|  |  |
| --- | --- |
| osThreadTerminate(NULL);  osThreadTerminate(myTask03Handle); | Позволяет **уничтожить задачу**, которая выполняется (c помощью параметра NULL) или ту на которую указывает хэндл . |
| //глобальное объявление  osThreadId myTask03Handle;  void StartTask03(void const \* argument);  //создание задачи  osThreadDef(myTask03, StartTask03, osPriorityIdle, 0, 128);  myTask03Handle = osThreadCreate(osThread(myTask03), NULL);  //описание задачи void StartTask03(void const \* argument) {  …  } | **Создает задачу:**   1. Объявляем   #define osThreadDef(name, thread, priority, instances, stacksz)  Name – имя задачи  Thread – указатель на адрес функции в памяти (название ф-ции)  Priority – начальный приоритет  Instances – количество возможных экземпляров потока (0 – не ограничено)  Stacksz – Размер стека в байтах   1. Создаем экземпляр задачи   osThreadId osThreadCreate (const osThreadDef\_t \*thread\_def, void \*argument)  osThreadDef\_t – имя потока  argument – аргумент |
| osThreadList((unsigned char \*)str\_buf); | Вернет в массив str\_buf **информацию о текущих процессах** Работает только при включенных режимах отладки операционной системы, конкретно  GENERATE\_RUN\_TIME\_STATS  USE\_TRACE\_FACILITY  USE\_STATS\_FORMATTING\_FUNCTIONS  Описание данных: (B : Blocked, R : Ready, D : Deleted, S : Suspended), далее приоритет, далее стек, далее ID задачи |
| uint32\_t osKernelSysTick(void); | Возвращает **количество системных квантов** операционной системы в формате uint32\_t. |
| //глобальное объявление  osMessageQId myQueue01Handle;  // создание очереди  osMessageQDef (myQueue01, 16, uint32\_t);  myQueue01Handle = osMessageCreate(osMessageQ(myQueue01), NULL);  // положить что-то в очередь  osMessagePut(myQueue01Handle, VARIABLE, 50000);  //взять из очереди  osEvent event; //создаем структуру данных которые нужно принять  event = osMessageGet(myQueue01Handle, 100); //забираем данные из структуры  if (event.status == osEventMessage) { //проверяем, не пустая ли структура  do {  printf(" %i", event.value.v); //данные хранятся в event.value.v  event = osMessageGet(myQueue01Handle, 100); // проверяем, есть ли что еще в очереди  } while (event.status == osEventMessage); // проверяем, есть ли что еще в очереди  } | Добавляет **очередь из переменных** |
|  |  |
|  |  |
|  |  |