24 STEP

割込み

H8 マイコン実験キットを使ったマイコン制御実習もいよいよ大詰めです。
STEP24 からは、割込みについて学習します。
この割込みが最終課題となります。
もう一踏ん張りです。頑張りましょう。

今までのプログラムでは、条件分岐を行うとき if 文で条件の確認を行ってきました。しかし、動作によっては反応が遅れてしまう場合があります。試しに、課題 24-1 を考えてみてください。

課題　24-1

常時、ステッピングモータを 1 ステップ 1 秒刻みで回転させる。
SW1 を押すと、LED 点灯し、SW2 を押すと LED が消灯する。

STEP 24 割込み

課題を解決する簡単なフローチャートは以下のようになります。プログラミングして動作を確認してみましょう。

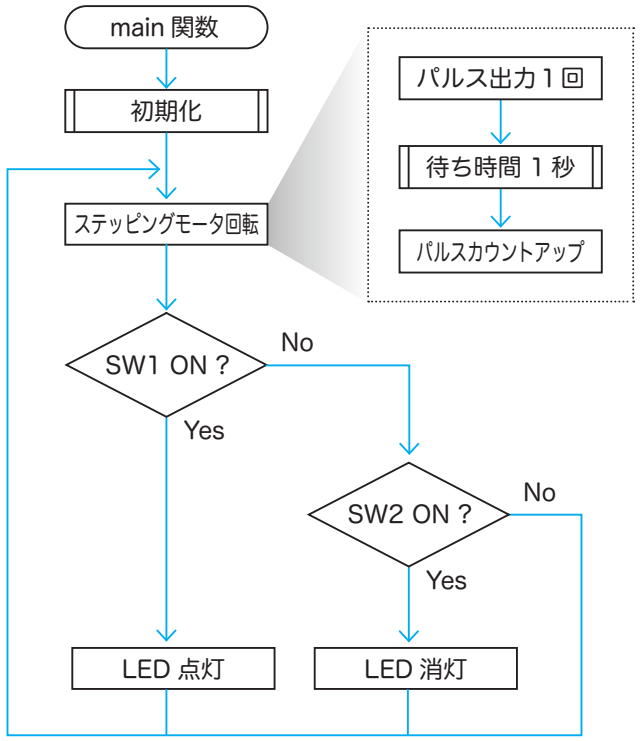

図 24-1　フローチャート

プログラム例　24-1

```
#include <3052f.h>                  // 3052F 固有の定数
#define ST_MOTOR_MS 1000            // 回転インターバル時間
#define SW1_ON (P8.DR.BIT.B0 == 0) // SW1 が押された時
#define SW2_ON (P8.DR.BIT.B1 == 0) // SW2 が押された時

// ステッピングモータに送るパルスパターンの配列
const int one_phase_excitation[4] = {0x40, 0x20, 0x10, 0x08};
```

中略

```
/*
 *  main 関数
 */
int main(void)
{
    int index = 3; // パルスパターン配列の添え字
    initIO();      // 初期化関数の呼び出し

    while (1)
    {
        // ステッピングモータ回転
        PA.DR.BYTE = one_phase_excitation[index];
        waitMs(ST_MOTOR_MS);  // 回転インターバル
        index++;              // 右回転
        if (index > 3)        // index 値の補正
            index = 0;

        if (SW1_ON) // SW1 を押すと LED 点灯
            P4.DR.BYTE = 0xFF;

        if (SW2_ON) // SW2 を押すと LED 消灯
            P4.DR.BYTE = 0x00;
    }

    return 0;
}
```

動作させてみると、SW を押してもすぐに反応しないことがあったでしょう。
ステッピングモータが回転している「待ち時間」の間は SW の状態を検出することができず、検出するタイミングが 1 秒に 1 回しかないためです。ですから、SW を押しっぱなしにすれば、待ち時間後に反応します。
このようなループによる確認動作をポーリングといいます。ポーリングでも「待ち時間」ループの中に SW の条件分岐を設ければ反応を良くすることもできます。しかし、処理によって待ち時間の長さが変わってしまうので時間調整が面倒です。そもそもステッピングモータの待ち時間をループで作っていることが間違いなのですが、これはポーリングの問題を明らかにするための課題ですのでご了承ください。

STEP 24 割込み

待ちループなどの処理中にかかわらず、いつでも SW に反応するようにするには「割込み」を使う方法があります。割込みとは、ある要因が発生したときに、実行中の処理 (主処理) を一時中断して、別の処理 (割込み発生による処理) を行い、その処理が終了すると元々実行していた処理 (主処理) を再開させることを言います。

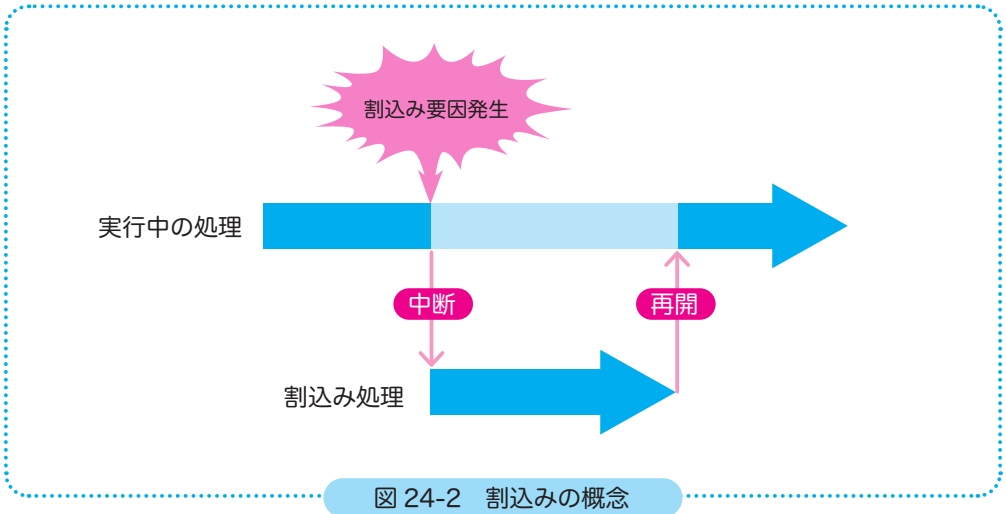

図 24-2　割込みの概念

また、割込み発生による処理を割込みハンドラや割込みルーチン、割込みサービスルーチンなどと言います。
割込みが発生すると、割込みハンドラへジャンプし、割込みハンドラの処理が終わると、中断していた主処理を再開します。

これから割込みを使った動作を考えていくのですが、一口に割込みと言っても下記の種類があります。
・外部割込み：外部から何らかの情報が与えられた時の割込み（キー入力割込み）
・内部割込み：内部から何らかの情報が与えられた時の割込み（タイマ割込み、オーバフロー割込み）

STEP23 のカウンタ動作も割込みの一種です。
本テキストでは STEP25 でキー入力割込み、STEP26 でタイマ割込みについて学習していきます。

割込みは、「ハードウェア的な仕組みによって、特別な関数が呼び出される」ことで実現されます。使用できる割込みの種類とハンドラについては、表 24-1 を参照してください。一般には、イベントは 1 種類だけとは限りませんし、必要な処理はイベントによって異なりますから、それぞれのイベント毎に個別にハンドラを定義します。また、割込み要因は個別に許可・禁止を設定することができます。要因が「禁止」に設定されていると、仮に当該イベントが発生しても割込み要求は発生せず、CPU にイベントが通知されることはありません。（当然ハンドラも呼び出されません）ハンドラの定義にはいくつかの約束事がありますが、このおかげで、発生したイベントに対応するハンドラが自動的に呼び出されます。

名前	ハードウェア	関数名	要因（イベント）	備考
IRQx	外部端子	IRQx_INT	IRQx 端子の状態	x は 0 ～ 5 までの整数
WOVI	ウォッチドッグタイマ WDT	WOVI_INT	インターバルタイマ	
CMI	リフレッシュコントローラ	CMI_INT	コンペアマッチ	
IMIAx	インテグレーテッドタイマユニット ITU	IMIAx_INT	コンペアマッチ / インプットキャプチャ A x	x は 0 ～ 4 までの整数
IMIBx		IMIBx_INT	コンペアマッチ / インプットキャプチャ B x	
OVIx		OVIx_INT	オーバフロー x	
DENDxA	DMA コントローラ DMAC	DENDxA_INT	転送終了	x は 0 ～ 1 までの整数
DENDxB		DENDxB_INT	転送終了	
ERIx	シリアルコミュニケーションインタフェース SCI	ERIx_INT	受信エラー	x は 0 ～ 1 までの整数
RXIx		RXIx_INT	受信データフル	
TXIx		TXIx_INT	送信データエンプティ	
TEIx		TEIx_INT	送信終了	
ADI	A / D コンバータ	ADI_INT	A / D エンド	

表 24-1　割込み要因とハンドラ定義

※ x は整数で、それぞれチャネルを示す。
　例えば、IRQx は、IRQ0 ～ IRQ5 まで 6 本の要因があることを示している。