# Lösung Vorlesungbegleitende Übungsaufgaben

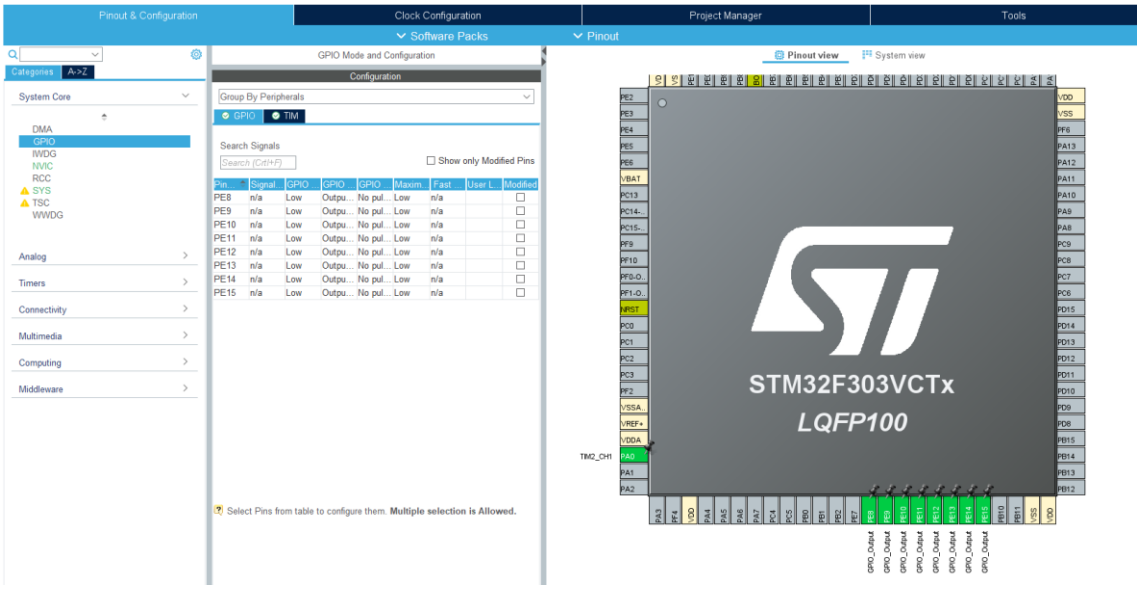
# -Mikrocomputertechnik-

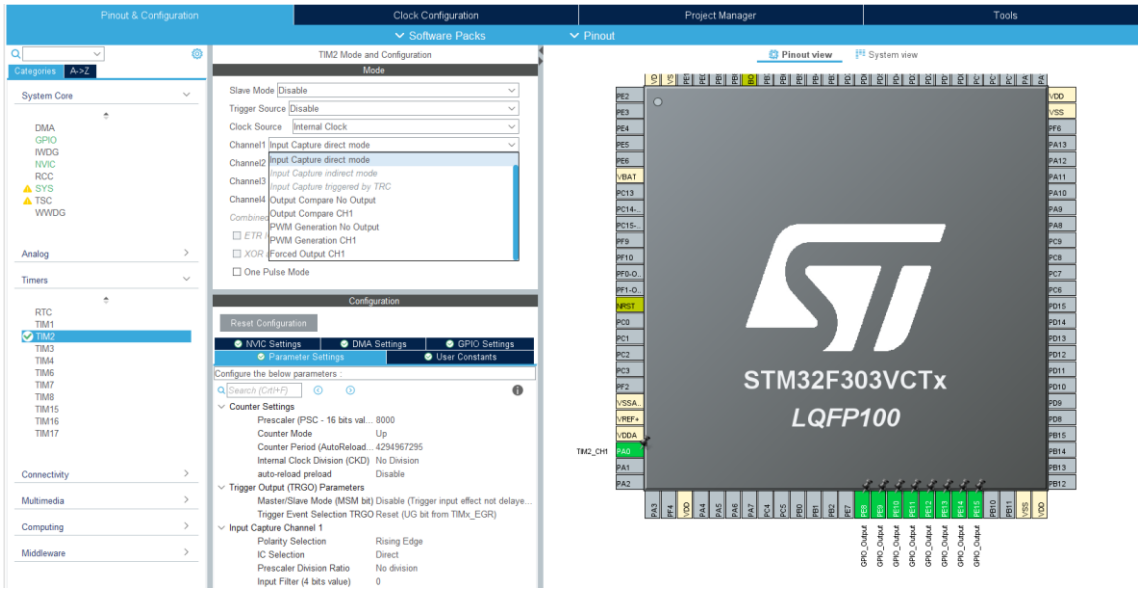
## Aufgabe 4: Input Capture Taster

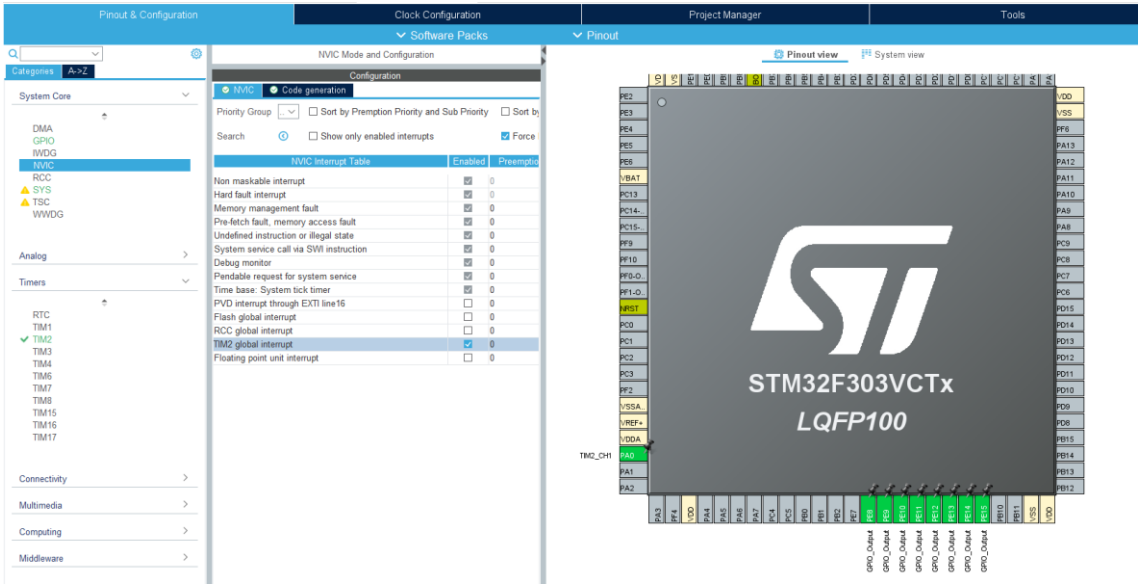
Schreiben Sie ein Programm, dass die Zeit zwischen zwei Betätigungen des USER Tasters misst. Die Messgenauigkeit soll dabei in etwa 1ms betragen.

Bei der ersten Betätigung sollen alle LEDs erlöschen. Bei der zweiten Betätigung soll für jede volle Sekunde eine LED eingeschaltet werden.

## Lösung:






```
volatile int CaptureTime_ms=0;

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
        static int CaptureFlag=0;
        static int FirstCapture=0;
        static int SecondCapture=0;
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
                if(CaptureFlag==0)
                {
                        FirstCapture=HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
                        CaptureTime_ms=0;
                        CaptureFlag=1;
                }
                else
                {
                        SecondCapture=HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
                        int CaptureDifference = SecondCapture-FirstCapture;
                        CaptureTime_ms =
(float)8000/HAL_RCC_GetPCLK1Freq()*CaptureDifference*1000;
                        CaptureFlag=0;
                }
        }
```

```
}

int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_TIM2_Init();
  HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);

  while (1)
  {
          if(CaptureTime_ms==0)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11|
                            GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
          }
          if(CaptureTime_ms>1000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>2000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>3000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_10, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>4000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_11, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>5000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_12, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>6000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_13, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>7000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_14, GPIO_PIN_SET);
          }
          if(CaptureTime_ms>8000)
          {
                  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_15, GPIO_PIN_SET);
          }
  }
}


static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

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

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);

  /*Configure GPIO pins : PE8 PE9 PE10 PE11 PE12 PE13 PE14 PE15 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
```

```
}

static void MX_TIM2_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 8000;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4294967295;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
}
```