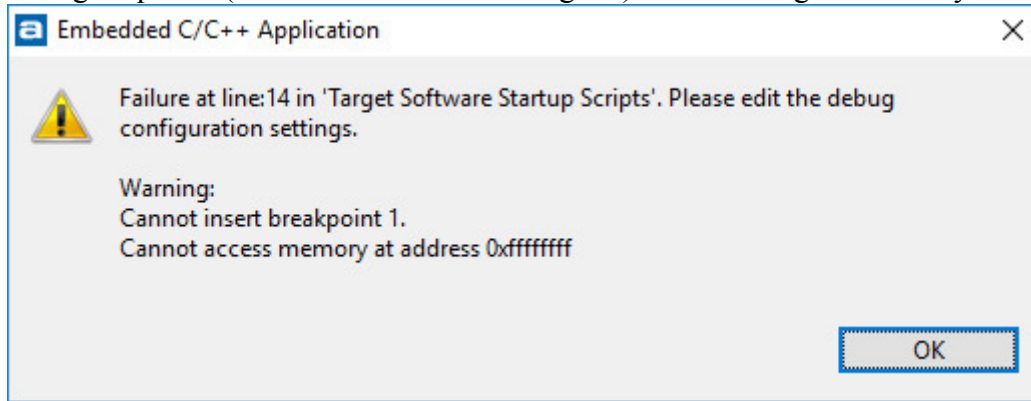


My programm occured error at the begining. You can see this at the screenshot. When i run code there is nothig on pin B1(there should be PWM singnal ). I would be gratfull if anyone could help.



```
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_SPI2_Init();
    MX_USART1_UART_Init();
    MX_TIM3_Init();
    MX_CRC_Init();
    MX_RTC_Init();

    /* Initialize interrupts */
    MX_NVIC_Init();

    /* USER CODE BEGIN 2 */
    HAL_SPI_Init( &hspi2 );

    xTaskCreate( StartDefaultTask,(const char*) "defaultTask",1024,NULL,1,NULL);

    vTaskStartScheduler();

    while (1)
    {
```

```

}
/* USER CODE END 3 */

}

/** System Clock Configuration
*/
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    RCC_PeriphCLKInitTypeDef PeriphClkInit;

    /**Initializes the CPU, AHB and APB busses clocks
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_LSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = 16;
    RCC_OscInitStruct.LSIState = RCC_LSI_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    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_HSI;
    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(__FILE__, __LINE__);
    }

    PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;
    PeriphClkInit.RTCClockSelection = RCC_RTCCCLKSOURCE_LSI;
    if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != 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, 15, 0);
}

```

```
/** NVIC Configuration
*/
static void MX_NVIC_Init(void)
{
    /* EXTI9_5_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(EXTI9_5_IRQn, 6, 0);
    HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
}

/* CRC init function */
static void MX_CRC_Init(void)
{
    hcrc.Instance = CRC;
    if (HAL_CRC_Init(&hcrc) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }
}

/* RTC init function */
static void MX_RTC_Init(void)
{
    /**Initialize RTC Only
    */
    hrtc.Instance = RTC;
    hrtc.Init.AsynchPrediv = RTC_AUTO_1_SECOND;
    hrtc.Init.OutPut = RTC_OUTPUTSOURCE_ALARM;
    if (HAL_RTC_Init(&hrtc) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }
}

/* SPI2 init function */
static void MX_SPI2_Init(void)
{
    /* SPI2 parameter configuration*/
    hspi2.Instance = SPI2;
    hspi2.Init.Mode = SPI_MODE_MASTER;
    hspi2.Init.Direction = SPI_DIRECTION_1LINE;
    hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
    hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
    hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
    hspi2.Init.NSS = SPI_NSS_SOFT;
    hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
    hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
    hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
    hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    hspi2.Init.CRCPolynomial = 10;
    if (HAL_SPI_Init(&hspi2) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }
}
```

```
}

/* TIM3 init function */
static void MX_TIM3_Init(void)
{

    TIM_MasterConfigTypeDef sMasterConfig;
    TIM_OC_InitTypeDef sConfigOC;

    htim3.Instance = TIM3;
    htim3.Init.Prescaler = 0;
    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim3.Init.Period = 531;
    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 199;
    sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

    HAL_TIM_MspPostInit(&htim3);

}

/* USART1 init function */
static void MX_USART1_UART_Init(void)
{

    huart1.Instance = USART1;
    huart1.Init.BaudRate = 9600;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

}

}
```

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

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);

    /*Configure GPIO pins : PA1 PA4 */
    GPIO_InitStructure.Pin = GPIO_PIN_1|GPIO_PIN_4;
    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*Configure GPIO pins : PA5 PA6 */
    GPIO_InitStructure.Pin = GPIO_PIN_5|GPIO_PIN_6;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*Configure GPIO pin : PB12 */
    GPIO_InitStructure.Pin = GPIO_PIN_12;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*Configure GPIO pin : PA8 */
    GPIO_InitStructure.Pin = GPIO_PIN_8;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_MEDIUM;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*Configure GPIO pin : PB6 */
    GPIO_InitStructure.Pin = GPIO_PIN_6;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

```

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/* StartDefaultTask function */
void StartDefaultTask(void const * argument)
{
    /* USER CODE BEGIN 5 */
    HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_4);
    /* Infinite loop */
    for(;;)
    {

    }
    /* USER CODE END 5 */
}

/**
 * @brief This function is executed in case of error occurrence.
 * @param None
 * @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,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */

}

#endif

```