


آزمایش ۳: منابع پالس ساعت و کنترل توان

هدف آزمایش: آشنایی با منابع تأمین پالس ساعت در STM32F429، آشنایی با واحد ضرب‌کننده و مقسم فرکانس

 مدت زمان آزمایش: ۴ ساعت

بخش اول، PLL

در میکروکنترلرهای ۸۰۵۱، فرکانس کریستال میکروکنترلر بر ۱۲ تقسیم شده و به CPU و سایر قسمت‌ها وارد می‌شود، در میکروکنترلرهای AVR پالس ساعت بدون مقسم به CPU و سایر قسمت‌ها وارد می‌شود اما در میکروکنترلرهای ARM این امکان وجود دارد که ضربی از پالس ساعت وارد شده به میکروکنترلر به CPU و سایر قسمت‌ها داده شود. به دلیل این که بخش‌های فرکانس بالای مدار در داخل میکروکنترلر قرار خواهند گرفت، موجب بهبود سازگاری الکترومغناطیسی (EMC) خواهد شد.

 با مراجعه به راهنمای مرجع STM32F429، با واحد Reset and Clock Control (RCC) آشنا شده و رجیسترهای آن را بررسی کنید (فصل ۶).

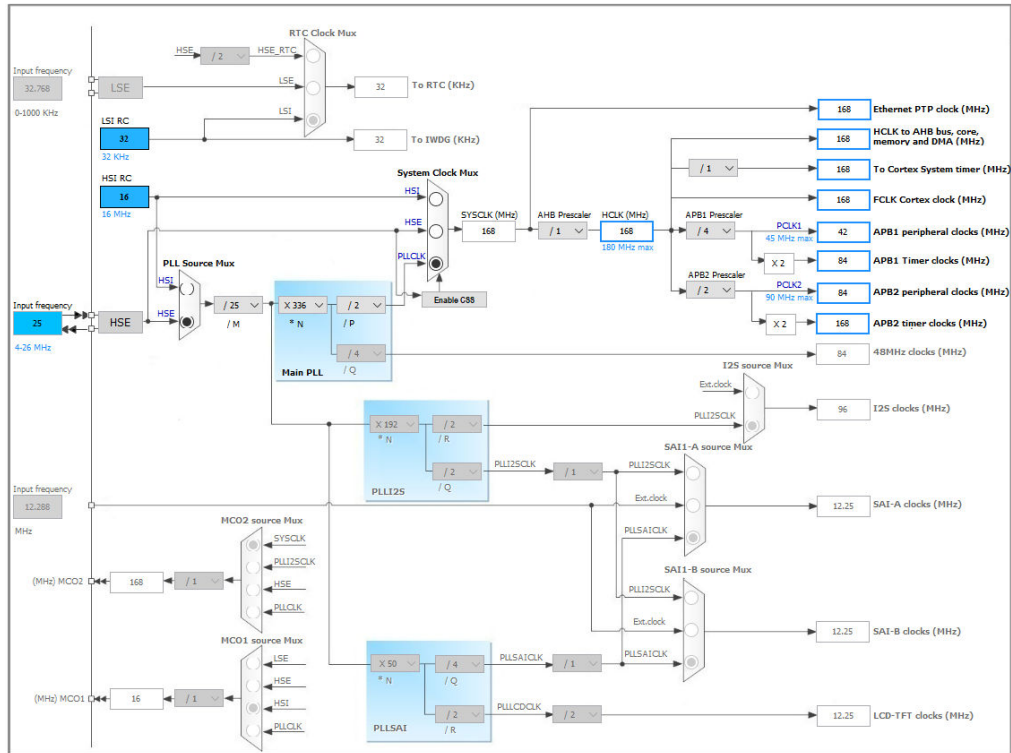
؟ سؤال ۱: نقش واحد PLL^۵ در میکروکنترلر چیست؟ چند واحد PLL در خانواده STM32F429 وجود دارد و کاربرد هر کدام چیست؟



در میکروکنترلرهای STM32F429 امکان استفاده از چند منبع پالس ساعت برای تأمین کلاک CPU و سایر قسمت‌های جانبی (کلاک سیستم) وجود دارد. این منابع شامل اسیلاتور خارجی (HSE)، PLL1 و اسیلاتور RC داخلی (HSI) است. هم‌چنین ورودی خود PLL1 می‌تواند از اسیلاتور خارجی یا اسیلاتور داخلی تأمین گردد (شکل ۷).

علاوه بر پالس ساعت مربوط به کلاک سیستم، منابع پالس ساعت دیگری نیز در STM32F429 وجود دارد (LSE و LSI) که برای راه‌اندازی واحد تاریخ و ساعت حقیقی (RTC) و چند کاربرد دیگر مورد استفاده قرار می‌گیرند.

^۵ Phase locked loop




شکل ۷ - نمودار منابع کلاک در STM32F429ZIT

باید به این نکته توجه داشت که هنگام تغییر منبع پالس ساعت ورودی و عوض کردن تنظیمات PLL1، ابتدا باید آن را به صورت نرم‌افزاری از خط پالس ساعت قطع کرده، سپس آن را غیرفعال نمایید و پس از آن اقدام به هر تغییری در PLL1 کرد. انتخاب منبع پالس ساعت برای PLL1 به صورت نرم‌افزاری و از طریق رجیستر CFGR صورت می‌پذیرد.

سؤال ۲: با بررسی رجیسترهای CR و CFRG در واحد RCC بررسی کنید که در حالت پیش‌فرض CPU با چه فرکانسی کار می‌کند و پالس ساعت آن چگونه تأمین می‌گردد.

تنظیمات PLL را می‌توان توسط برنامه‌ای آماده در فایل system_stm32f4xx.c و یا با استفاده از تنظیم رجیسترها در برنامه‌ی اصلی تغییر داد.


با مطالعه‌ی پیوست ۵ با روش‌های دیباگ به کمک ST-Link آشنا شوید. 


آزمایش ۱: ب.ب.ک. دو رجیستر CR و CFGR را در دیباگر نمایش داده و بررسی کنید CPU با چه فرکانسی در حال کار است. این پالس ساعت از کجا تأمین شده است؟ توجه داشته باشید که نرم‌افزار Keil با اجرای system_stm32f4xx.c تنظیمات پالس ساعت را تغییر می‌دهد؛ بنابراین در حالت پیش‌فرض نخواهید بود.

توجه: حداکثر فرکانس کاری هسته میکروکنترلرهای STM32F429 طبق داده‌های برگه‌ی اطلاعاتی آن ۱۸۰ مگاهرتز است. توجه نمایید که پیکربندی میکروکنترلر برای فرکانس‌های بالاتر از ۱۸۰ مگاهرتز ممکن است موجب آسیب دیدن آن می‌شود.

 **تحقیق ۱:** وظیفه‌ی واحد CSS چیست و چگونه فعال می‌گردد؟

خروجی PLL1 برای وارد شدن به سایر قسمت‌های جانبی میکروکنترلر بالا بوده و باید کاهش یابد. بدین منظور تقسیم‌کننده‌های مجزایی برای بخش‌های مختلف تعبیه شده است. تقسیم‌کننده‌ی مربوط به کلاک باس‌های پرسرعت (APB2) و کم‌سرعت (APB1) نیز از طریق رجیستر CFGR قابل تنظیم هستند.

 **تحقیق ۲:** در مورد نحوه‌ی تأمین کلاک مربوط به بخش GPIO تحقیق کنید و بگویید لزوم مقداردهی به رجیستر APB^{NR} در آزمایش ۲ چه بود؟

 **آزمایش ۲:** ب.ب.ک. PLL میکروکنترلر را خاموش کنید و سپس برنامه‌ای بنویسید که یک LED به صورت چشمک‌زن با فواصل حدود ۱ ثانیه روشن-خاموش شود. با فشردن کلید برد، PLL را به صورت نرم‌افزاری طوری تنظیم کنید که کلاک CPU بر روی ۴۰ مگاهرتز قرار گیرد که منبع آن اسیلاتور داخلی باشد. برنامه‌ی خود را بر روی برد آموزشی پیاده کرده و نتیجه را گزارش نمایید.

بخش دوم، SysTick Timer

یکی از پرکاربردترین توابع مورد استفاده در برنامه‌ریزی میکروکنترلرها، توابع ایجاد تأخیر دقیق هستند. ایجاد تأخیر دقیق را می‌توان به دو صورت انجام داد ۱- اجرای دستورات مشخص که تعداد کلاک لازم برای اجرای آن‌ها مشخص است. این کار معمولاً باید به زبان سطح پایین اسمبلی نوشته شود و عموماً شامل چند دستور NOP می‌شود. برنامه‌نویسی ایجاد تأخیر معمولاً به دلیل بهینه‌سازی‌هایی که کامپایلر بر روی کد نوشته شده انجام می‌دهد، به زبان C صورت نمی‌گیرد. ۲- استفاده از تایمرها و شمارنده‌ها. عموماً می‌توان تایمرها را طوری برنامه‌ریزی کرد که به ازای هر کلاک به اندازه یک واحد شمارش کنند و از طریق تنظیم مقدار شمارش‌ها، تأخیر دلخواه را ایجاد نمود. اما یکی از پرکاربردترین بخش‌های جانبی هر میکروکنترلر تایمر آن است که مورد استفاده قرار می‌گیرد و استفاده از آن به عنوان واحد ایجاد کننده‌ی تأخیر مطلوب نیست. در میکروکنترلرهای سری Cortex-M3 و بعد از آن، واحدی به نام System Tick Timer بدین منظور اضافه شده است. SysTick تایمر، یک تایمر ساده ۲۴ بیتی است که با هر بار صفر شدن می‌توان تولید وقفه کند. کلاک تأمین‌کننده‌ی

این واحد مستقیماً از کلاک CPU (باس AHB) گرفته می‌شود و یا می‌توان از یک-هشتم این کلاک را مورد استفاده قرار داد.

📖 با مراجعه به راهنمای برنامه‌نویسی میکروکنترلرهای STM32F4، با بخش SysTick Timer (فصل ۵) آشنا شده و رجیسترهای آن را بررسی نمایید.

🔧🏆 **آزمایش ۳:** فایلی کتابخانه‌ای با نام delay.h ایجاد کرده و در آن سه تابع با نام‌های delay_us، delay_ms، delay_s بنویسید که مقدار ورودی x را دریافت کرده و با استفاده از SysTick به ترتیب تأخیر بر حسب میکروثانیه، میلی ثانیه و ثانیه به اندازه‌ی x ایجاد کنند. سپس با استفاده از این توابع برنامه‌ای بنویسید که یک موج 1KHz بر روی پایه‌ای دلخواه از میکروکنترلر تولید کند. برای دستیابی به رجیسترهای این واحد از تعاریف جدول ۲ استفاده کنید.

توجه: تابعی با نام SysTick_Config در فایل core_cm3.h برای برنامه ریزی واحد SysTick تعریف شده است. ورودی این تابع تعداد کلاک بین دو وقفه ایجاد می‌کند. فراخوانی این تابع با مقدار دهی مناسب موجب فعال شدن دائمی این واحد همراه با تولید وقفه دائمی می‌شود. در صورت موفقیت در پیکربندی SysTick، تابع مقدار صفر و در غیر این صورت مقدار یک برمی‌گرداند. در این آزمایش دانشجویان نسبت به نوشتن یک تابع شخصی مشابه اقدام نمایند و مجاز به استفاده از این تابع پیش تعریف شده نیستند.

جدول ۲ - رجیسترهای تایمر SysTick

شرح	تعریف مورد استفاده
SysTick Control and Status Register	<code>SysTick->CTRL</code>
SysTick Reload Value Register	<code>SysTick->LOAD</code>
SysTick Current Value Register	<code>SysTick->VAL</code>
SysTick Calibration Register	<code>SysTick->CALIB</code>