デジタル演習ボードの活用（１）

テーマ

大全のボードで何か面白いものを作る

作るもの

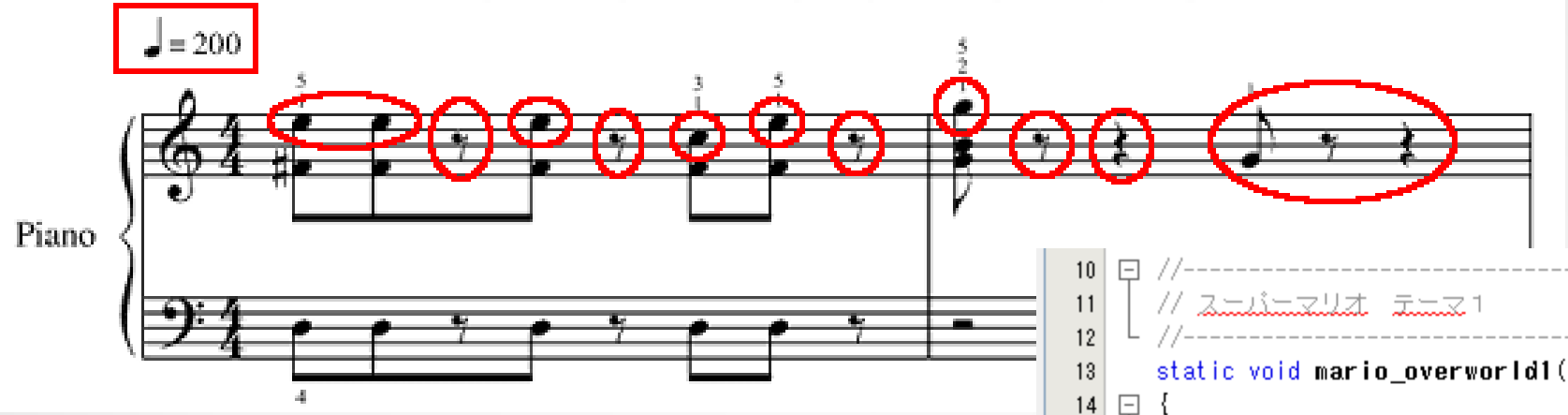
NCO による楽曲演奏

今回の特色

- MCC を使って楽に作りました
- 作成後も MCC から変更できるようにしました
- 楽譜を見ながら曲入力ができるようにしました
- 音符を１個ずつ読んで関数を呼ぶ形式にしました
- 使う関数は４つです

デジタル演習ボードの活用（２）

楽譜から音を拾う



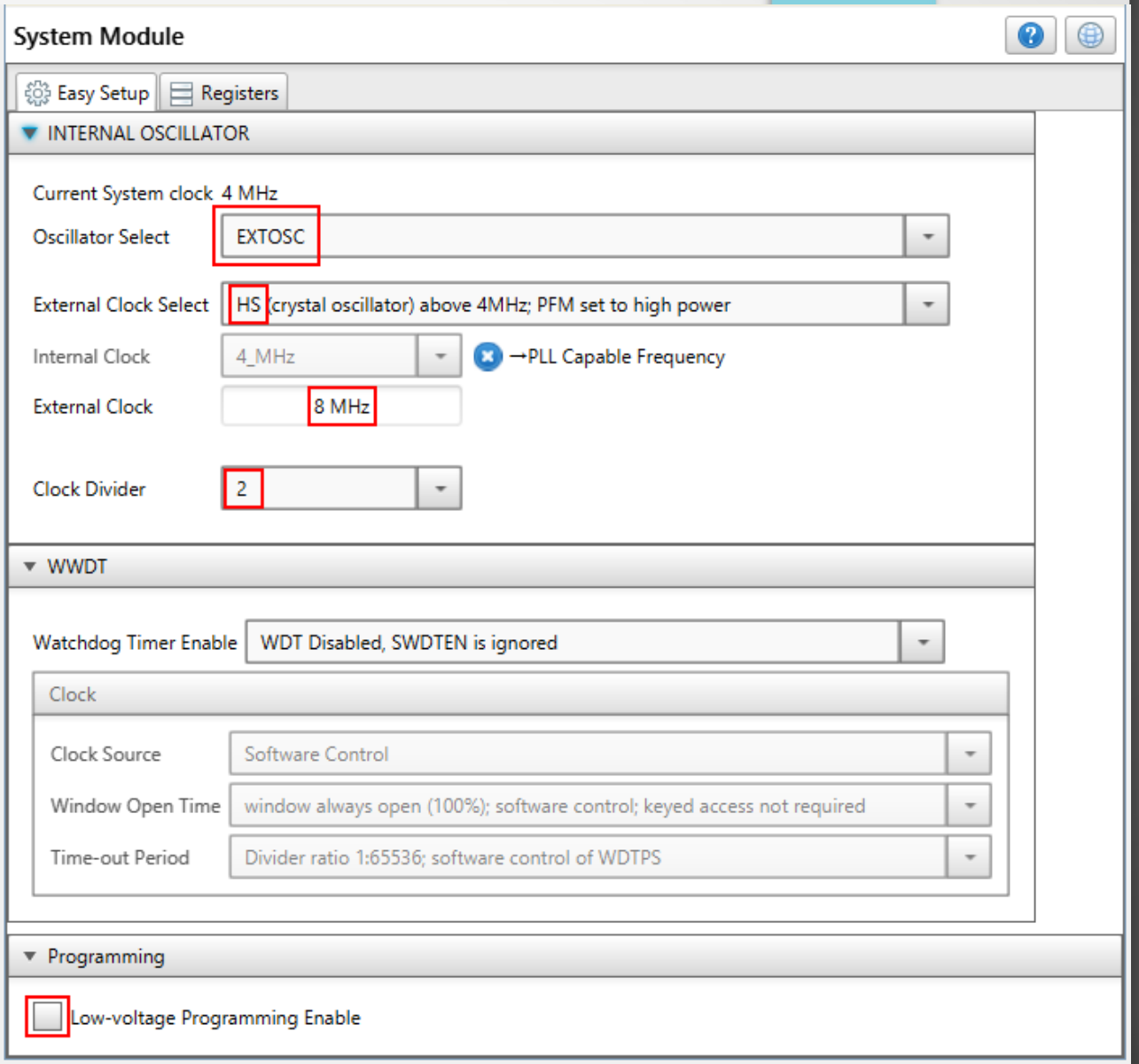

```
10  //--------------------------------------------------
11  // スーパーマリオ　テーマ１
12  //--------------------------------------------------
13  static void mario_overworld1(uint16_t tempo)
14  {
15      setTempo(tempo);              setOctave(1);
16
17      sound(p_Mi, eighth);
18      sound(p_Mi, eighth);
19      sound(rest, eighth);
20      sound(p_Mi, eighth);
21      sound(rest, eighth);
22      sound(p_Do, eighth);
23      sound(p_Mi, eighth);
24      sound(rest, eighth);
25
26      sound(p_So, eighth);
27      sound(rest, eighth);
28      sound(rest, quater);          setOctave(0);
29      sound(p_So, eighth);
30      sound(rest, eighth);
31      sound(rest, quater);
32  }
```

楽譜を読んで関数を呼ぶステップをきちんと踏んでもらうために、音符ひとつずつで sound 関数を呼ぶ仕様です。

デジタル演習ボードの活用（３）

System Module

デフォルトから赤枠部を変更します。

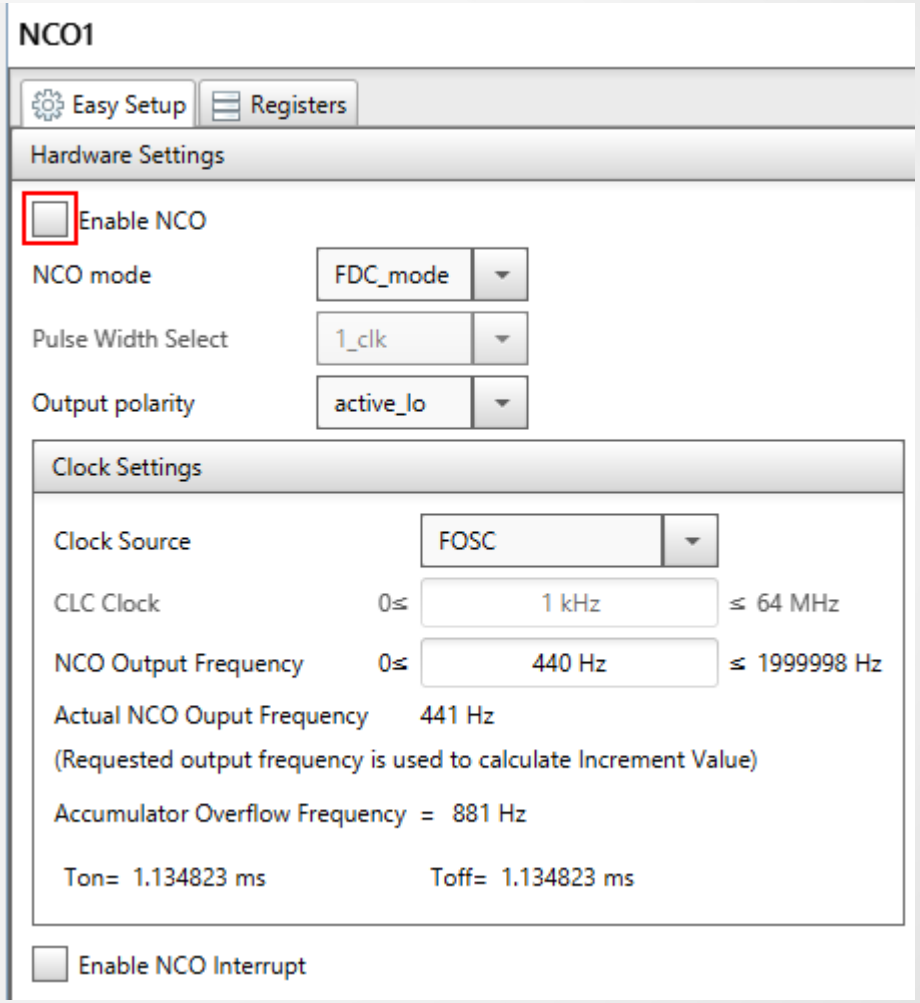


デジタル演習ボードの活用（４）

NCO1

大全と異なるのは１箇所のみです。

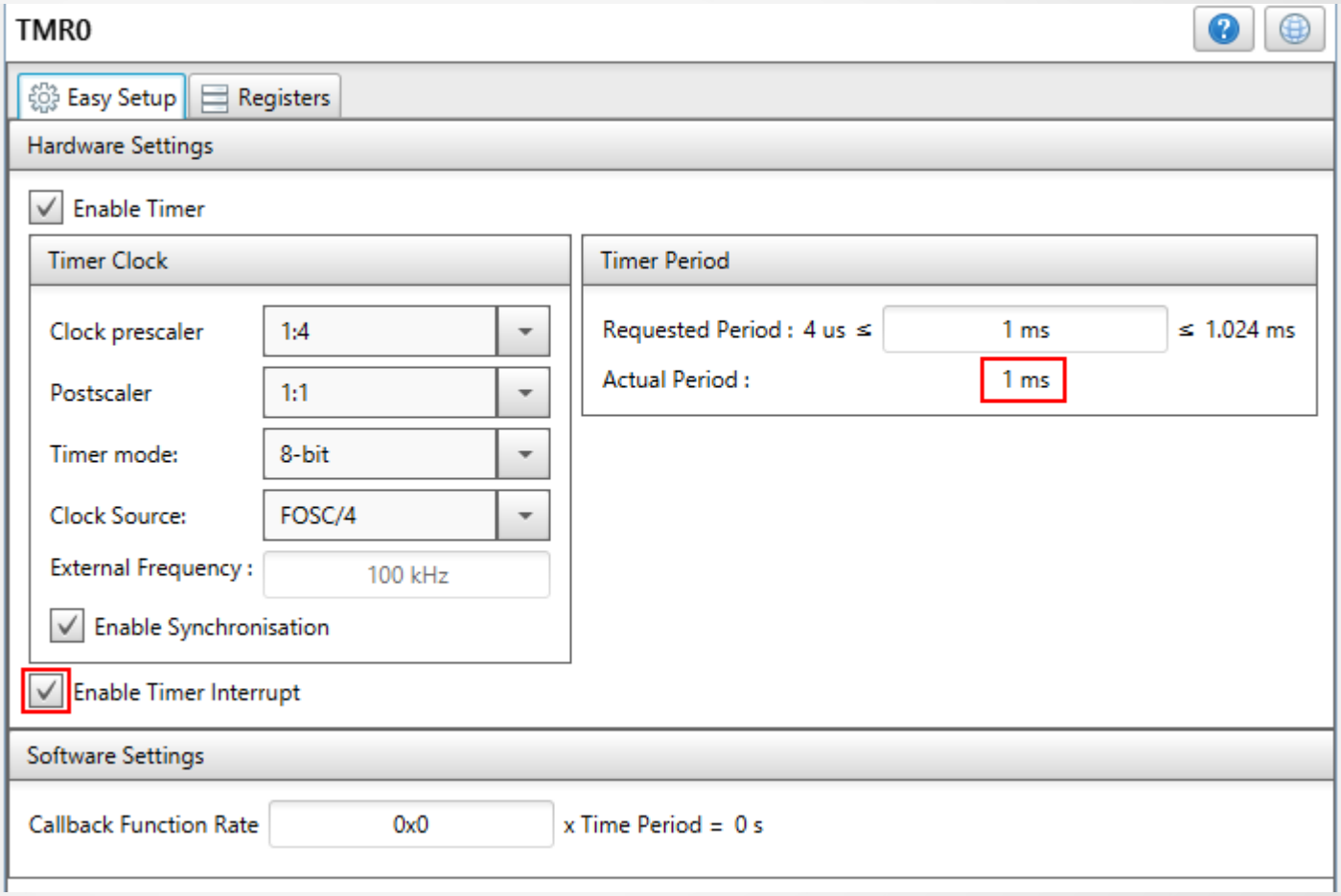
さらに、
Pin Manager: Grid View
で NCO1 に RC3 を割り当てます。



デジタル演習ボードの活用（５）

TMR0

割り込み有効に、
その他の設定は
Actual Period が
1ms になれば何でも
いいです。



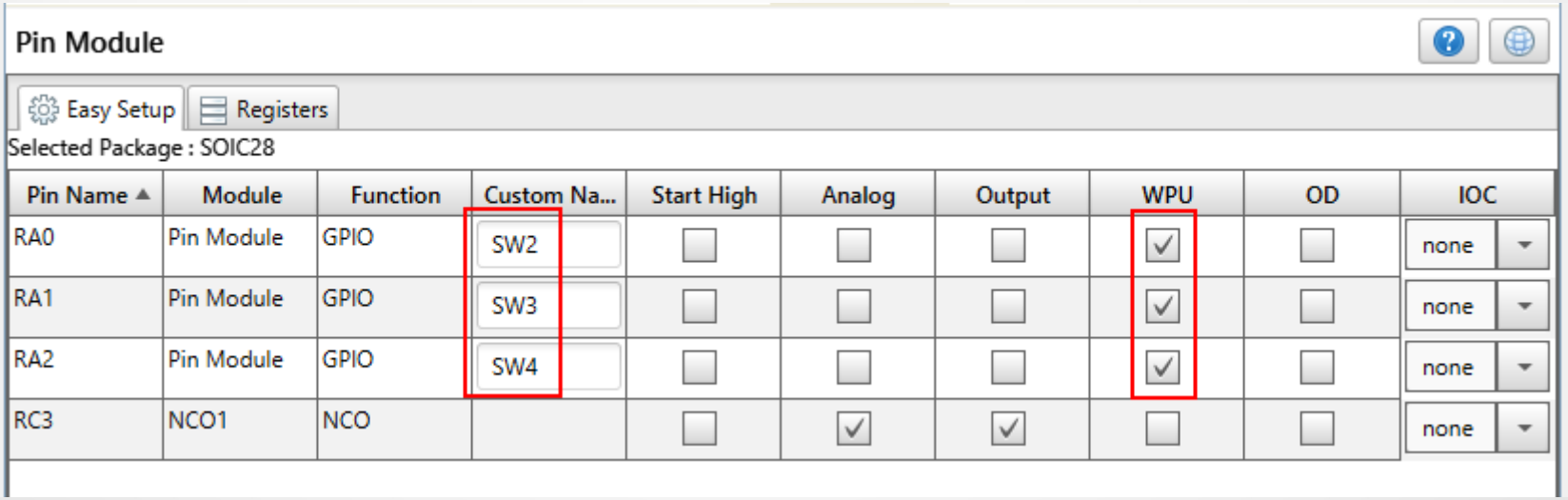
デジタル演習ボードの活用（６）

Pin Module

Pin Manager: Grid View で RA0, RA1, RA2 を GPIO input に割り当てます。

次に RA0, RA1, RA2 に下図の名前を付けます。

WPU(内蔵プルアップ) を有効にするのはデジタル演習ボードの仕様です。



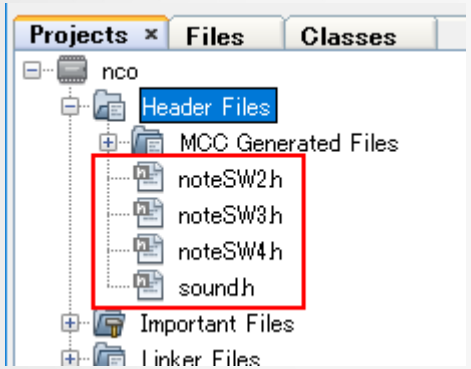
Pin Name	Module	Function	Custom Na...	Start High	Analog	Output	WPU	OD	IOC
RA0	Pin Module	GPIO	SW2	☐	☐	☐	☑	☐	none
RA1	Pin Module	GPIO	SW3	☐	☐	☐	☑	☐	none
RA2	Pin Module	GPIO	SW4	☐	☐	☐	☑	☐	none
RC3	NCO1	NCO		☐	☑	☑	☐	☐	none

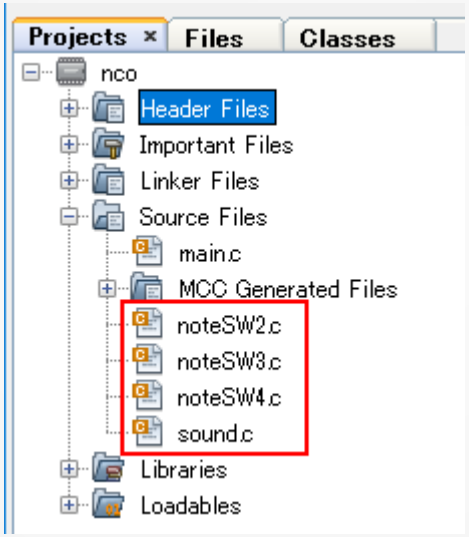
Generate して MCC を終了します。

デジタル演習ボードの活用（７）

プロジェクトに追加するファイル

```
noteSW2.h, noteSW3.h, noteSW4.h, sound.h
noteSW2.c, noteSW3.c, noteSW4.c, sound.c
```





note*.* は曲ファイル、 sound.* は音符変換ファイルです。

デジタル演習ボードの活用（８）

tmr0.c の編集

ヘッダファイルのインクルードを追加します。

```
50
#include <xc.h>
52  #include "tmr0.h"
53  #include "../sound.h"
54
```

割り込み処理の中身を記述します。

```
139 void TMR0_DefaultInterruptHandler(void){
140     // add your TMR0 interrupt custom code
141     // or set custom function using TMR0_SetInterruptHandler()
142     ctrlTime_ISR();
143 }
```

デジタル演習ボードの活用（９）

main.c の編集

ヘッダファイルのインクルードを追加します。

```
43
   #include "mcc_generated_files/mcc.h"
45 #include "noteSW3.h"
46 #include "noteSW2.h"
47 #include "noteSW4.h"
48
```

全体割り込み処理を許可します（コメントを外す）。

```
59
60     // Enable the Global Interrupts
61     INTERRUPT_GlobalInterruptEnable();
62
```

デジタル演習ボードの活用（１０）

main.c の編集

変数を用意してメインループを記述します。

```
70      //INTERRUPT_PeripheralInterruptDisable();
71
72      int8_t lastSW2 = SW2_PORT, lastSW3 = SW3_PORT, lastSW4 = SW4_PORT;
73
74      while (1)
75      {
76          // Add your application code
77          if(SW2_PORT != lastSW2) {
78              if(SW2_PORT == 0)   play_SW2();
79              else                __delay_ms(20);
80              lastSW2 = SW2_PORT;
81          }
82
83          if(SW3_PORT != lastSW3) {
84              if(SW3_PORT == 0)   play_SW3();
85              else                __delay_ms(20);
86              lastSW3 = SW3_PORT;
87          }
88
89          if(SW4_PORT != lastSW4) {
90              if(SW4_PORT == 0)   play_SW4();
91              else    __delay_ms(20);
92              lastSW4 = SW4_PORT;
93          }
94      }
95  }
```

デジタル演習ボードの活用（１１）

ビルド、書き込み、実行

SW を押して動作を確認します。

次期デジタル演習ボードへの要望

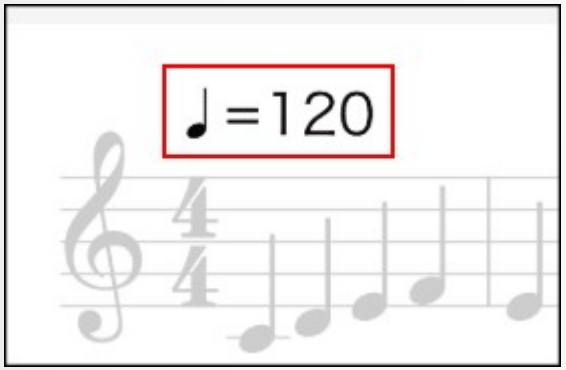
書き込むときの PICkit3 が裏返し。 LED が見たかったりします。

5V 入力にモバイルバッテリーを使うと相性があるようです。

デジタル演習ボードの活用（補足１）

```
void setTempo(uint16_t val)
```

val: 音を鳴らす基準時間をセットする
１分間に四分音符が入る個数



← これ
音符が二分音符のときは倍数を入れます。

デジタル演習ボードの活用（補足２）

```
void setOctave(int8_t val)
```

val: 440Hz を含むオクターブエリアからのオフセット値
-2 から 5 までを指定します

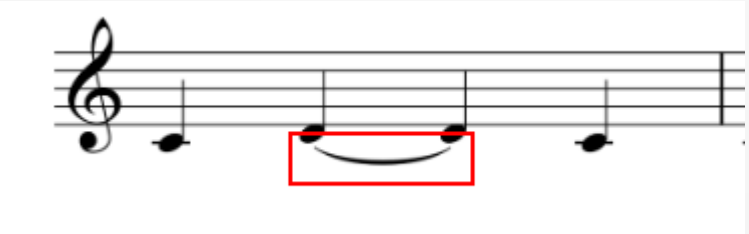
右図の左端の音符の高さ、
つまりドから上のシまでの音符を
指定するときに setOctave(0); とし、
1 オクターブ上なら setOctave(1); で、
1 オクターブ下なら setOctave(-1); です。

デジタル演習ボードの活用（補足３）

```
void setTieCount(int8_t val)
```

val: 音をつなげる時に直前で呼び、繋ぐ個数を指定します

音符と音符の間は10%程度無音時間を設定しているので、
これを無くすときに使います。下図がタイの記号です。
スラーのときもこの関数をつかいます。



デジタル演習ボードの活用（補足４）

```
void sound(pitch pit, note no)
```

pit:　音の高さを指定します

no:　音の長さを指定します。

どちらの引数も enum で記述しているので、読んだとおりにごく近い表現で記述できます。

先頭の４音は
「ミ、ミ、（休符）、ミ」
で長さは全て八分音符です。
enum は sound.h に置いていて参照を楽にしています。
ほとんどの作業は行のコピーと編集で行えます。

```
10  //-----------------------------------------------
11  // スーパーマリオ　テーマ１
12  //-----------------------------------------------
13  static void mario_overworld1(uint16_t tempo)
14  {
15      setTempo(tempo);            setOctave(1);
16
17      sound(p_Mi, eighth);
18      sound(p_Mi, eighth);
19      sound(rest, eighth);
20      sound(p_Mi, eighth);
```