FreeRTOSのAPIと似ているT-Kenrelのサービスコール

既にT-kernelやμT-Kernelを知っている人向けに、FreeRTOSのAPIと似ているサービスコールを対比する。最近、よく見かけるようになってきたFreeRTOSを使ってみる時の参考になればと思い、備忘録を残す。

特に下記リンクの記事がとても参考になった。
μT-Kernel 3.0とFreeRTOS - UCT

但し、どちらもプリエンティブ[1)]なマルチタスクRTOSだが、FreeRTOSとT-Kenrelでは似ているところもあるが細かな差異やコンセプトの違いもあり、単純に右から左へとはいかないので注意は必要である。

APIの対比

1. FreeRTOSのタスク優先度は0が最低で、(configMAX_PRIORITIES-1)が最高となっている。T-Kernelでは1が最高で、CNF_MAX_TSKPRI が最低である。
2. FreeRTOSではタスクから呼び出すAPIと割込みハンドラ(ISR)から呼び出すAPIが別になっている。ISRから呼び出すAPIは FromISR() という名称になっており、こちらを呼び出さなければならない。（T-Kernelでは共通の関数名）
3. FreeRTOSでは、各種の識別子はポインタでの扱いとなっている。（T-KernelではIDは整数値）
4. FreeRTOSのAPIでは時間はティック数で処理されているので、取得される経過時間もティック数となる。このため、アプリケーションでは必要に応じてミリ秒に変換(pdMS_TO_TICKS関数)する必要がある。（T-Kernelではミリ秒と規定される）

※FreeRTOSでの命名規則

先頭文字	説明
pc	戻り値が char *
pv	戻り値が void *
v	戻り値が void
ux	戻り値がUBaseType_t
x	戻り値がBaseType_t

タスク制御

項目	FreeRTOSのAPI	T-KernelのAPI	備考
タスクの生成	xTaskCreateStatic()	tk_cre_tsk(), tk_sta_tsk()	FreeRTOSでは生成と同時に起動される
タスクの終了	vTaskDelete()	tk_exd_tsk()	
タスクの実行状態を確認	eTaskGetState	tk_ref_tsk()	T-Kernelではタスクの各種情報を一括で取得
タスク識別子の取得	xTaskGetCurrentTaskHandle()	tk_get_tid()	FreeRTOSでは識別子はポインタ
タスクの待ち	ulTaskNotifyTake()	tk_slp_tsk(), tk_can_wup()	
タスクの待ち解除	xTaskNotifyGive()	tk_wup_tsk()	

項目	FreeRTOSのAPI	T-KernelのAPI	備考
タスクの遅延	vTaskDelay()	tk_dly_tsk()	

- T-KernelではタスクIDは整数値だが、FreeRTOSではタスクハンドルはTCB[2]構造体へのポインタとなっている。

ミューテックス

FreeRTOSでは(通常の)ミューテックスと再帰ミューテックスが利用可能である。

項目	FreeRTOSのAPI	T-KernelのAPI	備考
ミューテックスの生成	xSemaphoreCreateMutex(), xSemaphoreCreateRecursiveMutex()	tk_cre_mtx()	
ミューテックスの削除	vSemaphoreDelete()	tk_del_mtx()	
ミューテックスのロック	xSemaphoreTake(), xSemaphoreTakeRecursive()	tk_loc_mtx()	
ミューテックスのアンロック	xSemaphoreGive(), xSemaphoreGiveRecursive()	tk_unl_mtx()	

セマフォ

FreeRTOSではカウンティングセマフォであっても操作できる資源数が1に制限されている。

項目	FreeRTOSのAPI	T-KernelのAPI	備考
セマフォの生成	xSemaphoreCreateCounting()	tk_cre_sem()	
セマフォの削除	vSemaphoreDelete()	tk_del_sem()	
セマフォ資源の獲得	xSemaphoreTake(), xSemaphoreTakeFromISR()	tk_wai_sem()	
セマフォ資源の解放	xSemaphoreGive(), xSemaphoreGiveFromISR(), portYIELD_FROM_ISR()	tk_sig_sem()	

また、FreeRTOSでは待ちが発生しない場合はタスク独立部からもセマフォ資源獲得を呼び出せる。専用のAPI xSemaphoreGiveFromISR()が用意されていて、資源があれば獲得することもできる(資源がなければ即時エラー終了する)。

更に注意する点があり、FreeRTOSでは割込みハンドラでセマフォを制御した結果としてタスクの切り換えが必要になった場合、開発者が明示的にportYIELD_FROM_ISR()を呼び出さなければならない。

イベントフラグ

FreeRTOSで、イベントフラグで使用可能なビット数が8ビットか24ビット。configUSE_16_BIT_TICKSの定義に依存し 16ビット or 32ビットと定義（（T-Kernelではunsigned int型で32ビット全てが使用可能）

項目	FreeRTOSのAPI	T-KernelのAPI	備考
イベントフラグの生成	xEventGroupCreate()	tk_cre_flg()	
イベントフラグの削除	vEventGroupDelete()	tk_del_flg()	
イベントフラグのセット	xEventGroupSetBits(), xEventGroupSetBitsFromISR()	tk_set_flg()	
イベントフラグのクリア	xEventGroupClearBits(), xEventGroupClearBitsFromISR()	tk_clr_flg()	FreeRTOSでは1のビットがクリア
イベントフラグの確認	xEventGroupGetBits()	tk_ref_flg()	
イベントフラグ待ち	xEventGroupWaitBits()	tk_wai_flg()	

FreeRTOSでは戻値がビットパターンになっているのは注意が必要である。このため、タイムアウトが発生したかどうかは戻値をビットパターンと比較して検証しなければならない。

また、セマフォでも説明したとおり、FreeRTOSでは割込みハンドラでセマフォを制御した結果としてタスクの切り換えが必要になった場合、開発者が明示的に portYIELD_FROM_ISR() を呼び出さなければならない。この点も注意が必要である。

タイマ

タイマハンドラはT-Kernelではタスク独立部(割込みハンドラ)として実行されるが、FreeRTOSではタイマサービスタスクは他のタスクと同様に扱われるので、優先度によっては(時間が来ても)必ずしもタイマが動作するとは限らないので注意が必要である。

項目	FreeRTOSのAPI	T-KernelのAPI	備考
タイマの生成	xTimerCreate()	tk_cre_alm(), tk_cre_cyc()	
タイマの削除	xTimerDelete()	tk_del_alm(), tk_del_cyc()	
タイマの開始	xTimerChangePeriod(), xTimerStart()	tk_sta_alm(), tk_sta_cyc()	
タイマの停止	xTimerStop()	tk_stp_alm(), tk_stp_cyc()	
タイマの起動状態の確認	xTimerIsTimerActive()	tk_ref_alm(), tk_ref_cyc()	
システム稼働時間の取得	xTaskGetTickCount()	tk_get_otm()	FreeRTOSでは取得される経過時間はティック数

注意点&留意点

タスク優先度の違い

FrreRTOSのタスク優先度は、数値が小さい「0」が最低で、数値の大きい「configMAX_PRIORITIES - 1」が最高。この最高の優先度値はFreeRTOSConfig.hにて定義。

T-KernelやiTronの優先度は、数値が小さい方が優先度が高い仕様となっている。これは設計ポリシーの違いなのでどちらが優れているとか良いとかは言えないが、とにかく反対で違うので注意が必要。（ちなみにLinuxのpthreadも数値が大きい方が優先度が高い仕様となっている。）
あと、割り込みコントローラの割り込みレベルの優先度とも異なる場合があるので注意が必要。

🔧 Fix Me!

関連記事

- FreeRTOS
- T-Kernel

参考

1. µT-Kernel 3.0とFreeRTOS - UCT
2. FreeRTOSポーティングで学んだ事について
3. Chapter 4 割り込み管理(Interrupt Management) (FreeRTOS チュートリアル日本語訳)
4. FreeRTOS の概要（自動翻訳 無保証）
5. FreeRTOS ドキュメント
6. API Reference - FreeRTOS

FreeRTOS
[1]
割込み駆動で実行するOS
[2]
Task Control Block

From:
https://yokoweb.net/dokuwiki/ - **新石器Wiki**

Permanent link:
https://yokoweb.net/dokuwiki/technology/rtos/freertos/freertos_api-vs-tkernel_api?rev=1721008855

Last update: **2024/07/15 11:00**