Assignment 1

Spring 2024

Name: Mustafa Sayed Al-Said Mohamed

ID: 1900361 Department: CSE Section: 3

Course Code: CSE411s

Course: Real-Time and Embedded Systems Design

1. Tick Count Functions:

xTaskGetTickCount():

This function returns the number of ticks since the FreeRTOS scheduler was started. It provides a crude time measurement useful for tasks that need to measure time intervals or timeouts.

Implementation: This function returns the current tick count since the start of the RTOS tick count by directly accessing a variable that holds the tick count. The variable is incremented by the tick interrupt handler.

Usage Example:

```
void taskFunction(void *params) {
    TickType_t lastWakeTime = xTaskGetTickCount();
    // Perform task actions
}
```

xTaskGetTickCountFromISR():

Similar to xTaskGetTickCount(), but called from an interrupt service routine (ISR), it provides the tick count, time stamp and duration within interrupts.

Implementation: it involves disabling interrupts temporarily to ensure singular access to tick count variable.

Usage Example:

```
void ISR_Handler() {
    TickType_t tickCount = xTaskGetTickCountFromISR();
    // ISR code
}
```

2. Delay Functions:

vTaskDelay():

This function suspends the calling task for a specified number of ticks. It allows tasks to wait for a certain amount of time without using CPU cycles.

Implementation: vTaskDelay puts a task into the blocked state until the delay period ends and other tasks can run at that time.

Usage Example:

```
vTaskDelay(pdMS_TO_TICKS(1000)); // Delay for 1000 milliseconds
```

vTaskDelayUntil():

This function delays a task until a specified absolute time, instead of a relative time like vTaskDelay.It's useful for tasks that need to run periodically at fixed intervals.

Implementation: It calculates the time when the task should wake up and then blocks the task until that time is reached.

Usage Example:

```
TickType_t xLastWakeTime = xTaskGetTickCount();
const TickType_t xFrequency = pdMS_TO_TICKS(1000); // Execute every 1000ms
for (;;) {
    vTaskDelayUntil(&xLastWakeTime, xFrequency);
    // Task code...
}
```

3. Software timer Functions:

xTimerCreate():

This function creates a software timer. Software timers are a way to execute code at a specificed time or after a specified interval.

Implementation: it allocates memory for the timer control structure and initializes it, parameters include a name for the timer, the timer period, whether it auto-reloads, a timer ID, and a callback function that executes when the timer expires.

Usage Example:

```
TimerHandle_t xTimer = xTimerCreate("Timer", 1000 / portTICK_PERIOD_MS, pdTRUE, (void *) 0,
TimerCallback);
```

xTimerStart():

This function starts a software timer that we created.

Implementation: it sets up the timer's state and starts counting according to its period, and can be called from a task or an ISR context.

Usage Example:

```
xTimerStart(xTimer, 0);
```

xTimerStop():

This function stops a running software timer.

Implementation: It stops the timer's counting.

Usage Example:

```
xTimerStop(xTimer, 0);
```

xTimerReset():

This function Resets a software timer to its initial state.

Implementation: Resets the timer's counting to zero or the timer's period, effectively restarting the timer.

Usage Example:

```
xTimerReset(xTimer, 0);
```

xTimerDelete():

This function deletes a software timer, freeing up any allocated memory.
Implementation: it deallocates memory associated with the timer.
Usage Example:

```
xTimerDelete(xTimer, 0);
```

Recording link: