```c
/*******************************************************************************
 *
 *
 *
 *******************************************************************************
 *
 * File         : buzzer.c
 * Version      : 1.0
 *
 *******************************************************************************
 *
 * Description  : Managing buzzer state machine and sequences
 *
 *                Thanks to robsoncouto on GitHub for musics !
 *                https://github.com/robsoncouto/arduino-songs.git
 *
 *******************************************************************************
 *
 * Author       : Miguel Santos
 * Date         : 25.09.2023
 *
 *******************************************************************************
 *
 * MPLAB X      : 5.45
 * XC32         : 2.50
 * Harmony      : 2.06
 *
 ******************************************************************************/

#include "bzr.h"
#include "peripheral/oc/plib_oc.h"

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

/* Define the timer used by the buzzer */
#define BZR_TMR_ID TMR_ID_3

/* Define the OC output used by the buzzer */
#define BZR_OC_ID OC_ID_5

/* Set the volume of the buzzer by changing duty cycle */
#define BZR_VOLUME 0.1

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

/* Define frequencies of notes in Hz */
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
```

```
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST      0

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

/* All sequences are defined here */

/* Test sequence */
```

```
int16_t BZR_SEQUENCE_TEST[] = {
    NOTE_A4,4, REST,4, NOTE_A4, 4,
};

/* Super Mario Bros theme - by Koji Kondo*/
int16_t BZR_SEQUENCE_MARIO[] = {
  NOTE_E5,  8,  NOTE_E5,8,  REST,8, NOTE_E5,8,  REST,8, NOTE_C5,8, NOTE_E5,8, //1
  NOTE_G5,4, REST,4, NOTE_G4,8, REST,4,
  NOTE_C5,-4, NOTE_G4,8, REST,4, NOTE_E4,-4, // 3
  NOTE_A4,4, NOTE_B4,4, NOTE_AS4,8, NOTE_A4,4,
  NOTE_G4,-8, NOTE_E5,-8, NOTE_G5,-8, NOTE_A5,4, NOTE_F5,8, NOTE_G5,8,
  REST,8, NOTE_E5,4,NOTE_C5,8, NOTE_D5,8, NOTE_B4,-4,
  NOTE_C5,-4, NOTE_G4,8, REST,4, NOTE_E4,-4, // repeats from 3
  NOTE_A4,4, NOTE_B4,4, NOTE_AS4,8, NOTE_A4,4,
  NOTE_G4,-8, NOTE_E5,-8, NOTE_G5,-8, NOTE_A5,4, NOTE_F5,8, NOTE_G5,8,
  REST,8, NOTE_E5,4,NOTE_C5,8, NOTE_D5,8, NOTE_B4,-4,
  REST,4, NOTE_G5,8, NOTE_FS5,8, NOTE_F5,8, NOTE_DS5,4, NOTE_E5,8,//7
  REST,8, NOTE_GS4,8, NOTE_A4,8, NOTE_C4,8, REST,8, NOTE_A4,8, NOTE_C5,8, NOTE_D5,8,
  REST,4, NOTE_DS5,4, REST,8, NOTE_D5,-4,
  NOTE_C5,2, REST,2,
  REST,4, NOTE_G5,8, NOTE_FS5,8, NOTE_F5,8, NOTE_DS5,4, NOTE_E5,8,//repeats from 7
  REST,8, NOTE_GS4,8, NOTE_A4,8, NOTE_C4,8, REST,8, NOTE_A4,8, NOTE_C5,8, NOTE_D5,8,
  REST,4, NOTE_DS5,4, REST,8, NOTE_D5,-4,
  NOTE_C5,2, REST,2,
  NOTE_C5,8, NOTE_C5,4, NOTE_C5,8, REST,8, NOTE_C5,8, NOTE_D5,4,//11
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2,
  NOTE_C5,8, NOTE_C5,4, NOTE_C5,8, REST,8, NOTE_C5,8, NOTE_D5,8, NOTE_E5,8,//13
  REST,1,
  NOTE_C5,8, NOTE_C5,4, NOTE_C5,8, REST,8, NOTE_C5,8, NOTE_D5,4,
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2,
  NOTE_E5,8, NOTE_E5,8, REST,8, NOTE_E5,8, REST,8, NOTE_C5,8, NOTE_E5,4,
  NOTE_G5,4, REST,4, NOTE_G4,4, REST,4,
  NOTE_C5,-4, NOTE_G4,8, REST,4, NOTE_E4,-4, // 19
  NOTE_A4,4, NOTE_B4,4, NOTE_AS4,8, NOTE_A4,4,
  NOTE_G4,-8, NOTE_E5,-8, NOTE_G5,-8, NOTE_A5,4, NOTE_F5,8, NOTE_G5,8,
  REST,8, NOTE_E5,4, NOTE_C5,8, NOTE_D5,8, NOTE_B4,-4,
  NOTE_C5,-4, NOTE_G4,8, REST,4, NOTE_E4,-4, // repeats from 19
  NOTE_A4,4, NOTE_B4,4, NOTE_AS4,8, NOTE_A4,4,
  NOTE_G4,-8, NOTE_E5,-8, NOTE_G5,-8, NOTE_A5,4, NOTE_F5,8, NOTE_G5,8,
  REST,8, NOTE_E5,4, NOTE_C5,8, NOTE_D5,8, NOTE_B4,-4,
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,//23
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_D5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_G5,-8, NOTE_F5,-8,
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2, //26
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_B4,8, NOTE_F5,4, NOTE_F5,8, NOTE_F5,-8, NOTE_E5,-8, NOTE_D5,-8,
  NOTE_C5,8, NOTE_E4,4, NOTE_E4,8, NOTE_C4,2,
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,//repeats from 23
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_D5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_G5,-8, NOTE_F5,-8,
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2, //26
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_B4,8, NOTE_F5,4, NOTE_F5,8, NOTE_F5,-8, NOTE_E5,-8, NOTE_D5,-8,
  NOTE_C5,8, NOTE_E4,4, NOTE_E4,8, NOTE_C4,2,
  NOTE_C5,8, NOTE_C5,4, NOTE_C5,8, REST,8, NOTE_C5,8, NOTE_D5,8, NOTE_E5,8,
  REST,1,
  NOTE_C5,8, NOTE_C5,4, NOTE_C5,8, REST,8, NOTE_C5,8, NOTE_D5,4, //33
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2,
  NOTE_E5,8, NOTE_E5,8, REST,8, NOTE_E5,8, REST,8, NOTE_C5,8, NOTE_E5,4,
  NOTE_G5,4, REST,4, NOTE_G4,4, REST,4,
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_D5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_A5,-8, NOTE_G5,-8, NOTE_F5,-8,
  NOTE_E5,8, NOTE_C5,4, NOTE_A4,8, NOTE_G4,2, //40
  NOTE_E5,8, NOTE_C5,4, NOTE_G4,8, REST,4, NOTE_GS4,4,
  NOTE_A4,8, NOTE_F5,4, NOTE_F5,8, NOTE_A4,2,
  NOTE_B4,8, NOTE_F5,4, NOTE_F5,8, NOTE_F5,-8, NOTE_E5,-8, NOTE_D5,-8,
  NOTE_C5,8, NOTE_E4,4, NOTE_E4,8, NOTE_C4,2,
  //game over sound
  NOTE_C5,-4, NOTE_G4,-4, NOTE_E4,4, //45
  NOTE_A4,-8, NOTE_B4,-8, NOTE_A4,-8, NOTE_GS4,-8, NOTE_AS4,-8, NOTE_GS4,-8,
```

```
220     NOTE_G4,8, NOTE_D4,8, NOTE_E4,-2,
221
222   };
223   /* Dart Vader theme (Imperial March) - Star wars */
224   int16_t BZR_SEQUENCE_IMPERIAL[] = {
225       NOTE_A4,  -4, NOTE_A4,  -4, NOTE_A4,  16, NOTE_A4,  16,
226       NOTE_A4,  16, NOTE_A4,  16, NOTE_F4,   8, REST,      8,
227       NOTE_A4,  -4, NOTE_A4,  -4, NOTE_A4,  16, NOTE_A4,  16,
228       NOTE_A4,  16, NOTE_A4,  16, NOTE_F4,   8, REST,      8,
229       NOTE_A4,   4, NOTE_A4,   4, NOTE_A4,   4,
230       NOTE_F4,  -8, NOTE_C5,  16, NOTE_A4,   4,
231       NOTE_F4,  -8, NOTE_C5,  16, NOTE_A4,   2,
232       NOTE_E5,   4, NOTE_E5,   4, NOTE_E5,   4,
233       NOTE_F5,  -8, NOTE_C5,  16, NOTE_A4,   4,
234       NOTE_F4,  -8, NOTE_C5,  16, NOTE_A4,   2,
235       NOTE_A5,   4, NOTE_A4,  -8, NOTE_A4,  16,
236       NOTE_A5,   4, NOTE_GS5, -8, NOTE_G5,  16,
237       NOTE_DS5, 16, NOTE_D5,  16, NOTE_DS5,  8, REST,      8,
238       NOTE_A4,   8, NOTE_DS5,  4, NOTE_D5,  -8, NOTE_CS5, 16,
239       NOTE_C5,  16, NOTE_B4,  16, NOTE_C5,  16, REST,      8,
240       NOTE_F4,   8, NOTE_GS4,  4, NOTE_F4,  -8, NOTE_A4, -16,
241       NOTE_C5,   4, NOTE_A4,  -8, NOTE_C5,  16, NOTE_E5,   2,
242       NOTE_A5,   4, NOTE_A4,  -8, NOTE_A4,  16,
243       NOTE_A5,   4, NOTE_GS5, -8, NOTE_G5,  16,
244       NOTE_DS5, 16, NOTE_D5,  16, NOTE_DS5,  8, REST,      8,
245       NOTE_A4,   8, NOTE_DS5,  4, NOTE_D5,  -8, NOTE_CS5, 16,
246       NOTE_C5,  16, NOTE_B4,  16, NOTE_C5,  16, REST,      8,
247       NOTE_F4,   8, NOTE_GS4,  4, NOTE_F4,  -8, NOTE_A4, -16,
248       NOTE_A4,   4, NOTE_F4,  -8, NOTE_C5,  16,
249       NOTE_A4,   2,
250   };
251
252   /*****************************************************************************/
253
254   /* Hold informations about sequences */
255   S_BZR_SEQ BZR_SEQUENCES[] = {
256       /* TESTING */
257       {
258           .tempo = 200,
259           .size = sizeof(BZR_SEQUENCE_TEST),
260           .notes = BZR_SEQUENCE_TEST,
261       },
262       /* MARIO BROS SONG */
263       {
264           .tempo = 200,
265           .size = sizeof(BZR_SEQUENCE_MARIO),
266           .notes = BZR_SEQUENCE_MARIO,
267       },
268       /* IMPERIAL MARCH */
269       {
270           .tempo = 120,
271           .size = sizeof(BZR_SEQUENCE_IMPERIAL),
272           .notes = BZR_SEQUENCE_IMPERIAL,
273       },
274   };
275
276   /*****************************************************************************/
277
278   /* Declaration of global application data */
279   BZR_DATA bzrData;
280
281   /*****************************************************************************/
282
283   /* Static functions declaration */
284
285   static void BZR_SetFrequency(uint16_t frequency);
286   static void BZR_SetCounter(int8_t tempo);
287
288   /*****************************************************************************/
289
290   /**
291    * @brief BZR_Initialize
292    *
```

```
293     * Initialize buzzer state machine
294     *
295     * @param void
296     * @return void
297     */
298    void BZR_Initialize ( void )
299    {
300        /* Place the buzzer state machine in its initial state. */
301        bzrData.state = BZR_STATE_IDLE;
302
303        /* Flag to indicate a new sequence available */
304        bzrData.newSequence = false;
305
306        /* Calculate timer frequency only one time */
307        bzrData.tmrFrequency = (uint32_t)(SYS_CLK_FREQ /
308                                          PLIB_TMR_PrescaleGet(BZR_TMR_ID));
309
310        /* Init counter used to count time of notes */
311        CNT_Initialize(&bzrData.counterPlay, 0x00);
312    }
313
314    /*****************************************************************************/
315
316    /**
317     * @brief BZR_Tasks
318     *
319     * Execute buzzer state machine, should be called cyclically
320     *
321     * @param void
322     * @return void
323     */
324    void BZR_Tasks ( void )
325    {
326
327        /* Check the application's current state. */
328        switch ( bzrData.state )
329        {
330            /* Buzzer waiting for new sequence */
331            case BZR_STATE_IDLE:
332            {
333                if(bzrData.newSequence)
334                {
335                    bzrData.state = BZR_STATE_NOTE;
336                }
337                break;
338            }
339
340            /* Buzzer getting the note to play */
341            case BZR_STATE_NOTE:
342            {
343                BZR_SetFrequency(bzrData.currentNote[0]);
344                BZR_SetCounter(bzrData.currentNote[1]);
345                bzrData.state = BZR_STATE_PLAYING;
346                break;
347            }
348
349            /* Buzzer playing the note and waiting */
350            case BZR_STATE_PLAYING:
351            {
352                if(CNT_Check(&bzrData.counterPlay))
353                {
354                    if(bzrData.currentNote >= bzrData.lastNote)
355                    {
356                        PLIB_TMR_Stop(BZR_TMR_ID);
357                        PLIB_OC_Disable(BZR_OC_ID);
358                        bzrData.newSequence = false;
359                        bzrData.state = BZR_STATE_IDLE;
360                    }
361                    else
362                    {
363                        bzrData.currentNote += 2;
364                        bzrData.state = BZR_STATE_NOTE;
365                    }
```

```
366             }
367             break;
368         }
369 
370         /* The default state should never be executed. */
371         default:
372         {
373             /* TODO: Handle error in application's state machine. */
374             break;
375         }
376     }
377 }
378 
379 /*******************************************************************************/
380 
381 /**
382  * @brief BZR_PlaySequence
383  *
384  * Play a music sequence using the state machine
385  *
386  * @param   E_BZR_SEQ song  Call the music you wann play !
387  * @return  void
388  */
389 void BZR_PlaySequence(E_BZR_SEQ song)
390 {
391     bzrData.sequence = BZR_SEQUENCES[song].notes;
392 
393     bzrData.tempo = BZR_SEQUENCES[song].tempo;
394 
395     bzrData.currentNote = bzrData.sequence;
396 
397     bzrData.lastNote = bzrData.sequence + BZR_SEQUENCES[song].size / 2 - 2;
398 
399     bzrData.newSequence = true;
400 }
401 
402 /*******************************************************************************/
403 
404 /**
405  * @brief BZR_SetFrequency
406  *
407  * Set the frequency of the timer to play a note
408  * Start the timer and OC except if frequency is 0
409  *
410  * @param   uint16_t frequency  Frequency to play
411  * @return  void
412  */
413 static void BZR_SetFrequency(uint16_t frequency)
414 {
415     uint16_t period_tmr;
416 
417     PLIB_TMR_Stop(BZR_TMR_ID);
418     PLIB_OC_Disable(BZR_OC_ID);
419 
420     if(frequency != 0)
421     {
422         period_tmr = (uint16_t)( bzrData.tmrFrequency / frequency);
423 
424         PLIB_TMR_Period16BitSet(BZR_TMR_ID, period_tmr);
425         PLIB_OC_PulseWidth16BitSet(BZR_OC_ID, period_tmr * BZR_VOLUME);
426 
427         PLIB_TMR_Start(BZR_TMR_ID);
428         PLIB_OC_Enable(BZR_OC_ID);
429     }
430 }
431 
432 /*******************************************************************************/
433 
434 /**
435  * @brief BZR_SetCounter
436  *
437  * Set the duration of the note, using musical tempo
438  * (whole, half, quarter, eigth, ...)
```

```
439     *
440     * @param   int8_t tempo  note tempo
441     * @return  void
442     */
443    static void BZR_SetCounter(int8_t tempo)
444    {
445        uint32_t counter_ms;
446
447        if(tempo > 0)
448        {
449            counter_ms = (uint32_t)((60000 * 4 / bzrData.tempo) / tempo);
450        }
451        else if(tempo < 0)
452        {
453            counter_ms = (uint32_t)((60000 * 4 / bzrData.tempo)/ abs(tempo));
454            counter_ms = counter_ms * 1.5;
455        }
456        else
457        {
458            counter_ms = 0;
459        }
460
461        CNT_Set(&bzrData.counterPlay, counter_ms);
462        CNT_Reset(&bzrData.counterPlay);
463    }
464
465    /*****************************************************************************/
466
467    /* End of File ***************************************************************/
468
```