Inside Int Main

FREE-RTOS-CHEAT-SHEET

Task Priority ↑

Task Creation → xTaskCreate(vBlueLedTask, "BlueLedTask", 100, NULL, 1, NULL)

Func. Declaration after while(1)

- vBlueLedTask ↓ Function Task-Name
- "BlueLedTask" ↓ Task-Name
- 100 ↓ Stack Size
- NULL ↓ Task Parameter
- NULL ↓ Task Handler

Task Priority → 1, 2, --- Higher the no. Higher the priority.
- → Measured Execution by **Task Profiler**
- → Round Robin Scheduling = Tasks with same priority execute for same time as per Time Quanta (SysTick)
- → Lower cannot take Higher.
- → Task-Priority can be read at runtime ⟶ uxTaskPriorityGet Green_handle
- ↳ Using handler, task priority can be changed at Runtime.

Suspending Task → Task can be suspended inside its function → vTaskSuspend(handle); (Self Suspend) NULL ←
- ↳ vTaskResume(red_handle) → For Resuming Suspended Task.

Terminating Task → Not available by default to delete a task, need to enable.
- → To Enable > cmsis.h > Src > freertos.c > freertos.h > FreeRTOSConfig.h
- → #define INCLUDE_vTaskDelete 1
- → vTaskDelete(NULL); → Handler

Task-States → Ready, Running, Blocked, Suspended

Event → Resource or TimeDelay
- → Time Related Event
- → Synchronization of Tasks Event } **Blocked State** of task to make Delays.
  - ↳ vTaskDelay()
  - ↳ vTaskDelay(TickType_t xTicksDelay)
  - ↳ pdMS_TO_TICKS(100)

FreeRTOSConfig.h > Enable vTaskDelay 1

→ Use xTaskGetTickCount() & pdMS_TO_TICKS() in API func. **vTaskDelayUntil**(&xLastWakeTime, xPeriod)

Idle-Task → Task with 0 Priority; Runs by default when no other task running.

Tick Hook function → Called by Kernel during each tick interrupt, Not Recommend
→ Execute within the context of Tick-Interrupt.

Queues → FIFO Buffers (Finite no. of fixed Data Items) → Max length of queue defind.
→ Sender/Receiver Tasks
- → Either Pass by Value or Pass by Reference (Saves memory, for larger data)
- → Task reading from a queue can optionally specify a block Time for its reading.
- → Task writing is placed in Blocked state if queue is full & moves to ready when space available

xQueueSend(), xQueueSentToFront(), xQueueSendToBack(), ① xQueueRecieve() ② xQueueCreate() ③

Queuesets → Allows a Task to Receive data from more than one queue without Task Polling each queue.
- → Enable → configUSE_QUEUE_SETS() > freeRTOSConfig.h
- → Automatically recieve notification, when data is available from a queue.

xQueueCreateSet(), xQueueAddToSet(),
- ↳ Send Data from Task to Task or Task to ISR (Interrupt Service Routine) or btw task & interrupts.

Semaphores → signal/key sent btw tasks
→ Doesnot carry any data.
- → Binary : 1/0
- → Counting: Can be incremented/decremented. Counter indicates [No. of Keys] to [access a resource]
- → xSemaphoreCreateBinary(), xSemaphoreGive(), xSemaphoreGiveFromISR(),

xSemaphoreTake(), xSemaphore CreateCounting(), → Enable configUSE_COUNTING_SEMAPHORES ↓ FreeRTOSConfig.h
- ↳ Counting Events
- ↳ Resource Management

Mutex → Mutual Exclusion (Allows multiple tasks to access a single shared resource but only one at a time.)
→ configUSE_MUTEXES 1

xSemaphoreCreateMutex( )

→ Avoid Poiority Inversion & Poiority Inheritance & DeadLock.

Software Timers → to schedule the execution of a function at a set time in the future or periodically with a fixed frequency.

→ Auto-Reload-Timer (Auto-restart)

→ One-Shot-Timer (will not restart)

→ States (Dormant, Running)

→ xTimerCreate( ), xTimerStart( ), xTimerStop( )

Event-Groups → makes a task or tasks to wait in Blocked state for a combinati of one or more events to occour.

→ Synchomizing Tasks
→ Broadcasting events to multiple Tasks.

→ Replace multiple binary semaphores with single event group.

→ Its a ~~group~~ set of event flags.

→ 1/0, indicating an event has occoured or not. (Event Bits)

→ configUSE_16_BIT_TICKS = 1, Each event goup contains 8 useable event Bits.
↳ 0, 24 useable events.

→ EventGroupHandle_t xEventGroup; xEventGroup = xEventGroupCreate( );
xEventGroupSetBits(xEventGroup, TASK1_BIT), xEventGroupSync( )

Task Notifications Allow tasks to directly communicate directly without needing Comm. objects such as queues, semaphores & event groups!

→ Cannot be sent to multiple Tasks.

→ Cannot be used for Data Exchange & sent events.

→ ulTaskNotify( ), vTaskNotifyGiveFromISR( )

FreeRTOS Scheduler → Software routine that decides whose task state needs to be changed from Ready → Running.

→ Fixed Priority

→ Pre-emptive (First Task runs only ~~as Tasks with Same~~ if there is no blocked state)

→ Time-Slicing (configUSE_TIME_SLICING 0), vTaskDelay (_50 ms)