

FreeRTOS

Ходневич Віталій

КНУ ім. Т. Шевченка

2015

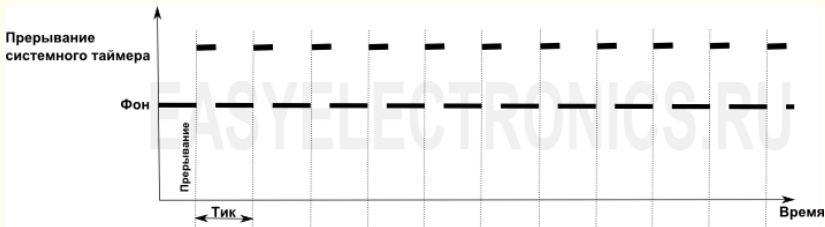


FreeRTOS – багатозадачна операційна система реального часу (ОСРЧ) для вбудованих систем

- ▶ Задача. Керування задачами. Пріоритети
- ▶ Черги
- ▶ Семафори
- ▶ М'ютекси
- ▶ Системний таймер
- ▶ Диспетчер і багатозадачність

Системний тік

Один з таймерів (SysTick у випадку STM32) налаштовується на генерацію системних тіків приблизно раз в 1 мс (можна змінювати в залежності від задач)



Кожен тік – це виклик переривання таймера в котрому викликається диспетчер. Цей час ще називається квантом часу.

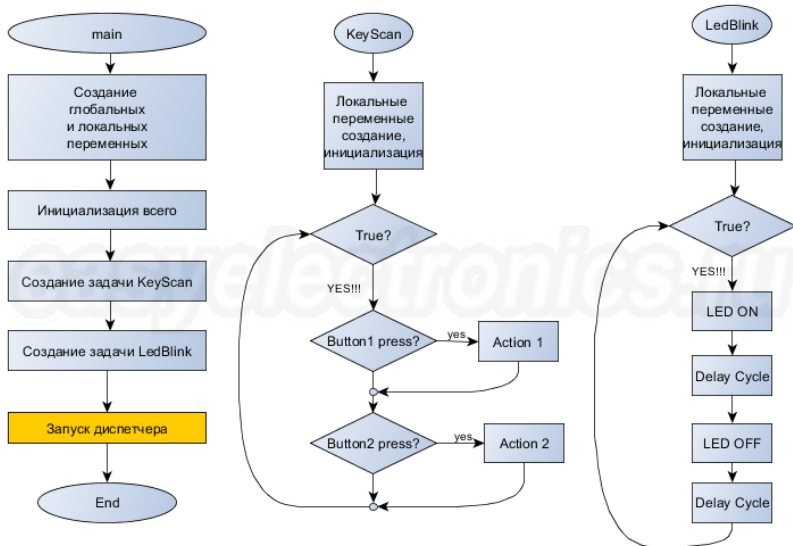
Задача (Thread)

Наріжним каменем будь якої ОС є задача. Задача виглядає як функція, що крутить нескінченний цикл виконуваних якихось простих процедур. Наприклад:

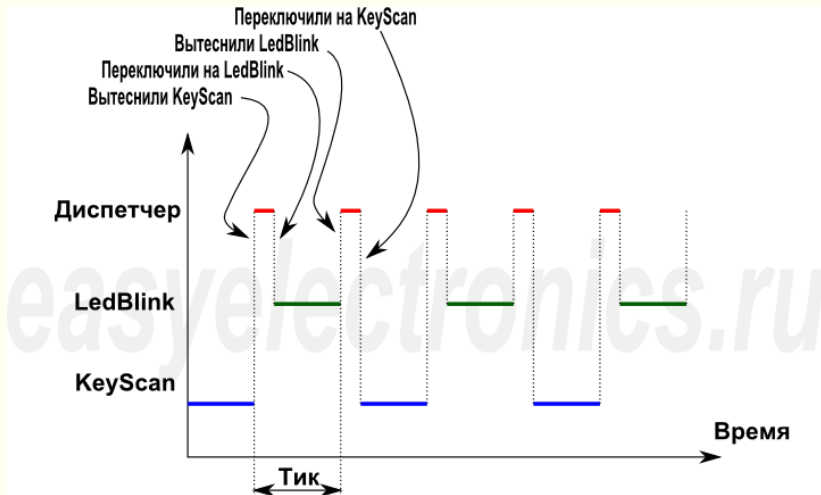
```
void StartLED1(void const * argument)
{for(;;){
HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,GPIO_PIN_SET);
osDelay(100);
HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,GPIO_PIN_RESET);
osDelay(100);
}}
```

Нормальними є нескінченні цикли і затримки у функціях.

Класичний алгоритм



Виконання з диспетчером



Виконання з диспетчером

Диспетчер піклується про те аби у кожної задачі все зберігалось і запам'ятовувалось: поточне положення, стек, змінні, реєстри... Сама ж задача має відповідати ряду правил (наприклад, не лізти куди не слід).

Початково задача не ініціалізована. Тобто задачі як такої нема, а є лише кусок скомпільованого коду за відомою адресою і ми знаємо, що се наша задача. В цьому випадку вона займає лише місце на флеші і все.

Запуск задачі

```
osThreadDef(defaultTask1, StartLED1,  
              osPriorityNormal, 0, 128);  
defaultTaskHandle = osThreadCreate(  
              osThread(defaultTask1), NULL);
```

Тут зазначається наша задача, її пріоритет, ім'я для налагодження і скільки пам'яті ми під неї виділяємо. В результаті під неї виділяється кусок пам'яті, заводиться свій стек і відпускається у вільне плавання. В котрого може бути кілька станів:

Запуск задачі

- ▶ **READY** Задача запущена і готова прийняти на себе управління. Чекає лише на увагу зі сторони диспетчера. Як тільки дійде її черга відразу задача переходить в режим RUN
- ▶ **RUN** Тобто диспетчер перемкнув увагу на неї і процесор проганяє її код в даний момент
- ▶ **WAIT** Задача в режимі очікування. Наприклад чекає на якусь подію (поки таймер натікає, кнопка натиснеться). При цьому процесорний час не витрачається. Як тільки очікувана подія відбулась RTOS назначає цій задачі READY.
- ▶ **SUSPEND** Вимкнено. Тобто задача не вигружена з пам'яті, дані її збережені, але вона неактивна. На жодні події не реагує і сама з цього стану не вийде.

Керування задачами

Вбити задачу можна командою `vTaskDelete(...)`. При цьому вона буде вивантажена з пам'яті, звільнено оперативну пам'ять і локальні змінні будуть втрачені. Але можна буде запустити її заново.

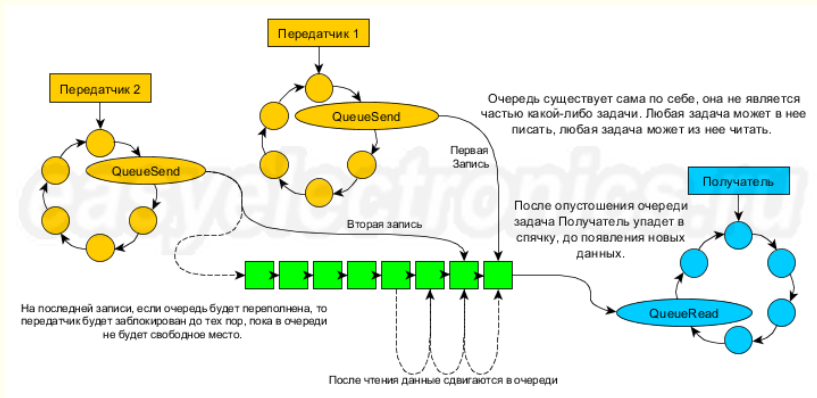


Пріоритети

Пріоритет задачі задається при її створені і визначає в якому порядку будуть працювати задачі. Тобто якщо є дві задачі `READY`, то першою виконається та задача у якій пріоритет вище, а інша чекатиме. Якщо є `READY` задач немає, то буде виконуватись `IDLE` цикл. Вньому йде обслуговування пам'яті, зачистка не використовуваної оперативки, видалення залишків від видалених задач та іншої службової інформації. Туди ж можна помістити свою `callback` функцію в котрій контролер відправляється в режим енергозбереження.

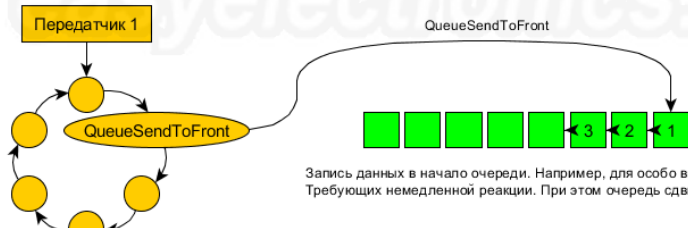
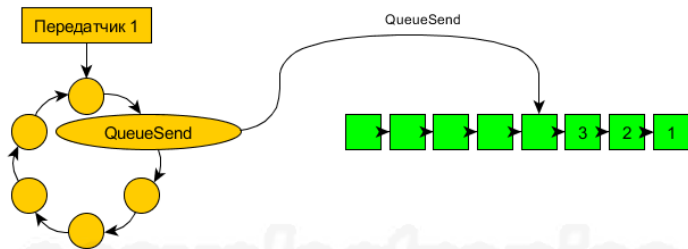
Черги. Обмін даними між задачами

Глобальні змінні - погана ідея!!!



Черги. Обмін даними між задачами

Диспетчер слідкує за тим, щоб з черги неможна було прочитати до того як там все запишеться, а також записати в чергу якщо вона заповнена.



Запись данных в начало очереди. Например, для особо важных данных. Требующих немедленной реакции. При этом очередь сдвигается назад.

Семафори

Семафор — це універсальний механізм для організації взаємодії процесів.

Розв'язує задачі взаємного виключення та синхронізації потоків. Він є одним з найстаріших засобів розподілення доступу процесів, що працюють паралельно, до критичних ресурсів. Семафори використовуються для контролю доступу до спільного ресурсу, або для синхронізації процесів (потоків).

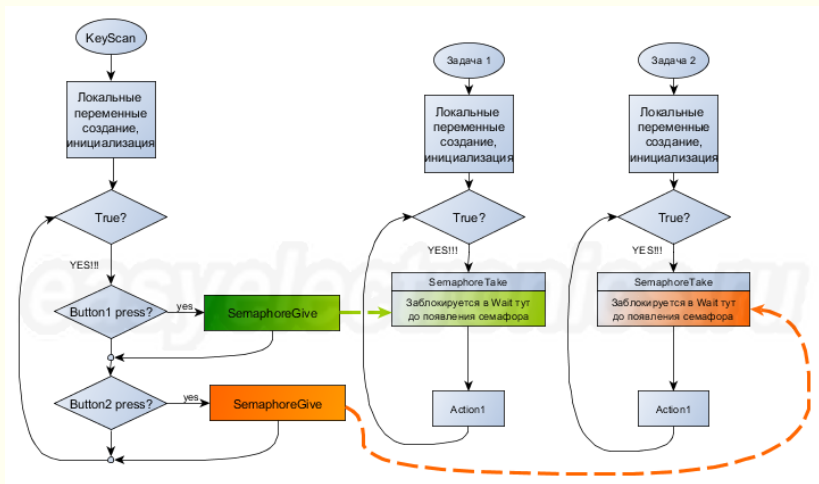


Семафори

Простий бінарний семафор схожий на звичайний флажковий механізм при традиційному програмуванні. Коли деякий процес (наприклад, периферія чи переривання) флажок піднімає, а фонові програми цей флажок відслідковує в циклі. Тільки тут цикл на себе бере диспетчер.

Семафори

Недолік: незрозуміло скільки разів поставили семафор:



Рішення: підрахунковий семафор. Різниця в тому, що в підрахункового семафору стан не одиничний флажок, а деяка змінна. Задача котра його видає цю змінну збільшує, а котра зчитує - зменшує. Коли дорахує до нуля – WAIT.

Mutex

М'ютекс – об'єкт взаємного виключення (mutual exclusion), призначений для захисту певного об'єкта у потоці від доступу інших потоків. М'ютекс є одним із засобів синхронізації роботи потоків або процесів.

Наприклад: запис в порт, читання з диску, користування туалетом =)

Основне призначення – організація взаємного виключення для потоків з одного і того ж або різних процесів.

М'ютекси – це прості двійкові семафори, які можуть перебувати в одному з двох станів, – сигнальному або несигнальному (відкритий і закритий відповідно). Коли потік стає власником м'ютекса, він переводиться в несигнальний стан.

Системний таймер

Це програмний таймер. Він дозволяє запустити будь-яку функцію по таймеру. На відміну від задачі, вона не крутиться в нескінченному циклі, а запуститься один раз або періодично.

Диспетчер і багатозадачність

Диспетчер керує усіма задачами, чергами, м'ютексами, семафорами...

Його можна лише запустити і зупинити. Зазвичай він сидить в таймері (SysTick). Таймер генерує системні тіки скажімо раз в 1 мс. І кожен такий тік викликається диспетчер і обслуговує всі функції ядра.

Багатозадачність може бути витісняючою (псевдопаралельне виконання) і кооперативною. В останньому випадку диспетчер викликається не кожен системний тік, а вручну програмістом. Тут є проблеми з пріоритетністю, але розумне використання кооперативної багатозадачності дозволяє економити пам'ять (оскільки OS переключається в конкретні, за відома відомі, моменти, а значить потрібно менше зберігати даних)

Залишилось:

- ▶ Обробка помилок
- ▶ Co-Routines

Де дізнатись більше?

- ▶ Таненбаум. Операційні системи
- ▶ Офіційна документація на FreeRTOS
- ▶ А. Курніц FreeRTOS –операционная система для микроконтроллеров