دانشکده مهندسی برق

KASRA KHALAFI: 9523038

Home Work 3

Microprocessors 1

Dr.Saeed Sharifian Khortoomi

در این سری از تمرین، به پیاده سازی موسیقی با کمک تایمر و وقفه ها پرداختیم.

از ما خواسته شده بود تا آهنگ Twinkle Twinkle Little Stars را پیاده سازی کنیم. بدین ترتیب که یک موج سینوسی با موج یاد شده ساخته و به کمک این موج و فرکانس مشخص، آهنگ یاد شده را اجرا کنیم.

نت موسیقیایی این آهنگ به صورت زیر می باشد:

هر وقفه بین نت ها در این آهنگ به مدت نیم یا یک ثانیه می باشد. در نتیجه به دو وقفه نیاز خواهیم داشت. یکی برای درست کردن مدت زمان آهنگ که از تایمر 2 استفاده شده است، تایمر دیگر نیز برای تولید موج سینوسی با فرکانس مد نظر.

برای این کار از یک موج سینوسی 100 عدد سمپل و نمونه برداری انجام میگیرد. که این 100 مقدار به صورت زیر می باشند:

```
int sinewave[100] = {2048,2176,2304,2431,2557,2680,2801,2919,3034,3145,3251,3353,3449,3540,3625,3704,3776,3842,3900,3951,
  3995,4031,4059,4079,4091,4095,4091,4079,4059,4031,3995,3951,3900,3842,3776,3704,3625,3540,3449,3353,3251,3145,3034,2919,
  2801,2680,2557,2431,2304,2176,2048,1919,1791,1664,1538,1415,1294,1176,1061,950,844,742,646,555,470,391,319,253,195,144,
  100,64,36,16,4,0,4,16,36,64,100,144,195,253,319,391,470,555,646,742,844,950,1061,1176,1294,1415,1538,1664,1791,1919};
```

همچنین فرکانس موج های سینوسی را مطابق فرکانس یاد شده مقدار دهی میکنیم. مقدار زمان اجرای هر نت موسیقی نیز همانطور که یاد شد، نیم یا یک ثانیه می باشد که پیاده سازی گشت. سپس یک i برای مقدار سمپل های نمونه برداری شده و یک state برای چک کردن اینکه نیم ثانیه یا یک ثانیه تمام شده است یا خیر پیاده سازی شد.

```
int A4 =440, C4 = 262, D4 = 294, E4 = 330, F4 = 349, G4 = 277;
int Q_Duration = 500;
int H_Duration = 1000;
int i = 0;
int state = 1;
```

برای این کار GPIO را در ابتدای کار فعال میکنیم. سپس باید DAC به همراه دو تایمر یاد شده همانند شکل زیر راه اندازی شوند. سپس DAC را شروع میکنیم. داخل حلقه ی While نیز به پیاده سازی نت ها به ترتیب میکنیم بدین صورت که تابع play_note دو ورودی یکی نوت و دیگری مدت زمان اجرای این نوت را میگیرد و به پیاده سازی به کمک ورودی ها می پردازد.

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DAC_Init();
MX_TIM2_Init();
MX_TIM3_Init();
/* USER CODE BEGIN 2 */
HAL_DAC_Start(&hdac, DAC1_CHANNEL_1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */
  play_note(C4, Q_Duration);
  play_note(C4, Q_Duration);
  play_note(G4, Q_Duration);
  play_note(G4, Q_Duration);
  play_note(A4, Q_Duration);
  play_note(A4, Q_Duration);
  play_note(G4, H_Duration);
  play_note(F4, Q_Duration);
  play_note(C4, Q_Duration);
```

در داخل تابع play_note، ابتدا چک میشود که فرکانسی ورودی جه مقداری می باشد و پس از مشخص شدن مقدار فرکانس ورودی، وارد یکی از حلقه های یاد شده می شود.

قانون آپدیت برای فرکانس به صورت زیر می باشد:

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler + 1)(Period + 1)(RepetitionCounter + 1)}$$

که فرمول بالا برای Advanced Timer ها می باشد و برای تایمر های Basic و General، RepetitionCounter ای وجود ندارد.

مقدار فرکانس کاری ARM در پیاده سازی ما برابر 84 مگاهرتز در نظر گرفته شده است.

تابع play_note به صورت زیر می باشد:

```
void play_note(int note_freq, int note_duration)
{
  if(note_freq == 440)
  {
    htim3.Instance ->CCR1 = 1909;
    __HAL_RCC_GPIOA_CLK_ENABLE();
    htim2.Init.Period = note_duration * 2;
    HAL_TIM_Base_Init(&htim2);
    HAL_TIM_Base_Start_IT(&htim2);
    HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_1);
    while(state)
    {

    }
    state = 1;
  }
  else if(note_freq == 262)
  {
    htim3.Instance ->CCR1 = 3200;
      HAL_RCC_GPIOA_CLK_ENABLE();
```

با توجه به اینکه آرم با 84 مگاهرتز در حال کار کردن می باشد برای تایمر2 که کار شمارش را دارد مقدار prescaler برابر 41999 در نظر گرفته شد، با توجه به مقدار ورودی زمان نوت، با تقسیم به عنوان period، به همان نیم ثانیه و یک ثانیه تبدیل میگردد.

برای قسمت تولید موج سینوسی با فرکانس خاص نیز، از تایمر 3 استفاده شد که مقادیر پریود و پری اسکیلر برابر صفر در نظر گرفته شدند. به کمک CCR به تولید موج با فرکانس مورد نظر می پردازیم. بدین صورت که 84 مگاهرتز تقسیم به 100 که برابر تعداد سمپل ها میباشد تقسیم بر فرکانس مد نظر (برای مثال 440) برابر CRR قرار میدهیم بدین ترتیب که با هر رسیدن به این مقدار CRR یک اینتراپت میخورد.

قسمت کال بک تایمر 2 مقدار state را برابر صفر میکنیم که از حلقه ی While بالایی خارج گردد.

در کال بک تایمر3 نیز موج سینوسی تولید شده را میفرستیم. هر بار نیز یک سمپل را میفرستیم که در مدت نیم یا یک ثانیه این موج با فرکانس مورد نظر ارسال میگردد:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim ->Instance == TIM2)
  {
    state = 0;
  }
}
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
{
  __HAL_TIM_SetCounter(&htim3, 0);
  HAL_DAC_SetValue(&hdac, DAC1_CHANNEL_1, DAC_ALIGN_12B_R, sinewave[i]);
  i++;
  if(i >= 99)
  {
    i = 0;
  }
}
```

برای تایمر2، قسمت init اش به صورت زیر می باشد:

```
/* TIM2 init function */
void MX_TIM2_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 41999;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 999;
  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();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
```

برای تایمر3 نیز قسمت init کردن اش به صورت زیر می باشد:

```
/* TIM2 init function */
void MX_TIM2_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 41999;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 999;
  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();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
```

سوالات:

۱. همان‌طور که گفته شد، با انتخاب m بزرگ‌تر می‌توان موج سینوسی دقیق‌تری تولید کرد. مشکلات و عوامل محدود کننده برای انتخاب m های بزرگ را بیان کنید.

هرچه m بزرگتر باشد هر نمونه از فرکانس سیستم بیشتر می شود در نتیجه تایمر به درستی نمیتواند بشمارد و فرکانس کاریش محدود میشه.

۲. مقادیر انتخابی یا محاسبه‌شده‌ی پارامترهایی مثل m، فرکانس تایمرها، مقادیر رجیسترهای تایمرها و نحوه‌ی انتخاب یا محاسبه‌ی آن‌ها را در گزارش کار ذکر کنید.

در قسمت های بالا این مقادیر مشخص شده اند.