بسمه تعالی

پروژه‌ی سوم درس میکروکنترلر(اختیاری)

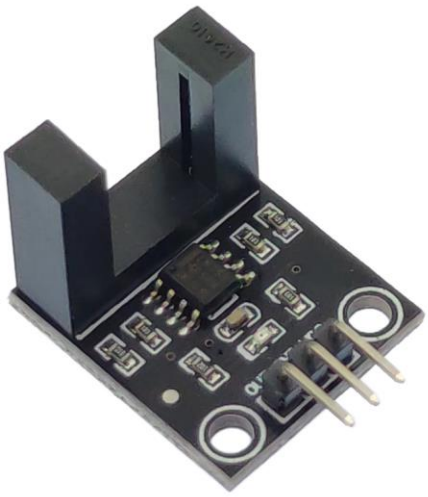
راه‌اندازی یک موتور DC و کنترل سرعت آن توسط واحد تایمر (با استفاده از input capture و PWM)

مقدمه

برای این کار به یک موتور DC با ولتاژ ۱۲ ولت، یک آداپتور ۱۲ ولت، یک اپتو کانتر، یک انکدر، یک درایور موتور(یا یک ماسفت یا یک BJT قدرت) و یک مقاومت متغیر احتیاج داریم.

در این آزمایش یک انکدر که آن را با استفاده از مقوا ساخته اید و بر سر موتور قرار داده‌اید و یا این که نوع برنجی آن را مشابه تصویر زیر تهیه نموده‌اید استفاده می‌کنیم.

این قطعه بر روی شفت موتور DC قرار میگیرد و در مقابل یک اپتوانکدر مانند شکل زیر قرار می‌گیرد:

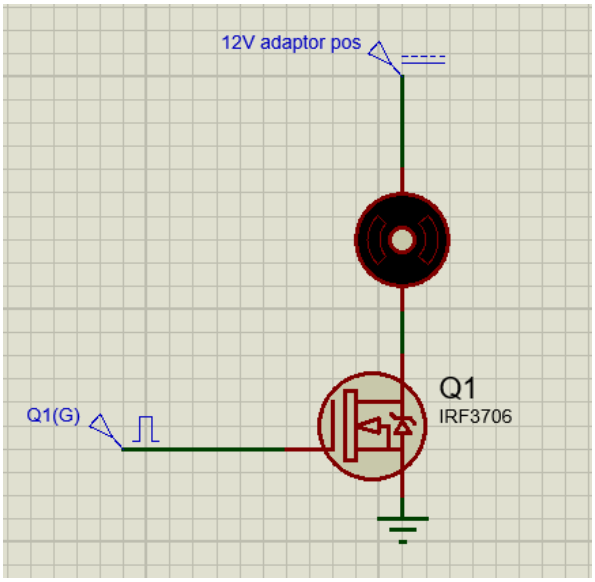


یک LED مادون قرمز در یک سمت و در سمت دیگر یک گیرنده‌ی مادون قرمز قرار دارد. یک مقایسه گر مانند lm393 با دریافت اطلاعات از خروجی گیرنده دریافت می‌کند که آیا شیئی در برابر فرستده قرار گرفته است یا خیر و خروجی خود را صفر و یک می‌کند.

می توان از لبه‌ی بالا رونده‌ی خروجی که با گذر انکدر از جلوی سنسور استفاده نمود و آن را به ورودی input capture از تایمر بدهیم.

تایمر در حال شمارش، با حس نمودن یک لبه بر روی یک از کانال‌های خود، مقدار رجیستر CNT خود را داخل رجیستر CCR کانال متصل شده به آن قرار می‌دهد. (هر تایمر دارای ۴ عدد کانال است پس رجیستر های مربوطه CCR1~CCR4 هستند)

شما با استفاده از یکی دیگر از خروجی های تایمر در حال تولید پالس PWM هستید و آن را به وردی درایور موتور، یا گیت ماسفت یا بیس BJT می‌دهید و بدین وسیله سرعت موتور را کنترل می‌کنید نحوه اتصال به صورت زیر است. (من ماسفت را برای مثال آورده ام)

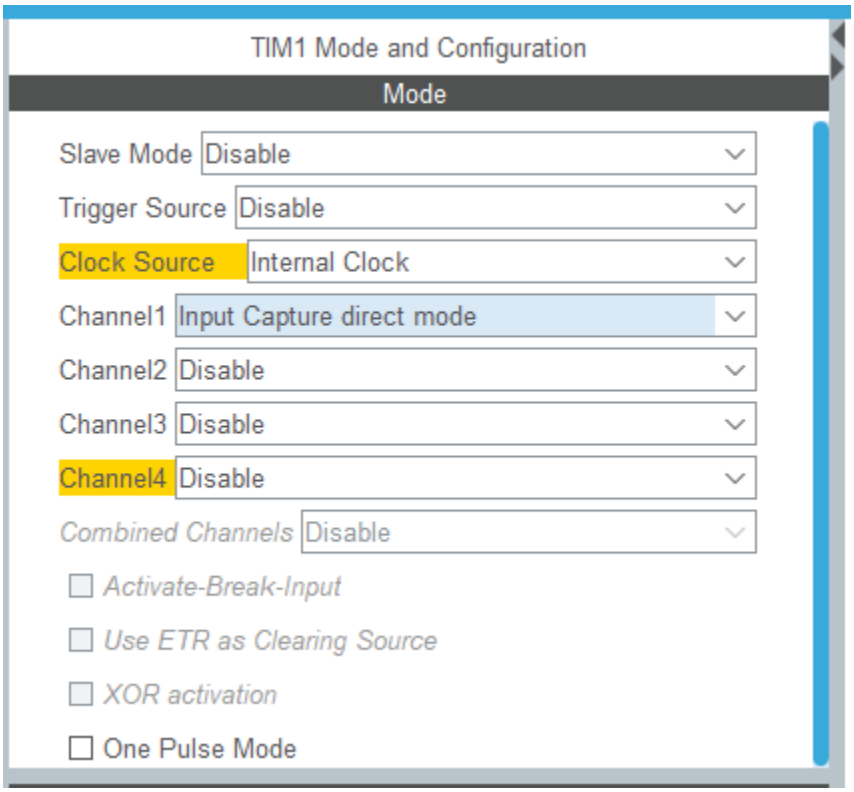


موتور از یک سمت به درین یک ماسفت نوع N و از طرف دیگر به ۱۲ ولت تامین شده از یک آداپتور متصل شده است.

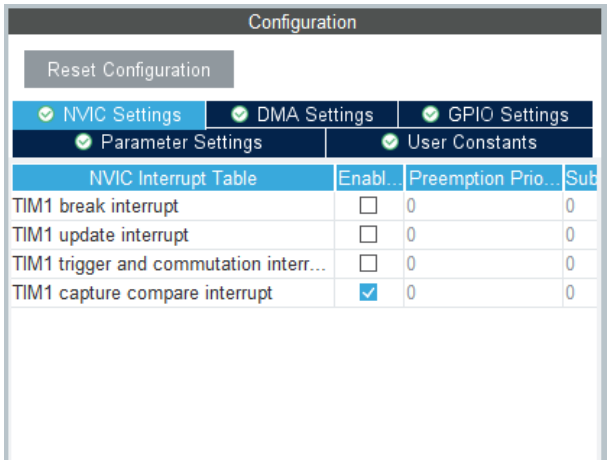
برای کنترل دور موتور از یک مقاومت متغیر استفاده می‌کنیم که به یکی از کانال‌های ADC متصل شده است.

نحوه راه اندازی PWM و ADC در آزمایش اول گفته شد. اکنون به راه اندازی input capture می‌پردازیم.

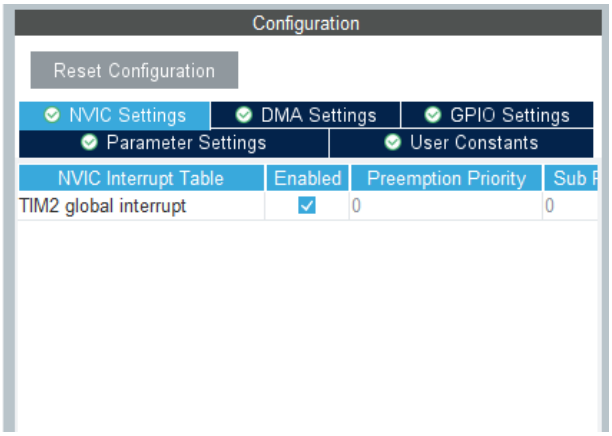
تنظیمات تایمر به صورت زیر است:



مقدار ARR را روی حداکثر قرار دهید و وقفه‌ی تایمر را فعال کنید. در بعضی تایمر ها global interrupt و در بعضی دیگر capture compare interrupt را باید فعال نمایید:



یا :



برای راه اندازی این واحد از تابع HAL_TIM_IC_START_IT در قبل از while(1) استفاده نمایید.

برای زمانی که یک لبه بر روی ورودی تایمر اتفاق، می افتد نیاز به یک callback داریم، آن را در فایل stm32f1xx_hal_tim.c می توانید پیدا کنید:

```
5556  __weak void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
5557  {
5558    /* Prevent unused argument(s) compilation warning */
5559    UNUSED(htim);
5560
5561    /* NOTE : This function should not be modified, when the callback is needed,
5562              the HAL_TIM_IC_CaptureCallback could be implemented in the user file
5563     */
5564  }
```

درون آن باید مقدار کانتر را صفر کنید(با استفاده از تابع __HAL_TIM_SET_COUNTER) و همچنین مقدار رجیستر کانال مربوطه را بخوانید(با استفاده از تابع __HAL_TIM_GET_COMPARE) . و با توجه به آن دور موتور را حساب نمایید.

مثلا اگر یک انکدر ۸ پره دارید، بایست دور موتور به صورت زیر محاسبه شود:

$$RPM = 8*(CCR1*(1/f_{timer}))$$

که مقدار CCR1 با فرض اتصال به کانال ۱ و f_{timer} با تقسیم فرکانس کلاک ورودی تایمر بر روی prescaler+1 محاسبه می‌شود.

نحوه‌ی انجام آزمایش:

بعد از روشن شدن میکرو ابتدا مقدار duty cycle از PWM را بر روی 100 درصد قرار دهید و مقدار دور موتور را بر روی LCD نشان دهید.(این کار برای ۱ دقیقه‌ی اول برنامه انجام دهید و در ادامه نیاز نیست)

۱. مقدار ADC را بایست خواند و در در خط دوم LCD آن را نشان داد
۲. مقدار دور موتور پیوسته بر روی خط اول LCD به همراه درصد آن نسبت به کل دور موتورممکن نشان داده شود.
۳. با توجه به نسبت LCD به کل رزولوشن، مقدار دور موتور را به همان نسبت باید تنظیم نمود. یعنی:

دور موتور = کل دور موتور ممکن * مقدار خوانده شده از ADC تقسیم بر رزولوشن ADC (۴۰۹۶ در صورت تنظیم بر روی ۱۲ بیت)

۴. ابتدا مقدار PWM احتمالی که همان درصد duty cycle برابر با درصد ADC است را تنظیم می-کنیم. سپس با فاصله‌های ۱ میلی ثانیه با توجه به فیدبک دریافتی از input capture مقدار PWM را کم یا زیاد می‌کنیم تا آنجا که به دور موتور مورد نظر خود برسیم.(اگر دور موتور کم است، یک عدد به CCR مربوط به pwm اضافه می کنیم و بالعکس)
۵. در نهایت درصد دور موتور نوشته شده در خط اول باید با درصد ADC در خط دوم LCD یکسان بماند.

موفق باشید.