25 STEP

キー入力割込み

STEP25 では、外部割込みの 1 つであるキー入力割込みについて学習していきます。
STEP24 の課題をキー入力割込みを用いて組み替えましょう。

STEP24 でも説明したように、外部割込みとは外部から何らかの情報が与えられた時に発生する割込みのことです。キー入力割込み発生から終了までの流れは、このようになります。割込みの条件が SW の入力になるということです。

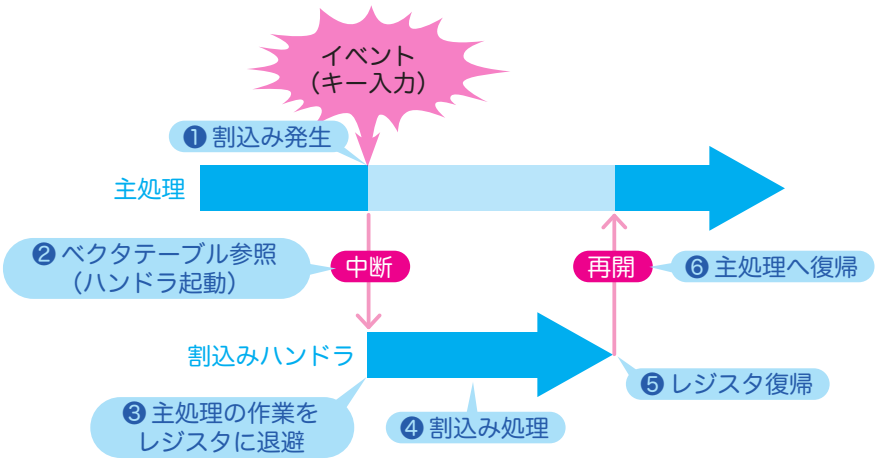

図 25-1　割込み発生による処理の流れ

H8/3052F には、割込みコントローラという回路が搭載されています。割込みの種類は、内部割込みが 30 要因、外部割込みが 7 端子あります。外部割込みのうち 1 本は「NMI 割込み」で本キットでは「NMI スイッチ」に配線済みです。
残りの 6 本（IRQ_0 ～ IRQ_5）をキー入力割込みに使うことができ、今回は IRQ_0 と IRQ_1 を使用してキー入力割込みを行うことにします。

STEP 25 キー入力割込み

25.1 IRQ 端子

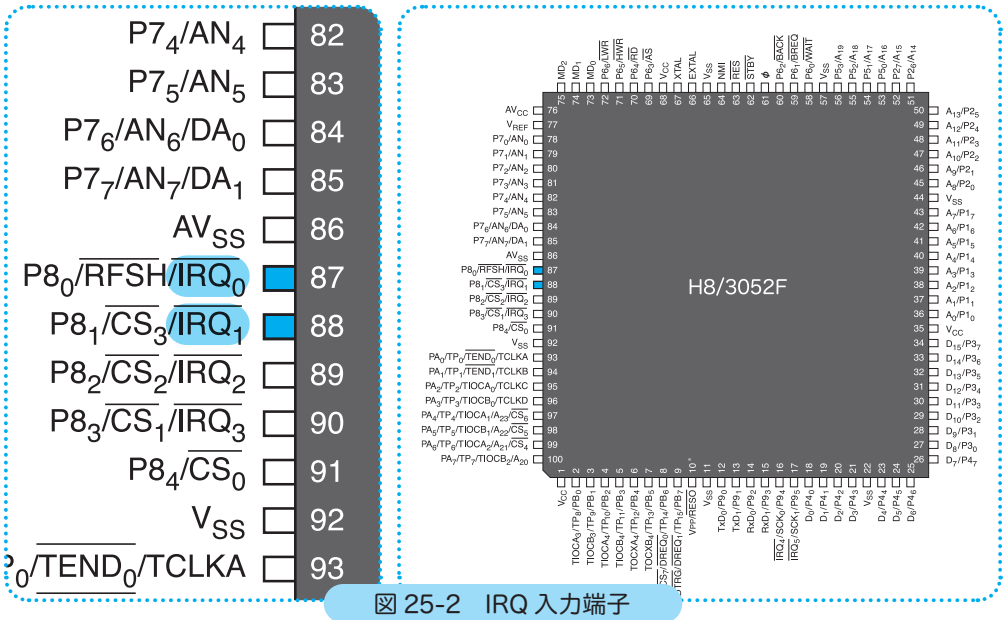

図 25-2 IRQ 入力端子

$\overline{IRQ_0}$ と $\overline{IRQ_1}$ はポート 8 の bit0 と bit1 に割り当てられています。P8$_0$ と P8$_1$ は SW1 と SW2 に配線済みなので、ここまま外部割込みスイッチとして利用できます。

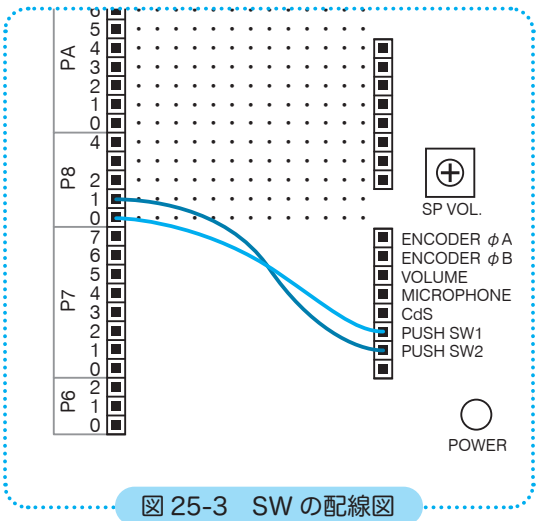

図 25-3 SW の配線図

25.2　IRQ イネーブルレジスタ【 IER 】

IER（IRQ イネーブルレジスタ）は、IRQ_0 ～ IRQ_5 の割込み許可と禁止を設定するレジスタです。
各ビットは各端子に対応しています。例えば、bit0 の IRQ0E で、IRQ_0 端子の設定を行います。

ビット：	7	6	5	4	3	2	1	0
	—	—	IRQ5E	IRQ4E	IRQ3E	IRQ2E	IRQ1E	IRQ0E
初期値：	0	0	0	0	0	0	0	0
R/W：	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

図 25-4　IRQ イネーブルレジスタ

【 bit 5 ～ 0　IRQnE 】IRQn イネーブル

IRQ_5 ～ IRQ_0 の割込みを許可、禁止するかを設定します。1 なら許可、0 なら禁止です。
今回は、IRQ_0 と IRQ_1 を使用するので IRQ0E と IRQ1E に 1 を書き込みます。

0	IRQn の割込みを禁止（初期値）
1	IRQn の割込みを許可

25.3 IRQ センスコントロールレジスタ【 ISCR 】

ISCR（IRQ センスコントロールレジスタ）は、IRQ_5 ～ IRQ_0 の割込み発生の要求条件を設定します。各ビットは各端子に対応しています。bit0 の IRQ0SC で、IRQ_0 端子の設定を行います。

ビット：	7	6	5	4	3	2	1	0
	—	—	IRQ5SC	IRQ4SC	IRQ3SC	IRQ2SC	IRQ1SC	IRQ0SC
初期値：	0	0	0	0	0	0	0	0
R/W：	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

図 25-5 IRQ センスコントロールレジスタ

【 bit 5 ～ 0 IRQnSC 】IRQn センスコントロール

IRQ_5 ～ IRQ_0 割込みを、各端子入力の L レベルで要求するのか、立下りエッジで要求するのかを選択します。0 なら L レベルで割込みを要求し、1 なら立下りエッジで割込みを要求します。今回は、立下りエッジで割込みを要求するよう IRQ0SC に 1 を書き込みます。

0	IRQn の L レベルで割込み要求
1	IRQn の立下りエッジで割込み要求

L レベルで要求　　：入力が L の間、繰り返し割込み
立下りエッジで要求　：入力信号が立下がったときだけ割込み

25.4　IRQ ステータスレジスタ【 ISR 】

ISR（IRQ ステータスレジスタ）は、IRQ_5 ～ IRQ_0 の割込み発生の要求のステータスを表示するスレジスタです。各ビットは各端子に対応しています。bit0 の IRQ0F で、IRQ_0 端子の確認を行います。

ビット：	7	6	5	4	3	2	1	0
	—	—	IRQ5F	IRQ4F	IRQ3F	IRQ2F	IRQ1F	IRQ0F
初期値：	0	0	0	0	0	0	0	0
R/W：	—	—	R/(W)※	R/(W)※	R/(W)※	R/(W)※	R/(W)※	R/(W)※

※フラグをクリアするための 0 ライトのみ可能。

図 25-6　IRQ ステータスレジスタ

【 bit 5 ～ 0　IRQnF 】IRQn フラグ

IRQSC が 1 の時、IRQ 端子に立下りエッジが発生すると 1 になり、割込みが開始されます。割込み要求が発生すると、1 にセットされます。

0	〔クリア条件〕 (1) IRQnF ＝ 1 の状態で IRQnF フラグをリードした後、IRQnF フラグに 0 をライトしたとき (2) IRQnSC ＝ 0、IRQ n 入力が High レベルの状態で割込み例外処理を実行したとき (3) IRQnSC ＝ 1 の状態で IRQn 割込み例外処理を実行したとき
1	〔セット条件〕 (1) IRQnSC ＝ 0 の状態で IRQ n 入力が Low レベルになったとき (2) IRQnSC ＝ 1 の状態で IRQ n 入力に立下りエッジが発生したとき

STEP 25 キー入力割込み

では、フローチャートを確認し、プログラム 25-1 を作成していきましょう。
P80 と P81 は外部割込み端子の IRQ0、IRQ1 として使うので、「初期化」で SW 用の入力設定は不要です。

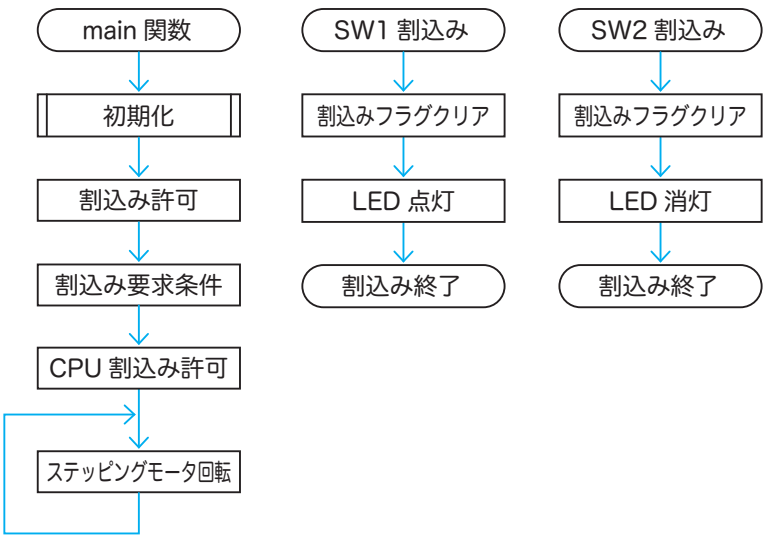

図 25-7　フローチャート

プログラム例　25-1

```
06  #define IRQ0_INT_HANDLER   // IRQ0 割込みを使用することを宣言
07  #define IRQ1_INT_HANDLER   // IRQ1 割込みを使用することを宣言
08  #define ST_MOTOR_MS 1000   // 回転インターバル時間
09  #include <3052f.h>         // 3052F 固有の定数
10  #include <interrupt.h>     // 割込みベクタテーブルの定義

    中略

42  /*
43   *  IRQ0 割込みハンドラ
44   */
45  void __attribute__((interrupt_handler))
46  IRQ0_INT(void)
47  {
48      if (INTC.ISR.BIT.IRQ0F)      // 割込みフラグがセットされていれば
49          INTC.ISR.BIT.IRQ0F = 0;  //（リードした後）割込みフラグクリア
50      P4.DR.BYTE = 0xFF;
51  }
52
53  /*
54   *  IRQ1 割込みハンドラ
55   */
56  void __attribute__((interrupt_handler))
57  IRQ1_INT(void)
58  {
59      if (INTC.ISR.BIT.IRQ1F)      // 割込みフラグがセットされていれば
60          INTC.ISR.BIT.IRQ1F = 0;  //（リードした後）割込みフラグクリア
61      P4.DR.BYTE = 0x00;
62  }
63
64  /*
65   *  main 関数
66   */
67  int main(void)
68  {
69      initIO();       // 初期化関数の呼び出し
70      int index = 3;  // パルスパターン配列の添え字
71
72      INTC.IER.BYTE = 0x03;      // 割込み許可 0000 0011
73      INTC.ISCR.BYTE = 0x03;     // 割込み要求条件 0000 0011
74      asm("andc.b #0x7f,ccr");   // CPU の割込み許可
75
76      while (1)
77      {
78          // ステッピングモータ回転
79          PA.DR.BYTE = one_phase_excitation[index];
80          waitMs(ST_MOTOR_MS);  // 回転インターバル
81          index++;              // 右回転
82          if (index > 3)        // index 値の補正
83              index = 0;
84      }
85
86      return 0;
87  }
```

STEP 25 キー入力割込み

割込み設定をした上で、割込みの要因となるイベントが発生すると、割込みハンドラという特別な関数が呼び出されます。割込みハンドラの記述方法を説明していきます。

```
10  #include <interrupt.h>      // 割込みベクタテーブルの定義
```

interrupt.h には、ベクタテーブルの定義やハンドラのプロトタイプ宣言が記述されていて、簡単に割込みを使ったプログラムが作成できるようになっています。

```
06  #define IRQ0_INT_HANDLER    // IRQ0 割込みを使用することを宣言
07  #define IRQ1_INT_HANDLER    // IRQ1 割込みを使用することを宣言
```

プログラムのはじめに、独自のハンドラを定義することを予め宣言する必要があります。具体的には、interrupt.h をインクルードする前に記述してください。
「IRQ0_INT_HANDLER」などのハンドラ名は interrupt.h 内で定義されています。
ハンドラ名は、表 24-1 の関数名に _HANDLER を足して「関数名 _HANDLER」になります。

これで、定義した関数

```
void IRQ0_INT(void) {  }
```

が使えるようになりましたが、この関数が割込みハンドラであることをコンパイラに教える必要があります。

```
56  void __attribute__((interrupt_handler))
57  IRQ0_INT(void)
58  {

62  }
```

割込みハンドラには「__attribute__((interrupt_handler))」を付けてプロトタイプ宣言し、C コンパイラが普通の関数と区別してコンパイルを行うようにします。
なお、割込みハンドラの記述方法は、C コンパイラによって異なります。

```
74  asm("andc.b #0x7f,ccr");    // CPU の割込み許可
```

割込みを発生させるには割込みコントローラの割込み許可フラグを立てるほかに、CPU の割込み許可が必要になります。CPU の割込み許可は、CPU 内部にある CCR というレジスタのビット操作で可能になります。しかし、C 言語では CCR に直接アクセスできません。そこで、アセンブリ言語のコマンドを使ってアクセスしています。

「CPU の割込み禁止」は以下のように記述します。

```
    asm("orc.b #0x80,ccr");    // CPU の割込み禁止
```

ベクタテーブル と 割込みベクタ

ハンドラも関数の一種ですから、他の関数と同様にメモリ上に一連の処理手順が格納されています。この関数の入り口に相当するアドレス、すなわちハンドラへのポインタは割り込みベクタと呼ばれます。割り込みベクタを要因ごとに順番に並べたものがベクタテーブルです。メモリ上にはベクタテーブルを格納するための特別な領域（リンカスクリプト内で定義されているセクション「.vectors」の領域）があります。「リセット」や「割り込み」のような例外が発生すると CPU はベクタテーブルを参照し、当該ベクタのアドレスへ分岐します。より詳しく知りたい場合は、interrupt.h を開いて内容を覗いてみましょう。VectorTable[ ] と定義されているのが、ベクタテーブルの実体です。