テーマ：割込みハンドラ

```
//user.c

/*****************************/
/*** reset process         ***/
/*****************************/
void uinit(void)
{    /* initialize */
     clock_switch();
     led_init();
     bz_init();
     int0_init();
     trb_init();
     asm("FSET I");
}
/*****************************/
/***  tasks                ***/
/*****************************/
void tsk01(void) {
     while(1){
          wai_flg(1,0x0001);
          led_set(0xff);
          tslp_tsk(500);
          led_set(0x00);
     }
}

void tsk02(void) {
     while(1) {
          wai_flg(1,0x0002);
          short_bz();
     }
}

void tsk03(void) {
     while(1) {
          slp_tsk();
     }
}
```

INT0の初期化

```
//int0.c

/*
 * int0 initialize
 */
void int0_init(void)
{
          int0en  = 1;
          int0pl  = 0;
          int0f0  = 1;
          int0f1  = 1;
          int0ic  = 0x11;
}

void intProc2(B mode)
{
     /* interrupt process */
          // イベントフラグセット
          iset_flg(1,0x0001);
          // OSに制御を移す
          disp(mode);
}
```

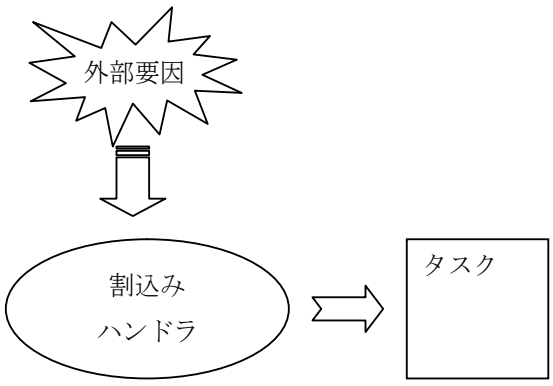
ポイント１　リアルタイム OS でも割込みが使える。リアルタイム OS では割込処理のプログラムを割込みハンドラという。

ポイント２　slp_tsk()や wai_flg()など、OS を呼び出す機能のことをサービスコールという。今まで学習してきた OS 関連のコマンドは全てサービスコールである。

ポイント３　割込みハンドラで実行できるサービスコールには先頭に i がついている。

iwup_tsk(2)・・タスク２を起床する
isus_tsk(2)・・タスク２を suspended 状態にする
iset_flg(1,0x01)・・イベントフラグをセットする

ポイント４　割込みハンドラはどのタスクよりも優先度が高く、最優先で実行される。



次のプログラムを、作成しなさい。

【実習１】外部割り込みが発生したら短くブザーを鳴らす。

【実習２】外部割り込みが発生したら短くブザーを鳴らし、sw が押されたら全ての LED を 0.5 秒間点灯する。ただし、ブザー、sw、LED は別タスクにすること。