Hi,

I have initialized the TIM2 for a 1us overflow interrupt. The clock of the timer is 60MHz and by initializing the counter period as "60 -1", we can fix the timer for 60MHz/60 = 1MHz. It means if we toggle a pin inside the interrupt routine, it should generate a 500KHz pulse.

But in practice, the speed does not get increase more than 178KHz. as we know the speed limit of the GPIOs are much higher than these levels even in the worst case (Picture 1)

This is not the Timer problem, because when I change the 60-1 to 600 - 1, it means 60MHz/600 = 100KHz, or 50KHz GPIO toggling frequency which is correct and confirmed by the oscilloscope (Picture 2)

That's a simple code and situation but this result is weird

Picture 1

Picture 2

```
**************************************************************************
*/
/* Includes ------------------------------------------------------------
*/
#include "main.h"
#include "stm32f4xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------
*/
TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------
*/

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------
*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------
*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

```
/**
* @brief The application entry point.
*
* @retval None
*/
int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration--------------------------------------------------------
*/

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

/* USER CODE BEGIN Init */

/* USER CODE END Init */

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

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1) {

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

HAL_Delay(5);

}
/* USER CODE END 3 */

}

/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
```

```
RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;

/**Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLM = 16;
RCC_OscInitStruct.PLL.PLLN = 120;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 4;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure the Systick interrupt time
*/
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the Systick
*/
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* TIM2 init function */
static void MX_TIM2_Init(void)
{
```

```
TIM_ClockConfigTypeDef sClockSourceConfig;
TIM_MasterConfigTypeDef sMasterConfig;

htim2.Instance = TIM2;
htim2.Init.Prescaler = 0;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 60 - 1;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/** Configure pins as
* Analog
* Input
* Output
* EVENT_OUT
* EXTI
*/
static void MX_GPIO_Init(void)
{

GPIO_InitTypeDef GPIO_InitStruct;

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOE_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, GPIO_PIN_SET);

/*Configure GPIO pin : PE8 */
GPIO_InitStruct.Pin = GPIO_PIN_8;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

}
```

```
/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @param file: The file name as string.
* @param line: The line in file as a number.
* @retval None
*/
void _Error_Handler(char *file, int line)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the HAL error return state */
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,
tex: 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****/
```

Interrupt routine

```
void TIM2_IRQHandler(void)
{
/* USER CODE BEGIN TIM2_IRQn 0 */

/* USER CODE END TIM2_IRQn 0 */
HAL_TIM_IRQHandler(&htim2);
/* USER CODE BEGIN TIM2_IRQn 1 */
```

```
HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);

/* USER CODE END TIM2_IRQn 1 */
}
```