Print to PDF ▸

Music

Contents

4.1. Attributes

`music`

The instance of class Music is used to control the on-board buzzer to play melodies.

4.2. Classes

`class Music(pin, ticks=4, bpm=120)`

The Music class can be used to play melodies through the Pico:ed on-board buzzer.

- **pin -** buzzer pin
- **ticks -** Many ticks to form a beat. The default value is 4.
- **bpm -** The number of beats per minute. The default value is 120.

```
**set_tempo(ticks=4, bpm=120)**
```

Sets the approximate speed of play.

- **ticks -** Many ticks form a beat. The default value is 4.
- **bpm -**The number of beats per minute. The default value is 120.

```
**get_tempo()**
```

Get the current tempo as an integer tuple:（ticks，bpm）。

```
**play(music)**
```

Play the melody.

- **music -** Music DSL.

```
**play_async(music)**
```

Play melody asynchronously.

- **music -** Sheet music DSL.

```
**pitch(frequency, duration=-1)**
```

Play the pitch for the given number of milliseconds at the given integer frequency.

- **frequency -** The specified frequency.
- **duration -** The number of delay milliseconds.

```
**pitch_async(frequency, duration=-1)**
```

Play the specified number of milliseconds of pitch asynchronously at the given integer frequency.

- **frequency -** The specified frequency.
- **duration -** The number of milliseconds of delay.

```
**stop()**
```

Stop playing the music. In fact, it only works for`**play_async(music)**`。

```
**reset()**
```

Resets to default status.

4.3. Sheet Music DSL

The individual notes are as follows :

```
NOTE[octave][:duration]
```

For example, `A1:4` refers to the note “A” in octave 1 that lasts for four ticks (a tick is an arbitrary length of time defined by a tempo setting function - see below). If the note name `R` is used then it is treated as a rest (silence). Accidentals (flats and sharps) are denoted by the `b` (flat - a lower case b) and `#` (sharp - a hash symbol). For example, `Ab` is A-flat and `C#` is C-sharp. **Note that names are not case-sensitive.** The `octave` and `duration` parameters are states that carry over to subsequent notes until re-specified. The default states are `octave = 4` (containing middle C) and `duration = 4` (a crotchet, given the default tempo settings - see below). For example, if 4 ticks is a crotchet, the following list is crotchet, quaver, quaver, crotchet based arpeggio:

```
['c1:4', 'e:2', 'g', 'c2:4']
```

The opening of Beethoven’s 5th Symphony would be encoded thus:

```
['r4:2', 'g', 'g', 'g', 'eb:8', 'r:2', 'f', 'f', 'f', 'd:8']
```

The definition and scope of an octave conforms to the table listed on this page about scientific pitch notation. For example, middle “C” is `'c4'` and concert “A” (440) is `'a4'`. Octaves start on the note “C”. The library has quite a lot of built-in melodies. Here’s a complete list:

- DADADADUM
- ENTERTAINER
- PRELUDE
- ODE
- NYAN
- RINGTONE
- FUNK
- BLUES
- BIRTHDAY
- WEDDING
- FUNERAL
- PUNCHLINE

- PYTHON
- BADDY
- CHASE
- BA_DING
- WAWAWAWAA
- JUMP_UP
- JUMP_DOWN
- POWER_UP
- POWER_DOWN

4.4. Example

1. Play built-in music

```
from picoed import music

music.play(music.DADADADUM)
```

1. Simultaneous play

```
from picoed import music

# play Prelude in C.
notes = [
    'c4:1', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5',
'e5', 'g4', 'c5', 'e5',
    'c4', 'd', 'a', 'd5', 'f5', 'a4', 'd5', 'f5', 'c4', 'd', 'a', 'd5', 'f5',
'a4', 'd5', 'f5',
    'b3', 'd4', 'g', 'd5', 'f5', 'g4', 'd5', 'f5', 'b3', 'd4', 'g', 'd5',
'f5', 'g4', 'd5', 'f5',
    'c4', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5', 'e5',
'g4', 'c5', 'e5',
    'c4', 'e', 'a', 'e5', 'a5', 'a4', 'e5', 'a5', 'c4', 'e', 'a', 'e5', 'a5',
'a4', 'e5', 'a5',
    'c4', 'd', 'f#', 'a', 'd5', 'f#4', 'a', 'd5', 'c4', 'd', 'f#', 'a', 'd5',
'f#4', 'a', 'd5',
    'b3', 'd4', 'g', 'd5', 'g5', 'g4', 'd5', 'g5', 'b3', 'd4', 'g', 'd5',
'g5', 'g4', 'd5', 'g5',
    'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'b3', 'c4', 'e', 'g', 'c5',
'e4', 'g', 'c5',
    'a3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'a3', 'c4', 'e', 'g', 'c5',
'e4', 'g', 'c5',
    'd3', 'a', 'd4', 'f#', 'c5', 'd4', 'f#', 'c5', 'd3', 'a', 'd4', 'f#',
'c5', 'd4', 'f#', 'c5',
    'g3', 'b', 'd4', 'g', 'b', 'd', 'g', 'b', 'g3', 'b3', 'd4', 'g', 'b', 'd',
'g', 'b'
]

music.play(notes)
```

1. Asynchronous play

```
import asyncio
from picoed import music, led

# play Prelude in C.
notes = [
    'c4:1', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5',
'e5', 'g4', 'c5', 'e5',
    'c4', 'd', 'a', 'd5', 'f5', 'a4', 'd5', 'f5', 'c4', 'd', 'a', 'd5', 'f5',
'a4', 'd5', 'f5',
    'b3', 'd4', 'g', 'd5', 'f5', 'g4', 'd5', 'f5', 'b3', 'd4', 'g', 'd5',
'f5', 'g4', 'd5', 'f5',
    'c4', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5', 'e5',
'g4', 'c5', 'e5',
    'c4', 'e', 'a', 'e5', 'a5', 'a4', 'e5', 'a5', 'c4', 'e', 'a', 'e5', 'a5',
'a4', 'e5', 'a5',
    'c4', 'd', 'f#', 'a', 'd5', 'f#4', 'a', 'd5', 'c4', 'd', 'f#', 'a', 'd5',
'f#4', 'a', 'd5',
    'b3', 'd4', 'g', 'd5', 'g5', 'g4', 'd5', 'g5', 'b3', 'd4', 'g', 'd5',
'g5', 'g4', 'd5', 'g5',
    'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'b3', 'c4', 'e', 'g', 'c5',
'e4', 'g', 'c5',
    'a3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'a3', 'c4', 'e', 'g', 'c5',
'e4', 'g', 'c5',
    'd3', 'a', 'd4', 'f#', 'c5', 'd4', 'f#', 'c5', 'd3', 'a', 'd4', 'f#',
'c5', 'd4', 'f#', 'c5',
    'g3', 'b', 'd4', 'g', 'b', 'd', 'g', 'b', 'g3', 'b3', 'd4', 'g', 'b', 'd',
'g', 'b'
]

async def blink(interval):
    while True:
        led.on()
        await asyncio.sleep(interval)
        led.off()
        await asyncio.sleep(interval)

async def main():
    player = asyncio.create_task(music.play_async(notes))
    light = asyncio.create_task(blink(0.1))
    await asyncio.gather(player)
    await asyncio.gather(light)

asyncio.run(main())
```

4.5.