ELEC-240 Lab3
Basic Timer Interrupts on the STM32F429 Nucleo-144 Development Board [*]

Stuart MacVeigh, Paul Davey

October 4, 2018

1 Introduction

In previous labs you have seen how to generate a delay using 'blocking' techniques where the micro-controller simply wastes its time performing a pointless task such as incrementing a variable to a very large value. This lab exercise introduces how to use a hardware timer interrupt to achieve the same result.

All micro-controllers have at least one hardware timer which, when enabled, counts without any intervention from the program counter and can generate an interrupt when the count reaches a pre-determined value. Using interrupts is much more efficient as it allows the micro-controller to perform other tasks while the delay is being generated, or if there is no other tasks to perform it can be put into low power mode during this time thus greatly decreasing power consumption. Timer interrupts are also much more accurate which makes them ideal for discrete-time sampling DSP systems, blocking delays cannot produce such accuracy as they can be influenced by other processes on the micro-controller.

1.1 Learning Outcomes

By the end of this lab exercise you should be able to:

1. Demonstrate an understanding of hardware timer operation and the advantages over blocking delays
2. Calculate the necessary auto-reload value for a desired delay period
3. Configure the timer registers to implement the hardware timers on the micro-controller and demonstrate an understanding of each registers operation

[*]document produced by Stuart MacVeigh

2 Hardware

The hardware timer derives its clock source from the main core clock. Optimally the core clock runs at 180MHz with the PLL enabled . Assuming no software division has been applied the following is true:

PLL Status	Optimal Core Clock Speed (F_{AHB})	Timer clock Speed (F_{APB})
Enabled	180MHz	90MHz (div 2)
Disabled	16MHz	16MHz (div 1)

and the value to be loaded into the ARR register for a desired delay time is given by

$$ARR = \frac{F_{APB}}{PSC + 1} \times T_{delay} \tag{1}$$

Register details are available in Table 1.

The timer works by incrementing a binary counter on every clock pulse of the APB clock. The instantaneous value of this counter is reflected in the CNT register. When this value becomes equal to the ARR register value an Update event occurs. The counter is reset to zero and an Update Interrupt is generated if enabled in the timer configuration registers. Timers 2 & 5 have 32bit ARR registers while Timers 3 & 4 have 16bit ARR registers meaning Timers 2 & 5 can count to much higher values and hence produce longer delays for a given input clock frequency. If the desired delay requires the ARR value to exceed the maximum value which can be contained in the ARR register then the incoming clock frequency F_{APB} can be divided using the pre-scaler (PSC). Each timer has its own pre-scaler so each one can be divided by different amounts if necessary.

3 Software

There are a number of registers involved when configuring the hardware timer as it is capable of controlling many peripheral features. This exercise uses Timer 2 which is one of several general purpose hardware timers available on this micro-controller. Section 18.4 of the RM0090 Reference manual in Table 2 outlines the various registers used to configure the general purpose hardware timers.

The basic steps for enabling a timer interrupt are:

1. The timer clock must be enabled in the peripheral clock enable register
2. The update interrupt enable must be enabled in the timer DMA/Interrupt enable register
3. The prescaler and auto-reload registers must be loaded as using Equation (1) to generate the correct time delay
4. The global interrupt enable must be set in the interrupt controller register
5. The timer counter must be enabled in the timer control register

Please see Table 1 below for the register locations in the Reference Manual.

3.1 Interrupts

When any interrupt is triggered the program counter will immediately jump to the Interrupt Service Routine (ISR) associated with that interrupt source and execute the code contained therein. An interrupt flag for that interrupt is also asserted in the Status Register (SR) of the peripheral which generated the interrupt. When the code inside the ISR has been run the program counter will then return to the section of code which was being run at the moment the interrupt was invoked.

Before exiting any ISR the afore-mentioned interrupt flag must be cleared by software unless it is specifically stated in the Reference Manual that the flag is auto-cleared by hardware. Failure to clear the interrupt flag will result in no further interrupts being generated by the peripheral even though it is still running.

Register Name	**Reference Manual Section**
Hardware Clock Enable Register (AHB1ENR)	6.3.10
Peripheral Clock Enable Register (APB1ENR)	6.3.13
Interrupt Controller Vector table (NVIC)	12.3
Timer Control Register (CR1)	18.4.1
Timer DMA/Interrupt Enable Register (DIER)	18.4.4
Timer Status Register (SR)	18.4.5
Timer Count Register (CNT)	18.4.10
Timer Prescale Register (PSC)	18.4.10
Timer Auto-Reload Register (ARR)	18.4.10

Table 1: Table of relevant control registers for generating a basic hardware timer interrupt
The Reference Manual can be accessed through Table 2.

Task 1

The example code available on the DLE uses a hardware timer interrupt (using Timer 2) to blink the Green LED (PB0) on and off every 100ms. Compile and download the code to the board and confirm using the oscilloscope that the pulse width is 100ms. This can be done by probing PB0 where it is available on the header.

If you're using a Pico-Scope please refer to Section 3.2.

Task 2

Modify the code to blink the LED every:

a) 1ms
b) 1s

confirm each using the oscilloscope. There should be no blocking delays!

Task 3

Modify the code to use Timer 3 instead of Timer 2 and blink the LED every:

a) 100ms
b) 1ms
c) 1μs
d) 1s

confirm each using the oscilloscope. There should be no blocking delays!

Task 4

Modify the code to make the three LEDs blink together at different speeds. *It is not necessary to confirm this operation with the oscilloscope but the three LEDs should have visibly different blinking speeds*. There should be no blocking delays!

Task 5

1. What are the highest values that can be programmed into the ARR registers for Timer 2 and Timer 3 respectively?

2. If the micro-controller has its PLL enabled and no pre-scale set (PSC=0) what is:

   a) The clock speed of the micro-controller?

   b) The time delay that will be produced by Timer 2 and Timer 3 respectively when their ARR registers are loaded with the highest possible value?

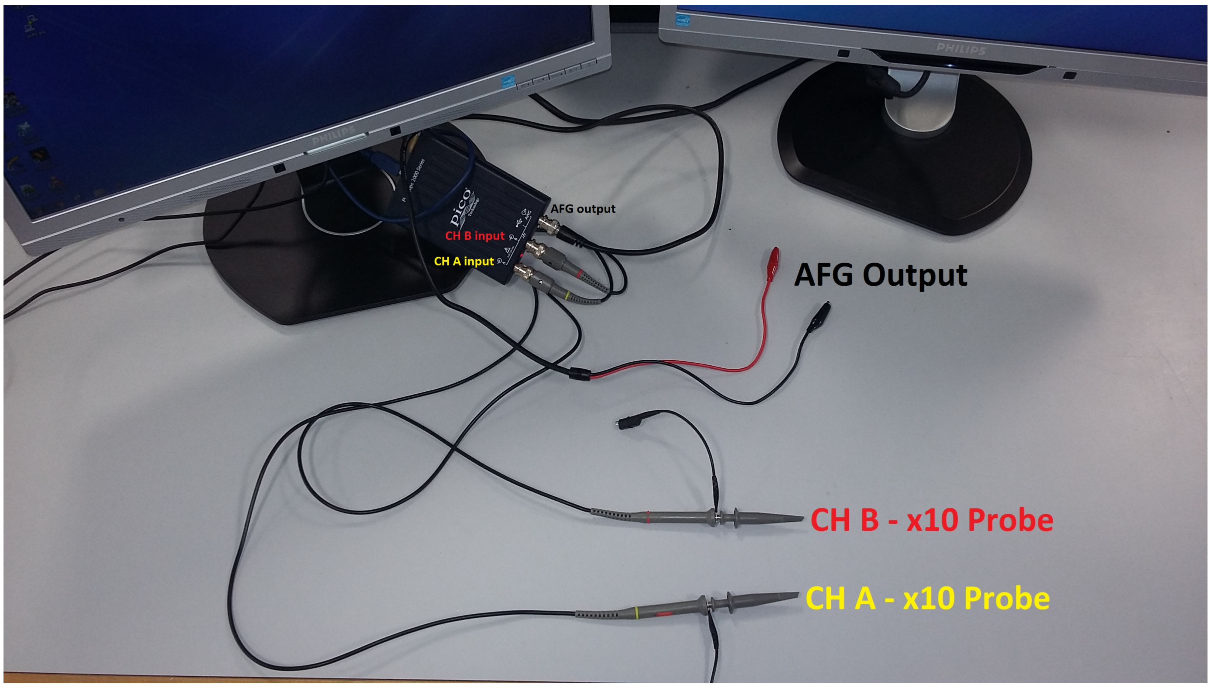
3. If the micro-controller has its PLL enabled and the Timer PSC registers are 16bit what is:

   a) The largest value that can be programmed into the PSC register?

   b) The longest time delay achievable by Timer 2 and Timer 3 respectively?

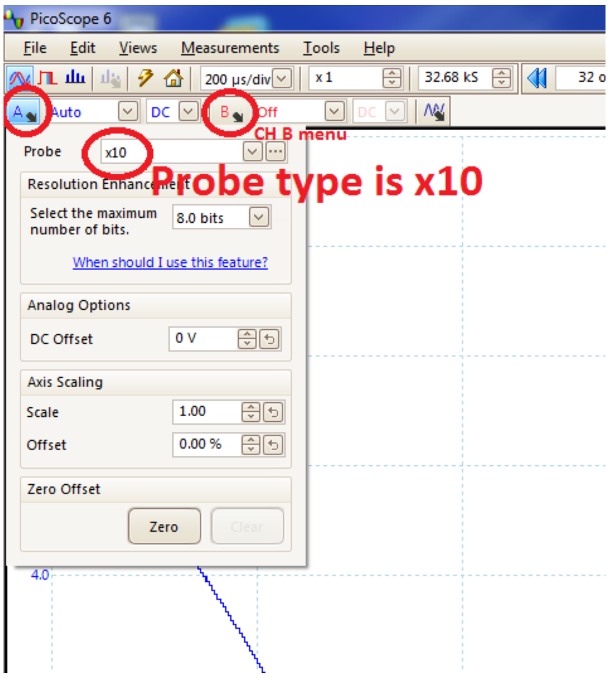
4. What will happen if the interrupt flag is **not** cleared before exiting the ISR?

3.2 Pico-Scope for PC



1. Open the Pico-Scope application by clicking the icon on the desktop.

2. The channels are configured here



Probe: x10

Resolution Enhancement

Select the maximum number of bits.: 8.0 bits

Analog Options

DC Offset: 0 V

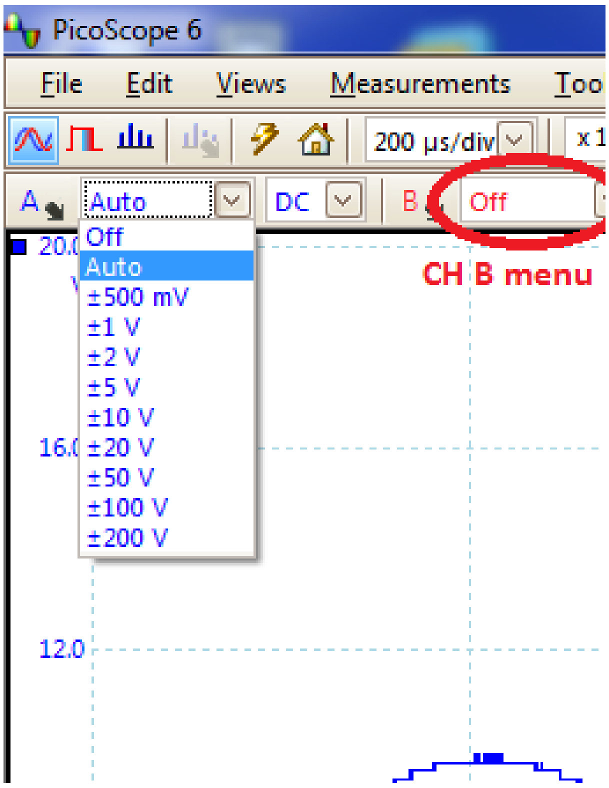
Axis Scaling

Scale: 1.00

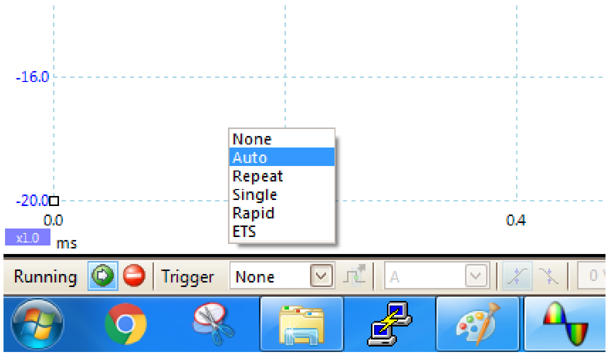
Offset: 0.00 %

Zero Offset

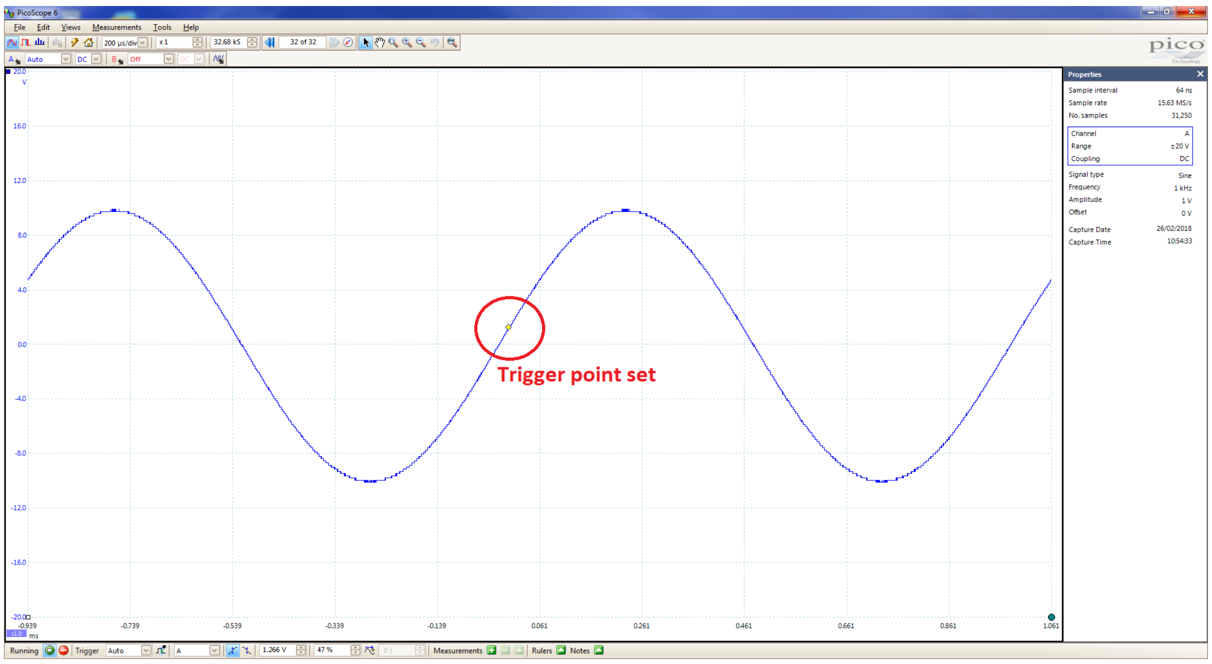
3. The voltage range is configured here



4. The trigger type is enabled here



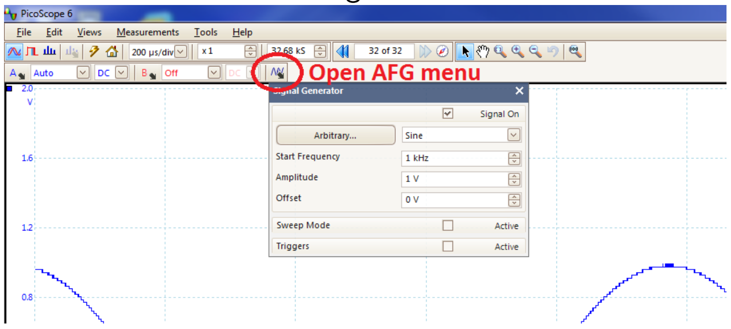
5. Drag the diamond to set the trigger point



Sample interval	64 ns
Sample rate	15.63 MS/s
No. samples	31,250
Channel	A
Range	±20 V
Coupling	DC
Signal type	Sine
Frequency	1 kHz
Amplitude	1 V
Offset	0 V
Capture Date	26/02/2018
Capture Time	10:54:33

3.2.1 Pico-Scope AFG

The AFG is enabled and configured here



Signal Generator
[✓] Signal On
Sine
Start Frequency: 1 kHz
Amplitude: 1 V
Offset: 0 V
Sweep Mode: ☐ Active
Triggers: ☐ Active

4 Support Documentation

Document Name	**Contained Information**
UM1974 User manual	• Pin identification and the supported special functions • Circuit schematics • Jumper and component identification • Header pinouts
RM0090 Reference manual	• MCU memory and peripherals architecture • Peripheral control registers, addresses and bit-fields

Table 2: Table of relevant support documentation for Nucleo-144 development boards (The document names are hyperlinks, please click on them to access the documents)