```c
/*********************************************************************************
 *
 *    ________  __________  _____     _____  ______          ________   ______
 *   |_   __  ||  _   _  ||_    \   /   _||_   _|        |_   __  |.' ____ \
 *     | |_ \_||_/ | | \_|  |    \/    |    | |   _______ | |_ \_|| (___ \_|
 *     |  _| _     | |      | |\  /| |    | |   _|_______|| _| _  _.____`.
 *    _| |__/ |   _| |_    _| |_\/_| |_  _| |__/ |       _| |__/ || \____) |
 *   |________|  |_____|  |_____||_____||________|      |________| \______.'
 *
 *********************************************************************************
 *
 * File         : buzzer.h
 * 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
 *
 ********************************************************************************/

#ifndef _BZR_H
#define _BZR_H

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

#include <stdint.h>
#include <stdbool.h>
#include "system_config.h"
#include "system_definitions.h"
#include "modules/counter.h"

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

/* Enumeration of sequences allow sequence
 * to be called by their name out of this library */
typedef enum
{
    BZR_SEQ_TEST,
    BZR_SEQ_MARIO,
    BZR_SEQ_IMPERIAL,
}E_BZR_SEQ;

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

/* Struct to hold informations about each sequence */
typedef struct
{
    /* Music tempo in BPM */
    uint16_t tempo;

    /* Size of the sequence (use sizeof) */
    uint16_t size;

    /* Point to the array holding the notes and timings */
    int16_t *notes;

}S_BZR_SEQ;

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

```

```c
/* Buzzer state machine */
typedef enum
{
    /* Buzzer waiting for new sequence */
    BZR_STATE_IDLE,

    /* Buzzer getting the note to play */
    BZR_STATE_NOTE,

    /* Buzzer playing the note and waiting */
    BZR_STATE_PLAYING,

} BZR_STATES;

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

/* Buzzer structure of global datas */
typedef struct
{
    /* The buzzer current state */
    BZR_STATES state;

    bool newSequence;

    uint32_t tmrFrequency;

    uint16_t tempo;
    int16_t *currentNote;
    int16_t *lastNote;
    int16_t *sequence;

    S_Counter counterPlay;

} BZR_DATA;

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

/**
 * @brief BZR_Initialize
 *
 * Initialize buzzer state machine
 *
 * @param void
 * @return void
 */
void BZR_Initialize ( void );

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

/**
 * @brief BZR_Tasks
 *
 * Execute buzzer state machine, should be called cyclically
 *
 * @param void
 * @return void
 */
void BZR_Tasks ( void );

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

/**
 * @brief BZR_PlaySequence
 *
 * Play a music sequence using the state machine
 *
 * @param   E_BZR_SEQ song  Call the music you wann play !
 * @return  void
 */
void BZR_PlaySequence(E_BZR_SEQ song);

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

```

```
147 #endif /* _BZR_H */
148
149 /* End of File ******************************************************************/
150
```