```
/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/***                              サウンド出力                             ***/
/***                                Sound                                  ***/
/***                                                    by:  S.Suzuki      ***/
/*****************************************************************************/
/*****************************************************************************/
#include    <machine.h>
#include    "typedefine.h"
#include    "iodefine.h"

#include    "InitSystemClock.h"
#include    "Sound.h"

/*****************************************************************************/
/* 注意 : このプログラムはPCLK=25MHz専用です。                               */
#if (PCLK_HZ != 25000000L)
    #error "It is exclusive for PCLK = 25 MHz"
#endif

static int  st_nSoundRPos;
static SOUND_T* st_pSound;
static SOUNDSTAT_T SoundStatus;

/*****************************************************************************/

/* 時報音 */
SOUND_T Chime_1[]={
    {RA3_, T16_},       /*  ラ (A7)  */
    {STP_, T00_},
};

/* ビープ音 */
SOUND_T Melody_1[]={
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},
    {DO3_, T16_},
    {RST_, T32_},

    {RPT_, T00_},
};

/* ロンドン橋 */
SOUND_T Melody_2[]={
    {SO1_, T08_ + (T08_/2)},
    {RA1_, T16_},
    {SO1_, T08_},
    {FA1_, T08_},
    {MI1_, T08_},
    {FA1_, T08_},
    {SO1_, T04_},
    {RE1_, T08_},
    {MI1_, T08_},
    {FA1_, T04_},
    {MI1_, T08_},
    {FA1_, T08_},
    {SO1_, T04_},
    {SO1_, T08_ + (T08_/2)},
    {RA1_, T16_},
    {SO1_, T08_},
    {FA1_, T08_},
```

```
    {MI1_, TO8_},
    {FA1_, TO8_},
    {SO1_, TO4_},
    {RE1_, TO4_},
    {SO1_, TO4_},
    {MI1_, TO8_},
    {DO1_, TO4_ + (TO4_/2)},

    {SO1_, TO8_ + (TO8_/2)},
    {RA1_, T16_},
    {SO1_, TO8_},
    {FA1_, TO8_},
    {MI1_, TO8_},
    {FA1_, TO8_},
    {SO1_, TO4_},
    {RE1_, TO8_},
    {MI1_, TO8_},
    {FA1_, TO4_},
    {MI1_, TO8_},
    {FA1_, TO8_},
    {SO1_, TO4_},
    {SO1_, TO8_ + (TO8_/2)},
    {RA1_, T16_},
    {SO1_, TO8_},
    {FA1_, TO8_},
    {MI1_, TO8_},
    {FA1_, TO8_},
    {SO1_, TO4_},
    {RE1_, TO4_},
    {SO1_, TO4_},
    {MI1_, TO8_},
    {DO1_, TO4_ + (TO4_/2)},

    {RPT_, TOO_},
};

/*****************************************************************************/
/*                    音階および長さを生成するMTUを初期化                    */
/*****************************************************************************/
/* 注意 : PCLK=25MHzであること。                                             */
/* 機能 : MTU0で音階を作成し、MTIOC0A(PB3)端子から出力する。                 */
/*      : MTU3で音の長さを作成する。                                         */

void Init_Sound(void)
{
    /*** MTIOC0A(PB3)端子の初期化 ***/

    /* MTIOC0A(PB3)端子（ポートＢ）のモード設定 */
//  PORTB.PDR.BIT.B3 = 1;
    PORTB.PMR.BIT.B3 = 0;               /* PB3をI/Oポートとして使う設定 */
    /* マルチファンクションピンコントローラ（MPC） */
    MPC.PWPR.BYTE = 0x00;               /* 書き込みプロテクトレジスタ（BOWI=0にする）       */
    MPC.PWPR.BYTE = 0x40;               /* 書き込みプロテクトレジスタ（PFSWE=1にして解除） */
    /* PB3端子機能制御レジスタ */
    MPC.PB3PFS.BYTE = 0x01;             /* PB3端子をMTIOC0A出力として使用する               */
    /* マルチファンクションピンコントローラ（MPC） */
    MPC.PWPR.BYTE = 0x80;               /* 書き込みプロテクトレジスタ（POWI=1にして保護）   */
    /* MTIOC0A(PB3)端子（ポートＢ）のモード設定 */
    PORTB.PMR.BIT.B3 = 1;               /* PB3を周辺機能として使う設定 */


    /*** マルチファンクションタイマパルスユニット(MTU)初期化 ***/

    /* 消費電力低減機能関連レジスタへの書き込み許可 */
    SYSTEM.PRCR.WORD = 0xA502;
    /* MTU0～MTU5のモジュールストップ解除 */
    MSTP(MTU) = 0;                      /* SYSTEM.MSTPCRA.BIT.MSTPA9 = 0;（MTU0～MTU5共通） */
    /* 消費電力低減機能関連レジスタへの書き込みを禁止する */
    SYSTEM.PRCR.WORD = 0xA500;

    /* マルチファンクションタイマパルスユニット(MTU0)初期化 */
    /* MTU0カウント停止 */
    MTU.TSTR.BIT.CST0 = 0;
    MTU0.TCR.BYTE = 0x21;               /* TGRAのコンペアマッチでTCNTクリア、立ち上がりエッジ、
```

```
148 PCLK/4 (6.25MHz, 160nS) でカウント */
149     MTU0.TMDR.BYTE = 0x02;                  /* 動作モード=PWMモード1 */
150     MTU0.TIORH.BYTE = 0x00;                 /* MTU0.TGRAをアウトプットコンペアレジスタとしてMTIOC0A
    端子=出力禁止 */
151                                             /* MTU0.TGRBをアウトプットコンペアレジスタとしてMTIOC0B
    端子=出力禁止 */
152     MTU0.TIORL.BYTE = 0x00;                 /* MTU0.TGRCをアウトプットコンペアレジスタとしてMTIOC0C
    端子=出力禁止 */
153                                             /* MTU0.TGRDをアウトプットコンペアレジスタとしてMTIOC0D
    端子=出力禁止 */
154     MTU0.TIER.BYTE = 0x00;                  /* 割り込み要求 (TGIA,TGIB,TGIC,TGID,TCIV,TCIU) を禁止
                */
155     MTU0.TIER2.BYTE = 0x00;                 /* 割り込み要求 (TGIE,TGIF) を禁止
                */
156     MTU.TSYR.BIT.SYNC0 = 0;                 /* MTU0.TCNTは独立して動作
                */
157
158     /* マルチファンクションタイマパルスユニット(MTU3)初期化 */
159     MTU.TRWER.BYTE = 0x01;                  /* レジスタ (MTU3,MTU4用) の読み出し/書き込みを許可  */

160     /* MTU3カウント停止 */
161     MTU.TSTR.BIT.CST3 = 0;
162     MTU3.TCR.BYTE = 0x25;                   /* TGRAのコンペアマッチでTCNTクリア、立ち上がりエッジ、
    PCLK/1024(24.4140KHz,40.96uS) でカウント */
163     MTU3.TMDR.BYTE = 0x00;                  /* 動作モード=通常動作  */
164     MTU3.TIORH.BYTE = 0x00;                 /* MTU3.TGRAをアウトプットコンペアレジスタとしてMTIOC3A
    端子=出力禁止 */
165                                             /* MTU3.TGRBをアウトプットコンペアレジスタとしてMTIOC3B
    端子=出力禁止 */
166     MTU3.TIORL.BYTE = 0x00;                 /* MTU3.TGRCをアウトプットコンペアレジスタとしてMTIOC3C
    端子=出力禁止 */
167                                             /* MTU3.TGRDをアウトプットコンペアレジスタとしてMTIOC3D
    端子=出力禁止 */
168     MTU3.TIER.BYTE = 0x01;                  /* 割り込み要求 (TGIA) 許可、その他 (TGIB,TGIC,TGID,TCI
    V,TCIU) 禁止  */
169     MTU.TSYR.BIT.SYNC3 = 0;                 /* MTU3.TCNTは独立して動作
                */
170
171     /* MTU0タイマジェネラルレジスタをセット */
172     MTU0.TCNT = 0x0000;                     /* タイマカウンタ */
173     MTU0.TGRA = 0xFFFF;                     /* タイマジェネラルレジスタA */
174     /* MTU3タイマジェネラルレジスタをセット */
175     MTU3.TCNT = 0x0000;                     /* タイマカウンタ */
176     MTU3.TGRA = 0xFFFF;                     /* タイマジェネラルレジスタA */
177
178     /* MTU3の割り込み優先順位セット */
179     IPR(MTU3, TGI) = 5;                     /* TGIA3,TGIB3,TGIC3,TGID3は共通 */
180     /* MTU0,MTU3コンペアマッチ割り込み禁止 */
181     IEN(MTU0, TGIA0) = 0;
182     IEN(MTU0, TGIB0) = 0;
183     IEN(MTU0, TGIC0) = 0;
184     IEN(MTU0, TGID0) = 0;
185     IEN(MTU0, TGIE0) = 0;
186     IEN(MTU0, TGIF0) = 0;
187     IEN(MTU3, TGIA3) = 0;
188     IEN(MTU3, TGIB3) = 0;
189     IEN(MTU3, TGIC3) = 0;
190     IEN(MTU3, TGID3) = 0;
191     /* MTU0,MTU3オーバフロー／アンダーフロー割り込み禁止 */
192     IEN(MTU0, TCIV0) = 0;
193     IEN(MTU3, TCIV3) = 0;
194 }
195
196 /*****************************************************************************/
197 /*                          MTU3のコンペアマッチ割り込み処理                 */
198 /*****************************************************************************/
199
200 /* コンペアマッチＡ割り込み */
201
202 #pragma interrupt (Int_MTU3_TGIA3)
203 void Int_MTU3_TGIA3(void)
204 {
205     if(SoundStatus.BIT.SOUND_SP == 1){
206         SoundStatus.BIT.SOUND_SP = 0;
```

```
207         SoundStatus.BIT.SOUND_IT = 0;
208         st_nSoundRPos = 0;
209     }
210     /* 終端マークだったら終了 */
211     if(st_pSound[st_nSoundRPos].NOTES == STP_){
212         Stop_Sound();
213     }
214     /* 繰り返しマークだったらSPACEフラグセット */
215     else if(st_pSound[st_nSoundRPos].NOTES == RPT_){
216         SoundStatus.BIT.SOUND_SP = 1;
217         /* MTIOCOA端子を出力禁止にする（Hi-Zになる） */
218         MTU0.TIORH.BIT.IOA = 0;
219         /* １秒の無音時間を設ける */
220         MTU3.TGRA = TO2_;
221     }
222     /* 発音中の場合 */
223     else{
224         /* 発音 */
225         if(SoundStatus.BIT.SOUND_IT == 0){
226             SoundStatus.BIT.SOUND_IT = 1;
227             /* 休符ならMTIOCOA端子の出力禁止 */
228             if(st_pSound[st_nSoundRPos].NOTES == RST_){
229                 /* MTIOCOA端子を出力禁止にする（Hi-Zになる） */
230                 MTU0.TIORH.BIT.IOA = 0;
231             }
232             /* 音符なら指定の音階を出力 */
233             else{
234                 /* 音階を出力 */
235                 MTU0.TGRA = st_pSound[st_nSoundRPos].NOTES;
236                 /* MTIOCOA端子をトグル出力にする */
237                 MTU0.TIORH.BIT.IOA = 3;
238             }
239             /* 音の時間（長さ）をセット */
240             MTU3.TGRA = (WORD)(st_pSound[st_nSoundRPos].TIME - TIT_);
241             st_nSoundRPos++;
242         }
243         /* 区切り */
244         else{
245             SoundStatus.BIT.SOUND_IT = 0;
246             /* MTIOCOA端子を出力禁止にする（Hi-Zになる） */
247             MTU0.TIORH.BIT.IOA = 0;
248             /* 音の区切り時間を設ける */
249             MTU3.TGRA = TIT_;
250         }
251     }
252 }
253
254 /*****************************************************************************/
255 /*                              サウンド発音開始                             */
256 /*****************************************************************************/
257 /* 引数 : nMelodyNum = サウンド・メロディ番号（1,2,99）                       */
258 /* 戻値 : Start_Sound() ==  TRUE : サウンド発生開始                           */
259 /*      :               == FALSE : サウンド発生なし                           */
260
261 BOOL Start_Sound(int nMelodyNum)
262 {
263     /* サウンド出力中なら停止する */
264     if(SoundStatus.BIT.SOUND_ON == 1){
265         Stop_Sound();
266     }
267     if(nMelodyNum == 1){
268         st_pSound = &Melody_1[0];   /* ビープ音 */
269     }
270     else if(nMelodyNum == 2){
271         st_pSound = &Melody_2[0];   /* ロンドン橋 */
272     }
273     else if(nMelodyNum == 99){
274         st_pSound = &Chime_1[0];    /* 時報音 */
275     }
276     else{
277         return FALSE;
278     }
279     st_nSoundRPos = 0;
280     /* 最初から終端マークまたは繰り返しマークだったら何もしない */
```

```
281     if((st_pSound[st_nSoundRPos].NOTES == RPT_) || (st_pSound[st_nSoundRPos].NOTES == STP_)){
282         return FALSE;
283     }
284     /* サウンド出力中を示す */
285     SoundStatus.BIT.SOUND_ON = 1;
286     SoundStatus.BIT.SOUND_SP = 0;
287     SoundStatus.BIT.SOUND_IT = 0;
288     /* タイマリードライト許可レジスタ（MTU3,MTU4に必要） */
289 //  MTU.TRWER.BYTE = 0x01;
290     /* 音の時間（長さ）をセット */
291     MTU3.TCNT = 0x0000;
292     MTU3.TGRA = T16_;           /* 0.125秒後に発音開始 */
293     /* MTU0,MTU3カウントスタート */
294     MTU.TSTR.BIT.CST0 = 1;
295     MTU.TSTR.BIT.CST3 = 1;
296     /* MTU3コンペアマッチA割り込み許可 */
297     IEN(MTU3, TGIA3) = 1;
298 
299     return TRUE;
300 }
301 
302 /*****************************************************************************/
303 /*                                サウンド停止                               */
304 /*****************************************************************************/
305 
306 void Stop_Sound(void)
307 {
308     /* MTU3コンペアマッチA割り込み禁止 */
309     IEN(MTU3, TGIA3) = 0;
310     /* MTIOC0A端子を出力禁止にする（Hi-Zになる） */
311     MTU0.TIORH.BIT.IOA = 0;
312     /* MTU0,MTU3カウント停止 */
313     MTU.TSTR.BIT.CST0 = 0;
314     MTU.TSTR.BIT.CST3 = 0;
315     /* 発音停止中を示す */
316     SoundStatus.BIT.SOUND_ON = 0;
317     SoundStatus.BIT.SOUND_SP = 0;
318 }
319 
320 
321 /* End of File */
```