|  |  |
| --- | --- |
| 1. Изучить теоретическую часть.  2. Изменить проект ЛР 3, добавив FreeRTOS с помощью CubeMX  (вкладка Middleware).  3. Изучить структуру нового проекта.  4. Скомпилировать проект без изменений в исходном коде и запустить в ITMO.cLAB.  5. Определить, почему в таком случае система cLAB возвращает ошибку (связано с трассировочным логом) и решить данную проблему.  6. Разработать алгоритм программы, реализующей действия согласно варианту из таблицы 1.  7. Написать программное обеспечение в соответствии с разработанным алгоритмом.  8. Собрать проект в Debug режиме.  В одной задаче необходимо с периодом 1 сек. генерировать значение uint32\_t X, которое будет являться параметром для третьей задачи.  В другой задаче необходимо генерировать с тем же периодом значение uint32\_t Y, которое также будет являться параметром для третьей задачи.  В третьей задаче считается следующая функция: Z = X - Y, причем параметры в эту задачу передаются через очереди. В трассировочный график выводить сгенерированные значения и вычисленные значения функции.  1. Теоретическую часть:  ○ Принцип работы FreeRTOS;  ○ Назначение и применение FreeRTOS;  ○ Преимущества и недостатки FreeRTOS;  ○ Примитивы синхронизации;  ○ Кооперативную и вытесняющую многозадачность;  ○ Принцип действия планировщика задач;  ○ Состояния задачи;  ○ Квант времени работы планировщика;  ○ Приоритет задачи.  2. Программную часть:  ○ Установку и использование FreeRTOS в STM32CubeMX и в  STM32CubeIDE с использованием фреймворка HAL;  ○ Структуру проекта, использующего операционную систему  FreeRTOS. | 1. Học phần lý thuyết.  2. Sửa đổi dự án LR 3 bằng cách thêm FreeRTOS bằng CubeMX  (Tab phần mềm trung gian).  3. Kiểm tra cấu trúc của dự án mới.  4. Biên dịch dự án mà không có thay đổi trong mã nguồn và chạy nó trong ITMO.cLAB.  5. Xác định lý do tại sao, trong trường hợp này, hệ thống cLAB trả về lỗi (liên quan đến bản ghi theo dõi) và giải quyết vấn đề này.  6. Phát triển một thuật toán cho một chương trình thực hiện các hành động phù hợp với tùy chọn từ Bảng 1.  7. Viết phần mềm phù hợp với thuật toán đã phát triển.  8. Xây dựng dự án ở chế độ Gỡ lỗi.  In one task it is necessary with a period of 1 sec. generate a uint32\_t X value, which will be a parameter for the third task.  In another task, it is necessary to generate the uint32\_t Y value with the same period, which will also be a parameter for the third task.  In the third task, the following function is considered: Z = X - Y, and the parameters are transferred to this task through queues. Output the generated values and the calculated values of the function to the trace graph.  1. Phần lý thuyết:  ○ Cách hoạt động của FreeRTOS;  ○ Mục đích và ứng dụng của FreeRTOS;  ○ Ưu nhược điểm của FreeRTOS;  ○ Nguyên thủy đồng bộ hóa;  ○ Hợp tác và đa nhiệm phủ đầu;  ○ Cách thức hoạt động của bộ lập lịch tác vụ;  ○ Trạng thái nhiệm vụ;  ○ Lượng thời gian làm việc của người lập lịch;  ○ Ưu tiên nhiệm vụ.  2. Phần mềm:  ○ Cài đặt và sử dụng FreeRTOS trong STM32CubeMX và trong  STM32CubeIDE sử dụng khung HAL;  ○ Cấu trúc dự án sử dụng hệ điều hành  FreeRTOS. |

|  |  |
| --- | --- |
| Общие сведения  FreeRTOS – многозадачная операционная система реального времени для встраиваемых систем. За переключение между задачами отвечает диспетчер, работающий по прерыванию от таймера. Задача выглядит как обычная функция Си, которая выполняет некоторое действие в бесконечном цикле. Для STM32 используется CMSIS-RTOS API.  Пример задачи, переключающей состояние светодиода раз 500мс:  void Start\_ledTask(void const \* argument)  {  for(;;)  {  HAL\_GPIO\_TogglePin(GPIOD,GPIO\_PIN\_13);  osDelay(500);  }  }  Так как задачи выполняются асинхронно, то при использовании общей переменной данные могут теряться. Для обеспечения правильного обмена данными между задачами используются очереди. Одни задачи кладут данные в очередь, другие оттуда читают. За переполнением очереди и чтением после окончания записи следит диспетчер. Если очередь пуста/переполнена, то та задача, которая собирается считать/записать туда данные переводится в состояние WAIT и диспетчер выведет ее из этого состояния только когда очередь будет готова принять/отдать данные.  Семафоры бывают двух типов: бинарные и счетные. Работа бинарного семафора очень похода на работу флага. То есть, когда некое событие поднимает флаг, а программа этот флаг отслеживает в цикле. В FreeRTOS функцию этого цикла выполняет диспетчер. Счетный семафор работает по такому же принципу. Разница лишь в том, что у него состояние это не единичный флаг, а некая переменная. Задача, которая выдает семафор эту переменную увеличивает, а которая считывает – уменьшает.  osThreadCreate Начать выполнение потока  osThreadTerminate Закончить выполнение потока  osThreadYield Передвать выполнение другому потоку  osThreadGetId Получить индефикатор потока для ссылки  на него  osThreadSetPriority Изменить приоритет выполнения потока  osThreadGetPriority Получить текущий приоритет выполнения  потока  osDelay Подождать в течение указанного времени  osWait Ожидать события типа Signal, Message, Mail  osTimerCreate Определить атрибуты Callback таймера  osTimerStart Запустить таймер с назначением времени  osSignalSet Установить сигнальные флаги потока  osSignalClear Сбросить сигнальные флаги потока  osMutexCreate Инициализация мьютекса  osMutexWait Получить мьютекс или подождать, пока он  не станет доступным  osMutexRelease Освободить мьютекс  osMutexDelete Удалить мьютекс  osSemaphoreCreate Инициализация семафора  osSemaphoreWait Получить семафор или подождать, пока он  не станет доступным  osSemaphoreRelease Освободить семафор  osSemaphoreDelete Удалить семафор  osMessageCreate Инициализация очереди  FreeRTOS и STM32CubeMX  Для того чтобы установить FreeRTOS, необходимо назначить один из таймеров  микроконтроллера (TIM1) на генерацию системных тиков. Конфигурация FreeRTOS производится в разделе Middleware. В этом разделе можно конфигурировать:  • Задачи;  • Очереди;  • Семафоры;  • Таймеры;  • Мьютексы. | Thông tin chung  FreeRTOS là hệ điều hành đa nhiệm thời gian thực dành cho các hệ thống nhúng. Việc chuyển đổi giữa các tác vụ là trách nhiệm của một điều phối viên làm việc trên bộ ngắt hẹn giờ. Tác vụ trông giống như một hàm C thông thường thực hiện một số hành động trong một vòng lặp vô hạn. Đối với STM32, API CMSIS-RTOS được sử dụng.  Ví dụ về tác vụ chuyển đổi trạng thái của đèn LED sau mỗi 500ms:  void Start\_ledTask (đối số void const \*)  {  cho (;;)  {  HAL\_GPIO\_TogglePin (GPIOD, GPIO\_PIN\_13);  osDelay (500);  }  }  Vì các tác vụ chạy không đồng bộ, dữ liệu có thể bị mất khi sử dụng biến được chia sẻ. Hàng đợi được sử dụng để đảm bảo giao tiếp chính xác giữa các tác vụ. Một số tác vụ đặt dữ liệu vào một hàng đợi, những tác vụ khác sẽ đọc từ đó. Điều phối viên giám sát tràn hàng đợi và đọc sau khi kết thúc quá trình ghi. Nếu hàng đợi trống / bị tràn, thì tác vụ đọc / ghi dữ liệu ở đó sẽ được chuyển sang trạng thái CHỜ và bộ điều phối sẽ đưa nó ra khỏi trạng thái này chỉ khi hàng đợi đã sẵn sàng để nhận / gửi dữ liệu.  Có hai loại semaphores: nhị phân và đếm được. Một semaphore nhị phân hoạt động rất giống một lá cờ. Có nghĩa là, khi một sự kiện nhất định tăng một cờ và chương trình sẽ giám sát cờ này trong một vòng lặp. Trong FreeRTOS, bộ điều phối thực hiện chức năng của vòng lặp này. Một semaphore đếm hoạt động theo cùng một cách. Điểm khác biệt duy nhất là trạng thái của anh ta không phải là một cờ đơn lẻ, mà là một loại biến số. Tác vụ đưa ra semaphore làm tăng biến này, và tác vụ đọc nó sẽ giảm biến đó.  osThreadCreate Bắt đầu thực thi chuỗi  osThreadTermina End thực thi chuỗi  osThreadYield Truyền thực thi sang một luồng khác  osThreadGetId Lấy mã định danh chủ đề cho liên kết  trên anh ta  osThreadSetP Priority Thay đổi mức độ ưu tiên của chuỗi  osThreadGetPooter Nhận ưu tiên thực thi hiện tại  lưu lượng  osDelay Chờ trong thời gian được chỉ định  osWait Chờ các sự kiện kiểu Tín hiệu, Tin nhắn, Thư  osTimerCreate Xác định Thuộc tính Gọi lại Bộ hẹn giờ  osTimerStart Bắt đầu hẹn giờ với ấn định thời gian  osSignalSet Đặt cờ tín hiệu luồng  cờ tín hiệu luồng osSignalClear Clear  osMutexCreate Khởi tạo mutex  osMutexWait Nhận mutex hoặc đợi nó  sẽ không có sẵn  osMutexRelease Giải phóng mutex  osMutexDelete Xóa mutex  osSemaphoreCreate Khởi tạo Semaphore  osSemaphoreWait Nhận semaphore hoặc đợi nó  sẽ không có sẵn  osSemaphoreRelease Giải phóng semaphore  osSemaphoreDelete Xóa semaphore  osMessageCreate Khởi tạo hàng đợi  FreeRTOS và STM32CubeMX  Để cài đặt FreeRTOS, bạn phải chỉ định một trong các bộ hẹn giờ  vi điều khiển (TIM1) để tạo ra các tick của hệ thống. FreeRTOS được cấu hình trong phần Middleware. Trong phần này, bạn có thể cấu hình:  • Nhiệm vụ;  • Hàng đợi;  • Semaphores;  • Bộ hẹn giờ;  • Mutexes. |