آزمایش نهم

# مقدمه

تا اینجا با دو روش هم‌زمان‌سازی کارها در میکروکنترلرها آشنا شده‌اید. در این آزمایش به یادگیری روشی به روز برای هم‌زمان‌سازی به کمک یک هسته سیستم عامل خواهید پرداخت. FreeRTOS یک هسته سیستم‌عامل بی‌درنگ، ساده، کوچک و پر کاربرد برای سیستم‌های نهفته است که بر روی بیش از ۳۵ معماری متفاوت قابل استفاده است. در این آزمایش به کمک FreeRTOS به هم‌زمان‌سازی چند کار متفاوت خواهید پرداخت و از Timer، UART، LCD، GLCD و LED که در آزمایش‌های قبل یاد گرفتید نیز استفاده خواهید کرد. فایل کتابخانه‌های مورد نیاز این آزمایش در یک فایل Zip در صفحه درس قرار داده شده است.

* برای راه‌اندازی FreeRTOS بر روی میکروکنترلر LPC1768 در برنامه Keil uVision می‌بایست در هنگام ایجاد پروژه، در قسمت انتخاب Run-Time Environment، موارد زیر را انتخاب کنید تا هسته FreeRTOS برای شما فعال شود:
  + CMSIS / Core
  + CMSIS / RTOS / FreeRTOS
  + CMSIS / RTOS2 / FreeRTOS
  + RTOS / Config (CMSIS RTOS2)
  + RTOS / Core
  + RTOS / Coroutines
  + RTOS / Event Groups
  + RTOS / Heap (Heap\_4)
  + RTOS / Message Buffer
  + RTOS / Stream Buffer
  + RTOS / Timers
* برای استفاده از این هسته و برنامه‌ریزی کارها ابتدا می‌بایست کتابخانه‌های لازم را در پروژه فراخوانی کنید:

#include <FreeRTOS.h>

#include <task.h>

* در ابتدای برنامه می‌بایست توابع انجام کار در فرمت‌های زیر ایجاد شوند:
  + برای کارهایی که باید به طور مداوم در حلقه اجرا شوند:

void task\_func(void \*pvParameters){

while (1){

// Your code should write here...

}

}

* + برای کارهایی که نیاز به حلقه ندارند و می‌بایست یکبار اجرا شوند:

void task\_name(void \*pvParameters){

// Your code should write here...

// Place delete function at the end of the function

}

* در ابتدای تابع main برای ایجاد کار جدید می‌بایست از دستور زیر، کمک بگیرید:

xTaskCreate(pvTaskCode, pcName, usStackDepth,

pvParameters, uxPriority, pxCreatedTask);

// pxTaskCode is pointer to the task entry function

// pcName is a descriptive name for the task

// usStackDepth is the size of the task stack specified

// pvParameters is that will be used as the parameter for the task

// uxPriority is the priority at which the task should run

// pxCreatedTask used to pass back a handle by which the created task

* + به عنوان مثال برای ایجاد یک کار ساده از دستور زیر استفاده می‌شود:

xTaskCreate(task\_func, “task\_name”, 128, NULL, 1, NULL);

* پس از تعریف کارها می‌بایست تابع زیرا را برای شروع به کار صدا بزنید. توجه کنید که این تابع موجب ایجاد مسدودیت در تابع اصلی میکرو خواهد شد و در صورت وجود برنامه‌ای پس از این تابع، آن برنامه اجرا نخواهد شد و تمام برنامه‌ها می‌بایست در قالب کارهای تعریف شده در هسته سیستم عامل تعریف و اجرا شوند.

vTaskStartScheduler(); // Blocks here

* برای ایجاد تاخیر در کارها می‌توانید از تابع زیر استفاده کنید. این تابع تأخیر با توابع قبلی که در کتابخانه‌ها وجود داشت، بسیار متفاوت است. برای تعیین زمان مورد نیاز برای تأخیر می‌بایست مقدار تأخیر مورد نیاز در واحد میلی‌ثانیه را بر ثابت portTICK\_PERIOD\_MS تقسیم کرد. مثال زیر تأخیر x میلی‌ثانیه را ایجاد خواهد کرد و به جای متغیر x می‌بایست مقدار تأخیر مورد نیاز خود را قرار دهید.

vTaskDelay(x/portTICK\_PERIOD\_MS); // Makes x milliseconds delay

* برای حذف یک کار می‌توانید از تابع زیر استفاده کنید.

vTaskDelete(xTask); //xTask returned from xTaskCreate();

* در صورت نیاز به استفاده از توابعی غیر از توابع توضیح داده شده در می‌توانید از «[لینک](https://www.freertos.org/Documentation/RTOS_book.html)» اقدام به مطالعه فایل‌های Reference Manual کتابخانه FreeRTOS نمایید.
* برای استفاده از سایر کتابخانه‌ها، پس از انتخاب موارد ذکر شده از قسمت Run-Time Env.. تمام فایل‌ها را در محیط ویندوز در کنار فایل main.c خود قرار دهید. فایل‌های .c و .s را در محیط برنامه keil به کنار فایل main.c واقع در Source Group 1 اضافه کنید. (کلیک راست روی فولدر، Add existing Files…)
* فراموش نکنید که در ابتدای برنامه خود تابع SystemInit() را فراخوانی کنید.
* حتما فایل‌های کتابخانه‌ها را مطالعه کنید و با نحوه کار آن‌ها آشنا شوید.

# پیش‌آزمایش

* در مورد مفاهیم semaphore و mutex در سیستم‌های عامل تحقیق کنید.
* تفاوت دو تابع DELAY\_ms در کتابخانه و vTaskDelay در هسته سیستم عامل چیست؟
  + مزایا و معایب استفاده از هر کدام چیست؟
* چه سیستم‌های عاملی قابلیت اجرا بر روی میکروکنترلرهای ARM را دارند؟

# صورت آزمایش

## اتصالات

اتصالات روی بورد در این آزمایش به شرح زیر است:

* اتصال اتصال چهار دیود نورانی LED0..LED3 به P1.0, P1.1, P1.4,P1.8
* اتصال پایه‌های GLCD‌ (RS,RW,EN,CS1,CS2,CS3) به P2.8..P2.13
* اتصال پایه‌های LCD‌G D0..D7 به P2.0..P2.7
* اتصال پایه‌های LCD‌ (RS,RW,E) به P0.0..P0.2
* اتصال پایه‌های LCD‌ DB4..DB7 به P0.4..P0.7
* اتصال پایه‌های USB-TTL (TX, RX) به P0.10,P0.11 (UART2)

## قسمت اول

چهار تابع برای چهار کار متفاوت تعریف کنید و در هر کدام از آن‌ها‌ یکی از چهار LED متصل به میکرو را با فرکانس‌های 1Hz, 2HZ, 5Hz, 10Hz روشن و خاموش کنید.

برنامه نوشته شده به همراه فیلم کوتاهی از عملکرد LEDها را در گزارش خود قرار دهید.

## قسمت دوم

قسمت سوم آزمایش سوم و قسمت دوم آزمایش چهارم را مجددا و این بار به کمک هسته سیستم عامل FreeRTOS در دو تابع کار مجزا پیاده‌سازی کنید.

یک شمارنده دو قسمتی طراحی کنید که هر قسمت با فرکانس متفاوتی کار می‌کند. هر قسمت از این شمارنده را در یک خط از LCD نمایش دهید. فرکانس هر قسمت از طریق فرمول زیر که در آن g معادل شماره گروه‌تان است، محاسبه می‌شود:

محاسبات تاخیرها، برنامه نوشته شده و یک فیلم کوتاه (حدود ۱دقیقه) از نمایشگر را در گزارش خود قرار دهید.

## قسمت سوم (۲۰ درصد امتیازی)

برنامه‌ای بنویسید که در یک تابع کار کاراکترها را از UART دریافت کند و به ازای هر کاراکتر **یک کار جدید ایجاد کند** و آن کاراکتر را به همراه مختصات Cursor LCD به عنوان پارامتر و نه به صورت متغیر جهانی به آن پاس بدهید تا آن را به صورت زنده بر روی LCD نمایش دهد. نام‌های اعضای گروه را از طریق برنامه Hercules به میکروکنترلر ارسال کنید و خروجی را مشاهده کنید. در این قسمت استفاده از LCD یا GLCD تفاوتی ندارد و انتخاب با خودتان است.

برنامه نوشته شده به همراه تصویر برنامه Hercules و خروجی نمایشگر را در گزارش خود قرار دهید.

موفق باشید