第4章　ビット・データ，イベント・フラグ，データ・キュー

タスクやハンドラ間でデータをやりとりする方法

関連データ

杉本 明加

前章では処理を行う主体となる，タスクやハンドラを使った処理方法を紹介した．本章では，処理を実装したタスクやハンドラ間でデータをやりとりすることを想定し，どのようにリアルタイムOSの機能を活用するかを紹介する．（筆者）

OSを使わないメイン・ルーチンだけのプログラムの場合，処理間のデータのやりとりは関数の返り値やポインタ参照を介するのが一般的です．しかし，マルチタスク環境で複数のタスクやハンドラに処理がまたがった場合，この方法は適用できず，ほかの方法を用いることになります．

OSを使わない場合でも，実際にはメイン・ルーチンと割り込みハンドラ間では同様のことが起きており，多くの場合はグローバル変数を介してやりとりしています．リアルタイムOS（RTOS）環境下でもグローバル変数を使うことは可能ですが，たいていのRTOSは使い勝手のよい機能を提供しているので活用しない手はないでしょう（これはRTOSだけでなく汎用OSでも同じ）．

TOPPERS/ASPでは同期・通信機能というカテゴリで何種類ものデータ送受信方法を提供します．それぞれに特性があり，扱うデータによって使い分ければアプリケーション開発の負荷を軽減できます．

なお，本章からは擬似コードをしばしば用います．擬似コードというのは，制御文（ifやwhileなど）と自然言語を使ってプログラム・コードのあらましを示す手法です．

1．ビット・データを扱う

組み込みシステムのプログラミングでは，ビット・データが頻繁に用いられます（**図1**）．メモリ制約の厳しさから，フラグのような1ビットで表現可能なデータはパッキングされて汎用データ型に格納されることが多々あります．典型的なパターンとしては，一つの変数に対してイベントごとにビット・マスクを割り付け，イベントの発生有無を表現する使い方があります（**リスト1**）．

同じ処理を自分でも実装できますが，割り込みを禁止してしまい，応答性が低下することがあります．また，デー

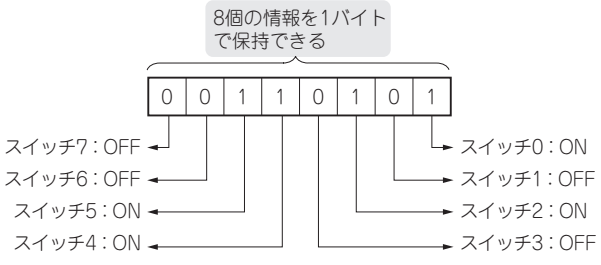

図1　ビット表現により情報の保持
スイッチ0～7の8個の状態を保持．0の場合はOFF，1の場合はON．

▶リスト1
変数を使ったイベント処理

```
enum
{
    EVENT0 = 1 ,
    EVENT1 = 2 ,
        :
};

static uint_t event = 0;

void func1(void)
{
    if((event & EVENT0) != 0)
    {
        /* EVENT0 が発生したときの処理*/
    }

    if((event & EVENT1) != 0)
    {
        /* EVENT1 が発生したときの処理*/
    }
                :
}

void func2(void)
{
    if(/* EVENT0 の発生条件*/)
    {
        event |= EVENT0;
    }
}

void func3(void)
{
    if(/* EVENT1 の発生条件*/)
    {
        event |= EVENT1;
    }
}
```

（ビット・マスクの割り付け）