第4章 タスクの作り方からバッファやセマフォの使い方まで

動かしながら学ぶ！ FreeRTOSプログラミング

後閑 哲也

1 タスクの作り方

リスト1 タスクの基本形

```
void TaskName(void *pvParameters)
{
    //初期化部

    while(1)
    {
        //タスク機能実行部
        //この中でAPI関数によりFreeRTOSを使う
    }
}
```

（a）タスクの基本形

```
void TASK1_Initialize ( void )
{
    task1Data.state = TASK1_STATE_INIT;

}
/*********************************/
void TASK1_Tasks ( void )
{
    switch ( task1Data.state )
    {
        case TASK1_STATE_INIT:
        {
            bool appInitialized = true;
            if (appInitialized) {

                task1Data.state = TASK1_STATE_SERVICE_TASKS;
            }
            break;
        }
        case TASK1_STATE_SERVICE_TASKS:
        {

            break;
        }
        /* ここにアプリステートを追加する */
        default:
        {
            break;
        }
    }
}
```

初期化部

永久ループ部

（b）Harmony v3で生成されるタスクのひな形

● タスクの基本形…初期化部と機能実行部

リスト1(a)に示すのは，FreeRTOSの管理下で動作するタスクの基本形です．

このように，タスク関数はちょうどC言語プログラムのmain関数と同じ構成です．FreeRTOSの内部構成は，C言語のmain関数を複数並べておき，これらを一定間隔，あるいはイベント発生により切り替えて実行させるものだと考えると分かりやすいと思います．

リスト1(b)に示すのは，Harmony v3で自動生成されるタスクのひな形です．**リスト1**(a)の基本形に対応した形式のステート関数となっています．TASK1_Tasks()関数が，FreeRTOSのスケジューラから実行可能になるたびに呼び出されて実行されます．一番上のTASK1_Initialize()関数は，システム初期化のときに1回だけ呼び出される関数で，ステートを初期ステートにしているだけです．タスクが2回目以降に呼び出されたときは，常にTASK1_STATE_SERVICE_TASKSのステートになるので，ここが永久ループに相当する部分になります．そのため，タスクの機能はここに記述します．ステートの状態を追加して，その次のステートという形でタスク機能を追加することもできます．

タスクは次のような形式で作成します．

▶ (1) 初期化部の記述

タスク関数内部では，ローカル変数があれば初期化部に宣言定義を記述します．この変数は，通常のローカル変数と全く同じ扱いです．タスク起動時に1回だけ実行する処理があれば，初期化部に記述します．ここには，主にタスク内の変数初期化や，内蔵モジュールなどの初期化に関する記述をします．

▶ (2) 機能実行部は永久ループとして記述する

組み込みシステムでは，通常アプリケーションは常に実行状態となっていて，イベントを待って処理するようにします．FreeRTOSのタスクも，常時動作状態としてイベントを待つようにして，通常は永久ループで繰り返すような形式で記述します．

この繰り返し中にTickの割り込みでタスクが切り替わり，実行を途中で中断して別のタスクに実行が移