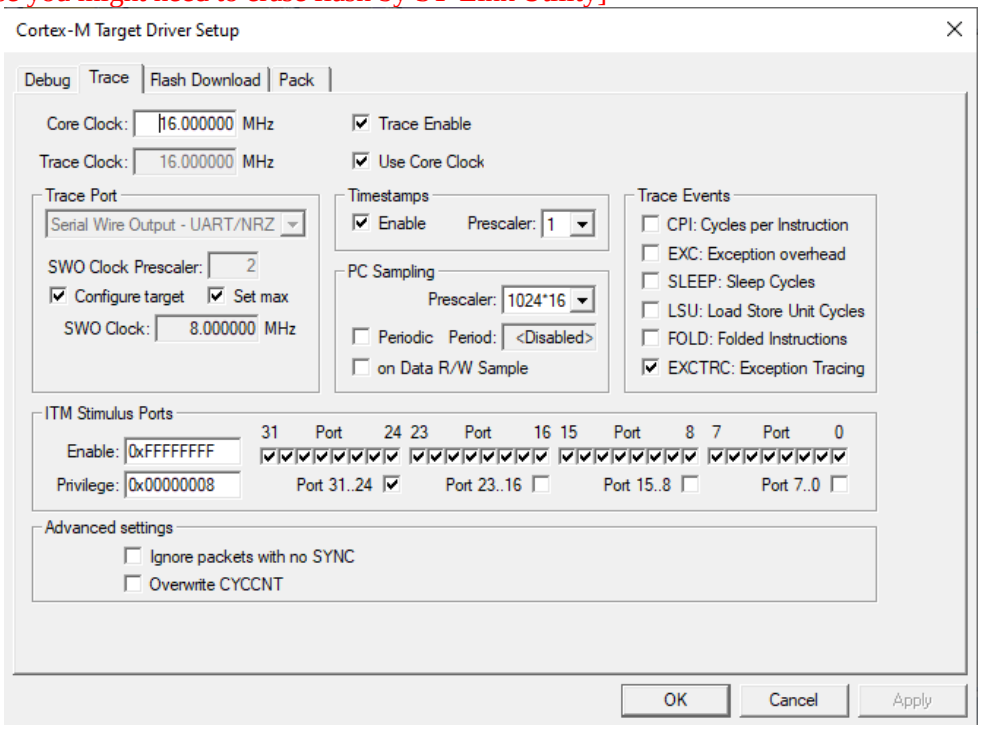
1. Open the Manage Run Time Environment.



2. Check the STDERR,STDIN,STDOUT and set their variant to ITM and click OK.



3. Go to Options for Target-->Debug-->Setting-->Trace. Check Trace Enable and set the core clock to your used system clock frequency. [Disconnect the STM board from PC before doing this. Otherwise you might need to erase flash by ST-Link Utility]

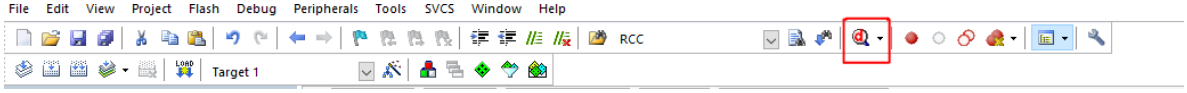


If we enable HSI, then system clock should be 16MHz. Then we have to set 16MHz in the core clock.

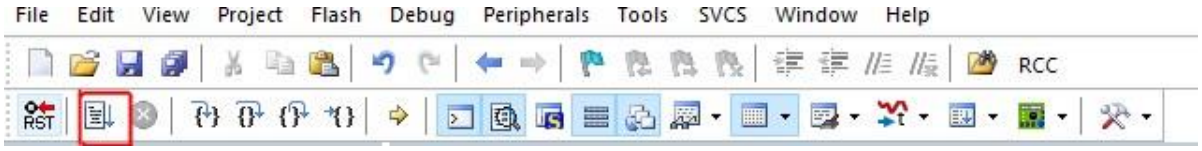
4. Add something with printf command in your C code. After completing writing your code, build and upload it.

```
printf("Hello Universe\n");//for debugging
```

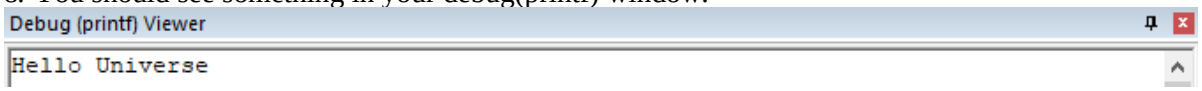
5. Now click on the start/stop debugging session icon. It will start debugging page.



6. In debugging page, from View tab on top toolbar, go to Serial Windows and select debug(printf) window. You should see a window named 'Debug(printf) Viewer' on the bottom right corner.
7. Click on run icon.



8. You should see something in your debug(printf) window.



You might see nothing in your debug(printf) window and on the bottom bar of Keil screen, you might see 'Trace:No synchronization'. If this happens, try to check if you have given the correct frequency in core clock in step 3.

5.6 Lab Exercise:

1. Build the circuit and write code for distance measurement with ultrasonic sensor. Connect the Gnd pin of the sensor before connecting the Vcc pin.
2. Upload your code to STM32 board.
3. Setup Debug(printf) window.
4. Show results.

Additional Exercise:

Interface the audio speaker with your micro controller board as you did in previous experiment. Write code in such a way that when there is no object in front of ultrasonic sensor, the speaker makes no sound. When sonar sensor detects any object, the speaker makes a sound and its frequency gradually increases as the object approaches the sonar sensor.

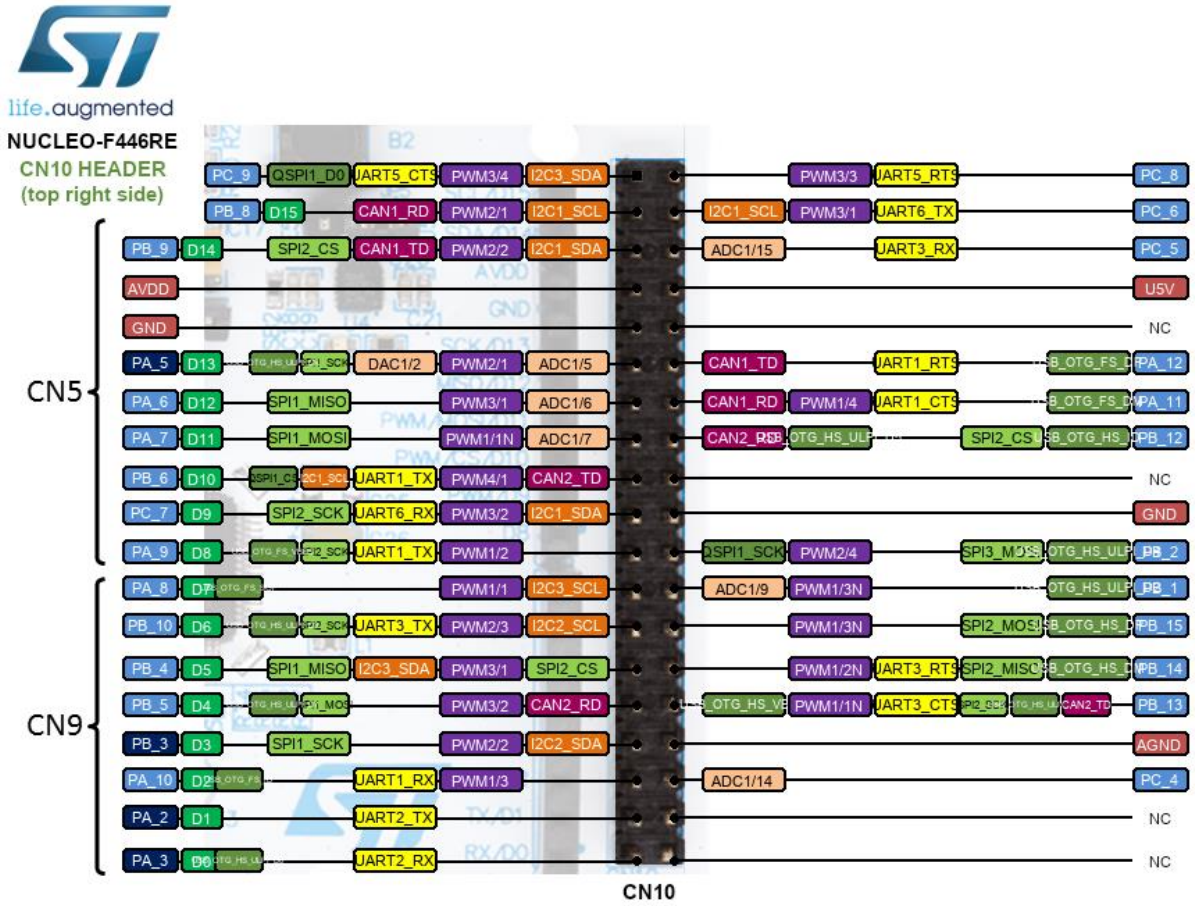
6 Code Printout

After this page, please attach printout of the main codes for each section. Use monospace fonts (Such as Noto Mono) for your codes. Use font size 8. Paste them in Microsoft Word. Give appropriate heading for each problem (Such as *Lab Exercise of Section 3*, etc).

7 Appendix A: Pin Connections on Nucleo-64 board

This diagram is common for both Nucleo-L476RG and Nucleo-L446RE

Board Component	Microcontroller Pin	Comment
Green LED	PA 5	SB42 closed and SB29 open by default
Blue user button	PC 13	Pulled up externally
Black reset button	NRST	Connect to ground to reset
ST-Link UART TX	PA 2	STLK_TX
ST-Link UART RX	PA 3	STLK_RX
ST-Link SWO/TDO	PB 3	Trace output pin/Test Data Out pin
ST-Link SWDIO/TMS	PA 13	Data I/O pin/Test Mode State pin
ST-Link SWDCLK/TCK	PA 14	Clock pin/Test Clock pin

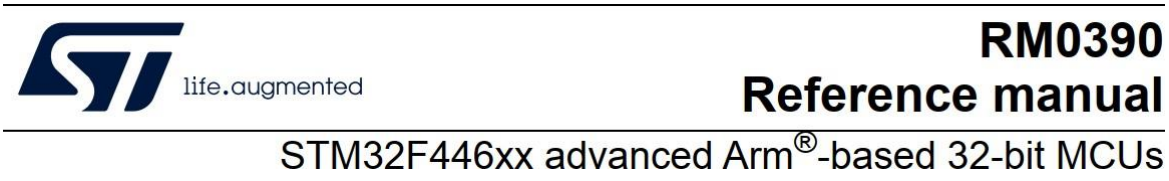


Appendix B: Acknowledgement and References

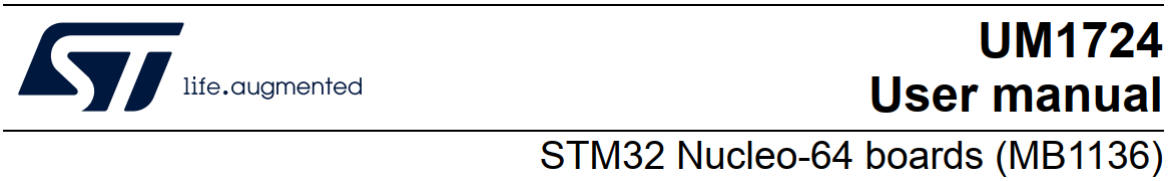
The labsheet is prepared by **Sadman Sakib Ahbab, Bejoy Sikder, Suzit Hasan Nayem, Nafis Sadik**, Dept of EEE, BUET, on 05/08/2022

The labsheet is based on Lab Materials provided as Instructor Supplement of "Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, Third Edition." By Dr. Yifeng Zhu,. The Labsheets are modified from STM32L4 architecture to STM32F4 architecture.

The following documents are strongly recommended as reference:

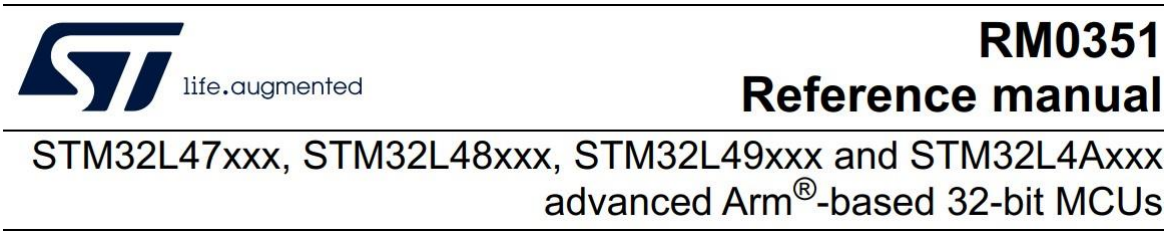


https://www.st.com/resource/en/reference_manual/rm0390-stm32f446xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf



https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

The following Document is recommended for Yifeng Zhu's book literature



https://www.st.com/resource/en/reference_manual/rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf