Input Capture DMA interrupt

Variables:

```
/* USER CODE BEGIN PV */
volatile uint32_t ic_dma_data[2];
volatile uint32_t last_rising_edge = 0;
volatile float frequency = 0;
volatile float duty_cycle = 0;
/* USER CODE END PV */
```

Callback Function:

```
/* USER CODE BEGIN 0 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
 if (htim->Instance == TIM2)
 {
   if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) // Rising edge
   {
     uint32_t rise_now = ic_dma_data[0];
     uint32_t period;
     if (rise_now >= last_rising_edge)
       period = rise_now - last_rising_edge;
     else
       period = (0xFFFFFFFF - last_rising_edge + 1) + rise_now;
     last_rising_edge = rise_now;
     float timer_clk = 64000000.0f;  // Assuming 64MHz
     frequency = timer_clk / period;
     // Re-arm DMA
     HAL_TIM_IC_Start_DMA(htim, TIM_CHANNEL_1,
(uint32_t*)&ic_dma_data[0], 1);
   }
   if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2) // Falling edge
   {
     uint32_t fall = ic_dma_data[1];
     uint32_t rise = last_rising_edge;
     uint32_t high_time;
```

```
    if (fall >= rise)
      high_time = fall - rise;
    else
      high_time = (0xFFFFFFFF - rise + 1) + fall;
    float period = 64000000.0f / frequency;
    duty_cycle = ((float)high_time / period) * 100.0f;
    // Re-arm DMA
    HAL_TIM_IC_Start_DMA(htim, TIM_CHANNEL_2,
(uint32_t*)&ic_dma_data[1], 1);
   }
 }
}
/* USER CODE END 0 */
```

In Main File:

```
 /* USER CODE BEGIN 2 */
 TIM14->CCR1 = 100;
 HAL_TIM_PWM_Start(&htim14, TIM_CHANNEL_1);  // Generate PWM
 HAL_TIM_IC_Start_DMA (&htim2, TIM_CHANNEL_1,
(uint32_t*)&ic_dma_data[0], 1);  // Rising edge
 HAL_TIM_IC_Start_DMA (&htim2, TIM_CHANNEL_2,
(uint32_t*)&ic_dma_data[1], 1);  // Falling edge
 /* USER CODE END 2 */

 while (1)
 {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */

        HAL_Delay(500);
 }
```