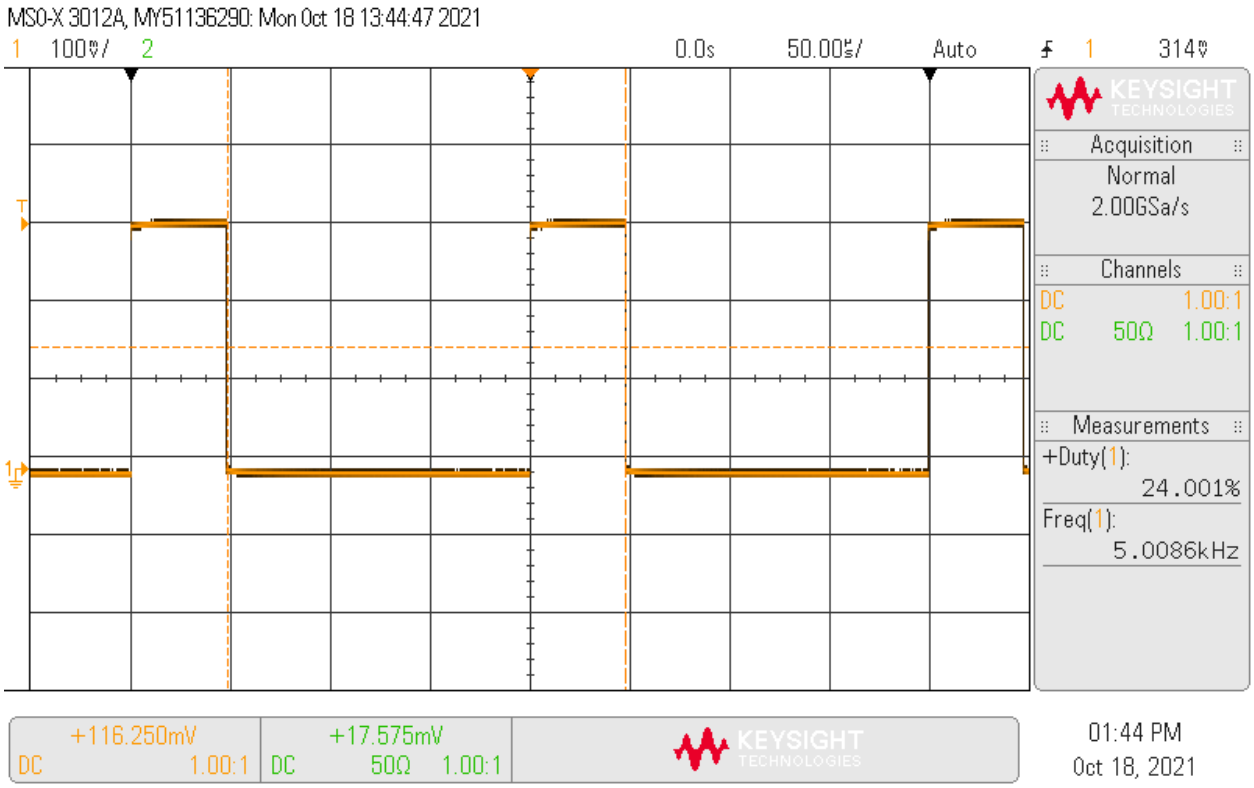
A4- Interrupts and Timers

Part A:

4MHz /5 khz = 800 counts
25% of 800 = 200 counts for high
75% of 800 = 600 counts for low




```
#include "interrupt.h"
#define FIVEKHZ 800
#define TWENTYFIVEPERCENT 600

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);

int main(void)
{
	/* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
	HAL_Init();
```

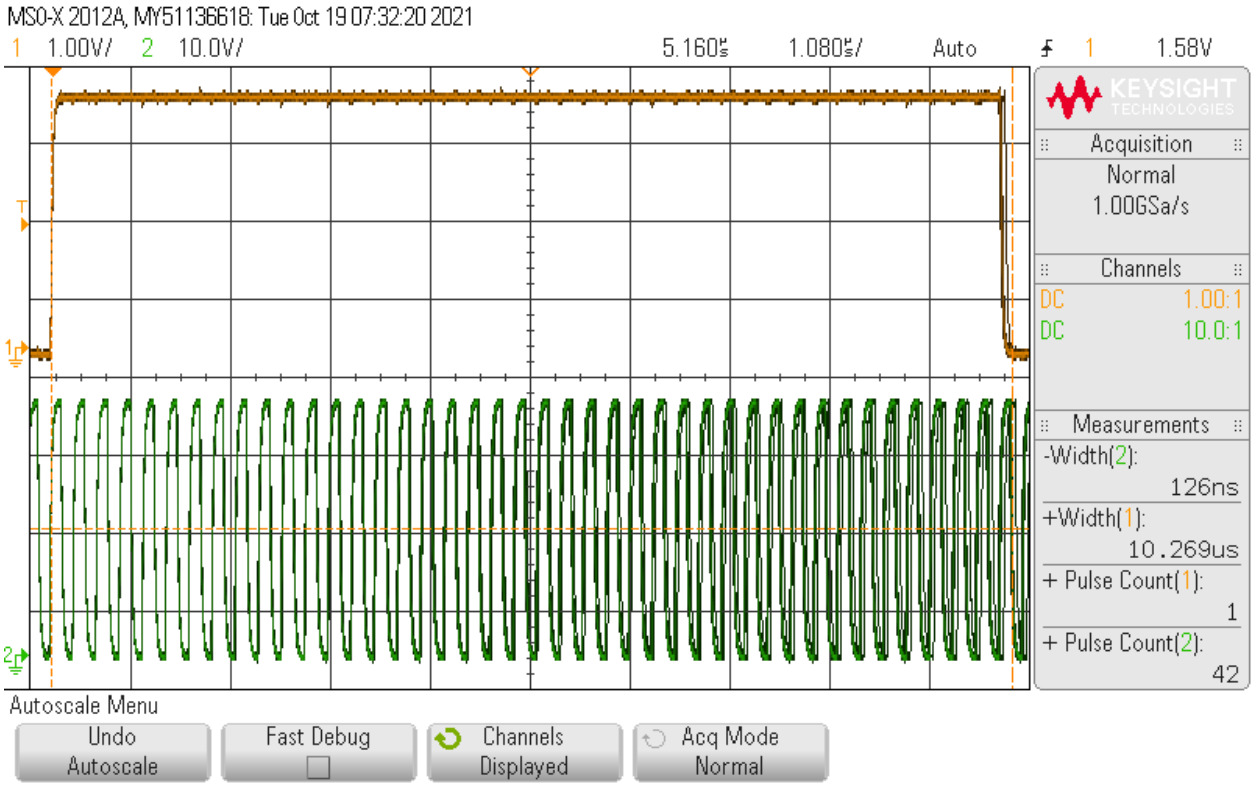
```
	/* Configure the system clock */
	SystemClock_Config();

	// Configure onboard LED (PA5)
//////////////////////////////////////////
	// enable GPIOA
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);
	GPIOA->MODER &= ~(GPIO_MODER_MODE5);                    // enable GPIO
output mode
	GPIOA->MODER |= (1 << GPIO_MODER_MODE5_Pos);
	GPIOA->OTYPER &= ~(GPIO_OTYPER_OT5);                    // set
push-pull output
	GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD5);                    // disable
pull up/pull down resistor
	GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED5);      // slow speed
	GPIOA->ODR &= ~(GPIO_ODR_OD5);                              // turn
LED off

	// Configure Timer TIM2
////////////////////////////////////////////////
	// enable TIM2 clock
	RCC->APB1ENR1 |= (RCC_APB1ENR1_TIM2EN);

	// enable interrupts on UEV (update event)
	TIM2->DIER |= (TIM_DIER_UIE);

	//enable interrupts on CCI
	TIM2->DIER |= (TIM_DIER_CC1IE);

	//enable the CC1
	TIM2->CCER |= (TIM_CCER_CC1E);

	// clear interrupt flag
	TIM2->SR &= ~(TIM_SR_UIF);

	// set auto reload register
	TIM2->ARR = FIVEKHZ - 1;      // 2x10^6 - 1 (0.5s)

	// set
	TIM2->CCR1 = TWENTYFIVEPERCENT - 1;
	// start timer
	TIM2->CR1 |= (TIM_CR1_CEN);
```

```
	// enable TIM2 ISR in NVIC
	NVIC->ISER[0] = (1 << (TIM2_IRQn & 0x1F));

	// enable interrupts globaly
	__enable_irq();


	while (1) // do nothing in main after setting up TIM2
	{
	}
}

// ISR name matches NVIC (replace n with Handler)
void TIM2_IRQHandler(void) {
	// check if update event flag is set & CC1IE
	if (TIM2->SR & TIM_SR_UIF || TIM2->SR & TIM_SR_CC1IF)
	{
		// toggle LED
		GPIOA->ODR ^= GPIO_PIN_5;
		// clear both interrupt flags
		TIM2->SR &= ~(TIM_SR_UIF);
		TIM2->SR &= ~(TIM_SR_CC1IF);
	}
}

void SystemClock_Config(void)
{
	RCC_OscInitTypeDef RCC_OscInitStruct = {0};
	RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

	/** Configure the main internal regulator output voltage
	 */
	if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
	{
		Error_Handler();
	}
	/** Initializes the RCC Oscillators according to the specified
parameters
	 * in the RCC_OscInitTypeDef structure.
	 */
	RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
	RCC_OscInitStruct.MSIState = RCC_MSI_ON;
```

```
	RCC_OscInitStruct.MSICalibrationValue = 0;
	RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
	RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
	if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
	{
		Error_Handler();
	}
	/** Initializes the CPU, AHB and APB buses clocks
	 */
	RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
			|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
	RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
	RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
	RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
	RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

	if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) !=
HAL_OK)
	{
		Error_Handler();
	}
}

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
	/* USER CODE BEGIN Error_Handler_Debug */
	/* User can add his own implementation to report the HAL error return
state */
	__disable_irq();
	while (1)
	{
	}
	/* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
```

```
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
      /* USER CODE BEGIN 6 */
      /* User can add his own implementation to report the file name and
line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
      /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/
```

Part B:



40 Pulses on pin A8 while B5 is high

```c
#include "interrupt.h"
#define FIFTYPERCENT 400

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);

int main(void)
{
	/* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
	HAL_Init();

	/* Configure the system clock */
	SystemClock_Config();

	// Configure onboard LED (PA5)
//////////////////////////////////////////
	// enable GPIOA
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);
	GPIOA->MODER &= ~(GPIO_MODER_MODE5);                    // enable GPIO
output mode
	GPIOA->MODER |= (1 << GPIO_MODER_MODE5_Pos);
	GPIOA->OTYPER &= ~(GPIO_OTYPER_OT5);                    // set
push-pull output
	GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD5);                    // disable
pull up/pull down resistor
	GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED5);      // slow speed
	GPIOA->ODR &= ~(GPIO_ODR_OD5);                                // turn
LED off

	// Enable MCO, select MSI (4 MHz source)
	RCC->CFGR = ((RCC->CFGR & ~(RCC_CFGR_MCOSEL)) | (RCC_CFGR_MCOSEL_0));

	// Configure MCO output on PA8
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);
	GPIOA->MODER &= ~(GPIO_MODER_MODE8);      // alternate function mode
	GPIOA->MODER |= (GPIO_MODER_MODE8_1);
	GPIOA->OTYPER &= ~(GPIO_OTYPER_OT8);      // Push-pull output
	GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD8);      // no resistor
	GPIOA->OSPEEDR |= (GPIO_OSPEEDR_OSPEED8); // high speed
	GPIOA->AFR[1] &= ~(GPIO_AFRH_AFSEL8);     // select MCO function
```

```
	// Configure Timer TIM2
////////////////////////////////////////////////
	// enable TIM2 clock
	RCC->APB1ENR1 |= (RCC_APB1ENR1_TIM2EN);

	// enable interrupts on UEV (update event)
	TIM2->DIER |= (TIM_DIER_UIE);

	//enable interrupts on CCI
	TIM2->DIER |= (TIM_DIER_CC1IE);

	//enable the CC1
	TIM2->CCER |= (TIM_CCER_CC1E);

	// clear interrupt flag
	TIM2->SR &= ~(TIM_SR_UIF);

	// set auto reload register
	TIM2->ARR = 0xFFFFFFFF;       // 2x10^6 - 1 (0.5s)

	// set
	TIM2->CCR1 = FIFTYPERCENT-1;

	// start timer
	TIM2->CR1 |= (TIM_CR1_CEN);

	// enable TIM2 ISR in NVIC
	NVIC->ISER[0] = (1 << (TIM2_IRQn & 0x1F));

	// enable interrupts globaly
	__enable_irq();


	while (1) // do nothing in main after setting up TIM2
	{
	}
}

// ISR name matches NVIC (replace n with Handler)
void TIM2_IRQHandler(void) {
	// check if update event flag is set & CC1IE
	 GPIOA->ODR |= GPIO_PIN_8;
```

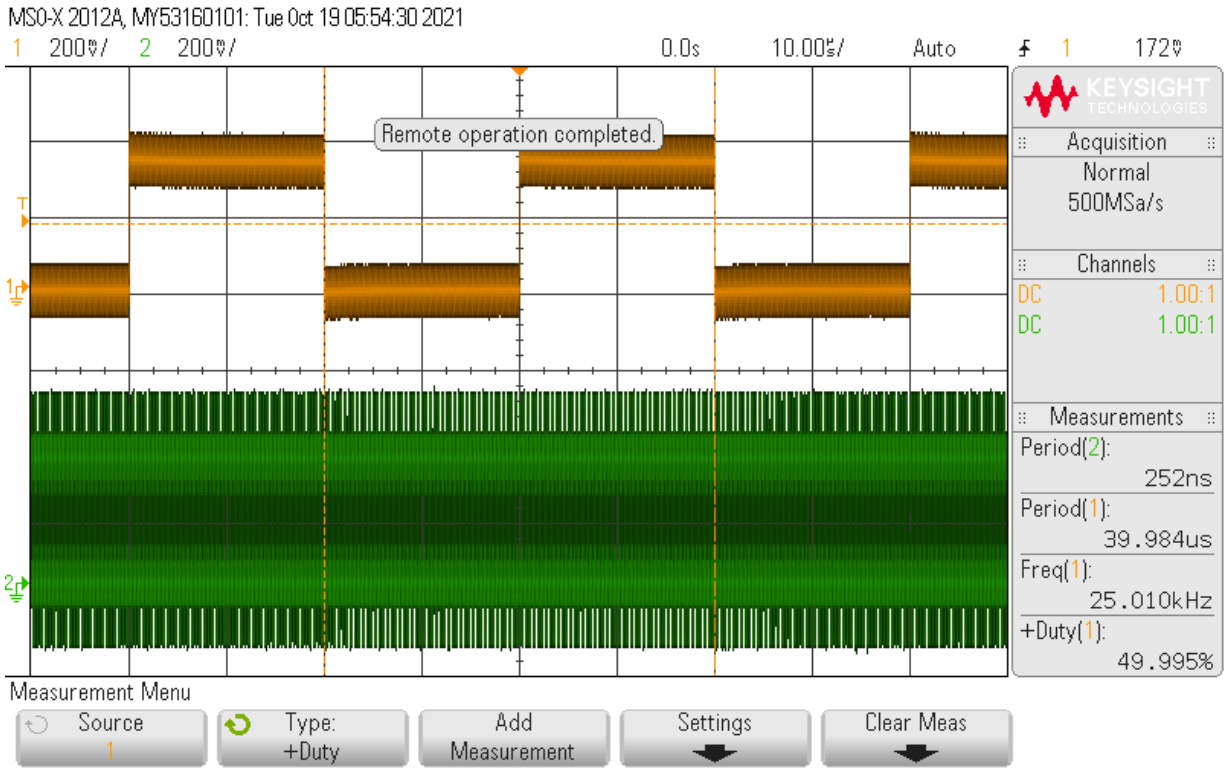
```
	if (TIM2->SR & TIM_SR_UIF || TIM2->SR & TIM_SR_CC1IF)
	{
		TIM2->CCR1 += FIFTYPERCENT;
		// toggle LED
		GPIOA->ODR ^= GPIO_PIN_5;
		// clear both interrupt flags
		TIM2->SR &= ~(TIM_SR_UIF);
		TIM2->SR &= ~(TIM_SR_CC1IF);
	}
	GPIOA->ODR &= ~GPIO_PIN_8;
}

void SystemClock_Config(void)
{
	RCC_OscInitTypeDef RCC_OscInitStruct = {0};
	RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

	/** Configure the main internal regulator output voltage
	 */
	if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
	{
		Error_Handler();
	}
	/** Initializes the RCC Oscillators according to the specified
parameters
	 * in the RCC_OscInitTypeDef structure.
	 */
	RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
	RCC_OscInitStruct.MSIState = RCC_MSI_ON;
	RCC_OscInitStruct.MSICalibrationValue = 0;
	RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
	RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
	if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
	{
		Error_Handler();
	}
	/** Initializes the CPU, AHB and APB buses clocks
	 */
	RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
			|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
	RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
	RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
```

```
      RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
      RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

      if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) !=
HAL_OK)
      {
            Error_Handler();
      }
}

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
      /* USER CODE BEGIN Error_Handler_Debug */
      /* User can add his own implementation to report the HAL error return
state */
      __disable_irq();
      while (1)
      {
      }
      /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
      /* USER CODE BEGIN 6 */
      /* User can add his own implementation to report the file name and
line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
      /* USER CODE END 6 */
}
```

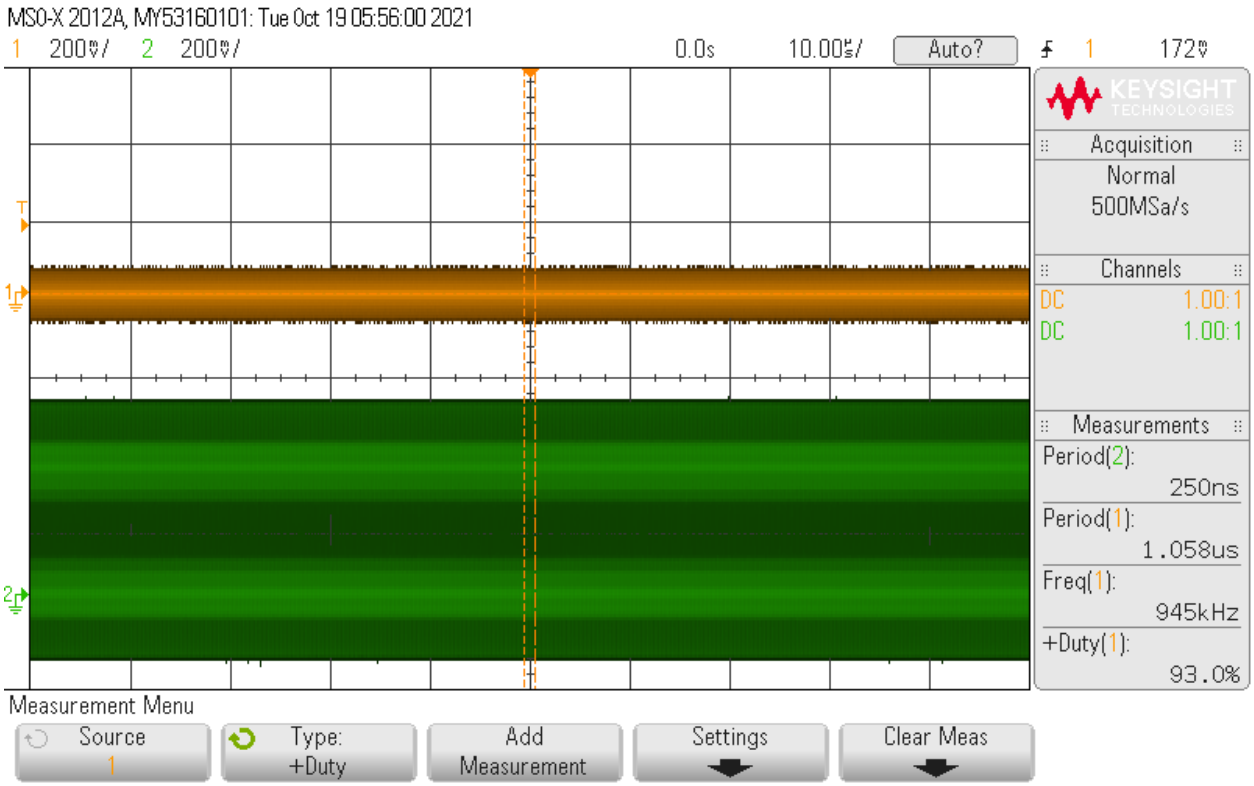
```
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/
```

1. Count how many MCO clock cycles it takes to execute your ISR
   40 clock cycles

Part C



CCR1 = 80



CCR1 = 79

No it does not correlate because the frequency of the clock does not change, but the frequency of CCR1 does change.