```
/*****************************************************************************/
/*                    キー入力チャタリングキャンセル・サンプル                 */
/*****************************************************************************/

#include <machine.h>
#include <stdlib.h>
#include "typedefine.h"
#include "iodefine.h"

/* ************************************************************************** */
typedef union {
    BYTE    ALL;
    struct {
        BYTE B3L:1;                 /* B7:スイッチ４長押し */
        BYTE B2L:1;                 /* B6:スイッチ３長押し */
        BYTE B5:1;                  /* B5:スイッチ６ */
        BYTE B4:1;                  /* B4:スイッチ５ */
        BYTE B3:1;                  /* B3:スイッチ４ */
        BYTE B2:1;                  /* B2:スイッチ３ */
        BYTE B1:1;                  /* B1:スイッチ２ */
        BYTE B0:1;                  /* B0:スイッチ１ */
    } BIT;
} KEYDAT_T;
#define KEY_SW3L    (0x80)          /* B7:スイッチ４長押し */
#define KEY_SW2L    (0x40)          /* B6:スイッチ３長押し */
#define KEY_SW5     (0x20)          /* B5:スイッチ６ */
#define KEY_SW4     (0x10)          /* B4:スイッチ５ */
#define KEY_SW3     (0x08)          /* B3:スイッチ４（長押しできる） */
#define KEY_SW2     (0x04)          /* B2:スイッチ３（長押しできる） */
#define KEY_SW1     (0x02)          /* B1:スイッチ２（オートリピートする） */
#define KEY_SW0     (0x01)          /* B0:スイッチ１（オートリピートする） */
/*****************************************************************************/

#define VALIDBITS (6)                   /* キー入力に使っているビット数（b0から） */
#define CHAT_TIME (5)                   /*   50mS（10mS単位）  */
#define RPT_STTIM (CHAT_TIME+95)        /* 1000mS（  〃    ）  */
#define RPT_INTTM (10)                  /*  100mS（  〃    ）  */

#pragma abs8(KeyState,KeyPush,KeyInpBak)
static KEYDAT_T KeyState;               /* 現在のキーON/OFF状態 */
static KEYDAT_T KeyPush;                /* 現在のキーON状態 */
static BYTE KeyInpBak;                  /* キーON/OFF状態検知用メモ */

#pragma section
static WORD awKeyChat[VALIDBITS];       /* チャタリングキャンセルタイマ */

/* キー入力チャタリングキャンセル ********************************************/
/* ※10mS周期割り込みで呼ぶ。                                                 */
/* ※キーを押すと、IO.PDRB.BIT.B5～B0の対応ビッが“L”になるものとする。       */

void KeyChatCancel(void)
{
    int     nChatCnt,nLpcnt;
    BYTE    byKeyDat,byKeyChg,byBitPtn;

    /* キー入力取り込み */
    byKeyDat  = (BYTE)(~IO.PDRB.BYTE & 0x3F);
    /* 前回との変化ビット取り出し */
    byKeyChg  = (BYTE)(byKeyDat ^ KeyInpBak);
    /* 今回の状態を保存 */
    KeyInpBak = byKeyDat;

    /* キー入力チャタリングキャンセル */
    byBitPtn = 0x01;
    nLpcnt = 0;
    do{
        if((byKeyChg & byBitPtn) != 0){
            awKeyChat[nLpcnt] = 0;
        }
        else{
            nChatCnt = awKeyChat[nLpcnt];
            nChatCnt++;
            if(nChatCnt <= RPT_STTIM){
                /* チャタリングキャンセル処理 */
                if(nChatCnt == CHAT_TIME){
                    if((byKeyDat & byBitPtn) != 0){
                        KeyState.ALL |= byBitPtn;
                        KeyPush.ALL  |= byBitPtn;
                    }
                    else{
                        KeyState.ALL &= (BYTE)~byBitPtn;
                    }
                }
                /* 長押し＆オートリピート処理 */
                else if(nChatCnt == RPT_STTIM){
                    if((byKeyDat & byBitPtn) != 0){
                        switch(nLpcnt){
```

```
                        case 0:         /* B0はオートリピートするキー */
                            KeyPush.BIT.B0 = 1;
                            nChatCnt = RPT_STTIM - RPT_INTTM;
                            break;
                        case 1:         /* B1はオートリピートするキー */
                            KeyPush.BIT.B1 = 1;
                            nChatCnt = RPT_STTIM - RPT_INTTM;
                            break;
                        case 2:         /* B2は長押しできるキー */
                            KeyPush.BIT.B2L = 1;
//                          nChatCnt = RPT_STTIM;
                            break;
                        case 3:         /* B3は長押しできるキー */
                            KeyPush.BIT.B3L = 1;
//                          nChatCnt = RPT_STTIM;
                            break;
                        default:
                            break;
                        }
                    }
                }
                awKeyChat[nLpcnt] = nChatCnt;
            }
        }
        byBitPtn <<= 1;
    }while(++nLpcnt < VALIDBITS);

}

/*****************************************************************************/
/*                              キー入力関数                                 */
/*****************************************************************************/

/* 現在のキースイッチの状態を読み込む ****************************************/

KEYDAT_T GetKeyState(void)
{
    return KeyState;
}

/* キースイッチが押されるのを待つ ********************************************/

KEYDAT_T WaitKey(void)
{
    KeyPush.ALL = 0;
    while(KeyPush.ALL == 0){
        nop();
    }

    return KeyPush;
}

/* 押されたキースイッチを読み込む ********************************************/

KEYDAT_T GetKey(void)
{
    KEYDAT_T KeyNow;
    BYTE    byCCR;

    /* 割り込みマスク */
    byCCR = get_ccr();
    set_imask_ccr(1);
    /* キー入力データ取得 */
    KeyNow.ALL  = KeyPush.ALL;
    KeyPush.ALL = 0;
    /* 割り込みマスク復帰 */
    set_ccr(byCCR);

    return KeyNow;
}
```