第3章 時間管理の単位チック・タイムと割り込みの設定

システム・タイマとタスクの時間待ち機能

豊山 祐一

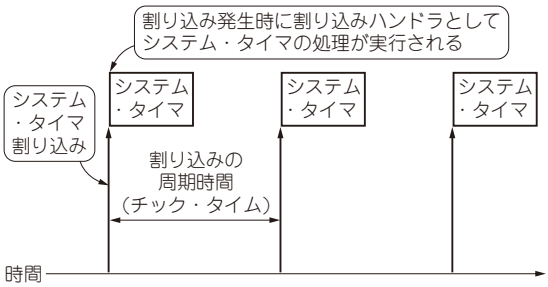

図1 システム・タイマと時間管理

リスト1 SysTickタイマの設定(boot¥reset_hdr.h)

```
static void init_systim(void)
{
    /* SysTick動作停止  << 変更点*/
    out_w(SYST_CSR ,SYST_CSR_CLKSOURCE |
                    SYST_CSR_TICKINT);
    /* リロード値設定 */
    out_w(SYST_RVR, (TIMER_PERIOD*TMCLK_KHz)-1);
    /* カウント値設定 */
    out_w(SYST_CVR, (TIMER_PERIOD*TMCLK_KHz)-1);
    /* SysTick動作開始 << 変更点 */
    out_w(SYST_CSR, SYST_CSR_CLKSOURCE |
            SYST_CSR_TICKINT | SYST_CSR_ENABLE);
}
```

RTOSのさまざまな時間管理の機能を実現するのがシステム・タイマです.

本章ではTry Kernelにシステム・タイマを実装します. そしてシステム・タイマを使って, タスクを指定した時間, 待ち状態とするAPIを実現します.

システム・タイマと時間管理の関係

● 時間管理のためにシステム・タイマを使う

システム・タイマはハードウェアのタイマを使って時間を計測し, OSの時間管理の処理を実行します. Try Kernelでは第2部で説明したように, Cortex-Mに内蔵のSysTickタイマを使用します.

SysTickタイマから一定の時間間隔(チック・タイム)で割り込みを発生させます. この割り込みをシステム・タイマ割り込みと呼びます.

▶時間管理の単位:チック・タイム

システム・タイマ割り込みにより実行される割り込みハンドラが, システム・タイマのプログラムの実体です. Try Kernelの時間管理の処理は, チック・タイムの周期で実行されます(**図1**).

チック・タイムはTry Kernelの時間管理の単位ですので, 時間管理の精度にそのまま反映されます. チック・タイムを短くすれば精度は上がりますが, 一方で割り込み処理の頻度も増えてCPUの処理時間を消費します. 両者の兼ね合いを考えて適切な時間に設定する必要があります.

Try Kernelのチック・タイムは10msを標準としますが, include¥sysdef.hファイルに以下のように記述されているTIMER_PERIODの値の定義を変えることによって変更が可能です.

```
#define TIMER_PERIOD  (10)
```

● システム・タイマ割り込みの設定

チック・タイムの周期でシステム・タイマ割り込みが発生するようにSysTickタイマを設定します.

既に第2部で作成したリセット・ハンドラでSysTickタイマの設定を行っていますが, 時間を計測するだけで割り込みは発生しない設定がされています. そこで, **リスト1**に示すように割り込み発生を有効とする設定に変更します

次に**リスト2**に示すように, 例外ベクタ・テーブルにシステム・タイマの割り込みハンドラを登録します. 割り込みハンドラの関数名はsystimer_handlerとしました.

タスクの時間待ち機能の動作

● タスクの待ち状態が必要になる

システム・タイマを使って, 指定した時間が経過す