情報工学基礎実験 組み込みシステム基礎

報告者

学生番号: 22122003 氏名: Shunnei Awano

メールアドレス: b2122003@edu.kit.ac.jp

実験実施日

実験週 (① ② 3 4 5)

実施日 2023 年 10 月 11 日

報告書提出日 2023 年 10 月 16 日 受理 / 要再提出

(再提出) 2023 年 10 月 19 日 受理 / 要再提出

(再提出) ＿＿年＿月＿日 受理 / 要再提出

自己チェック欄

☐ 実験結果は示されているか？ ☐ 図表の書き方は正しいか？

☐ 考察は十分か？ ☐ 演習問題はできたか？

☐ プログラムは見やすく示しているか？ ☐ 課題番号等が記述されているか？

☐ 構成はテキストの指示通りになっているか？

教員所見 (redmine で指摘された内容を報告者がここにコピーして下さい)	報告者の回答

目次

1 実験機材

- Tiva C Series LaunchPad(通し番号 03)
- DELL XPS13 9305(持ち込み PC)
- OS は Windows 11

2 理論

実験および課題に使用した技術，デバイスは PWM，I^2C，マイコンボード，LED，ブザー，LCD である．

2.1 PWM(Pulse Width Modulation)

周期一定の矩形波を出力し続けながら，パルス幅を変化させる機能である．早い周期でスイッチングを行うことで、オンのパルス幅に比例した任意の電圧が得られる．本実験では，ブザーを鳴らす，LED の光量を変化させる際にこの機能を利用している．

2.2 I^2C(Inter-Integrated Circuit)

低速な周辺機器をマザーボードへ接続するのに用いられている．I^2C は 1 つの master で複数の slave デバイスと通信することが可能な為、省配線や省スペース化が容易に実現可能. 一方で，通信速度は標準で 100kbps，最大で 400kbps と高速・大量のデータ転送には不向き．本実験では LCD の制御にこの技術を使用している．

2.3 マイコンボード (Tiva C Series LaunchPad)

本実験で利用するのは EK-TM4C123GXL というボード．インターフェースは電源スイッチ，デバック用 USB コード，スイッチ 2 つなど．その他にも RGBLED や LCD, ヘッダーピンが取り付けられている．80MHz で動作する 32bit ARM Cortex-M4 プロセッサ TM4C123GH6PM が搭載されており，256KB のフラッシュメモリ，32KB の SRAM を持ち，各種 I/O を制御することができる．システムクロックには，16MHz の水晶発振器を用意している．

2.4 LED

RGBLED はポート F に取り付けられている．GPIO() 関数により，点等・消灯を行う．詳しくは §4.2 を参照.

2.5 ブザー

GPIO ピン PB5 に接続されており，PWM 機能を利用してパルス波形を出力することによって音を鳴らす．繰り返しの周期を適切に設定することで，音階を表現する．一般的な音階の周波数が

$$f = 440 \times 2^{\frac{n}{12}} \tag{1}$$

で与えられるので，周期 T は

$$T = \frac{1}{f} = \frac{1}{440 \times 2^{\frac{n}{12}}} \tag{2}$$

で与えられる．式 (1),(2) における n と音階の関係を以下の図に示す．

音階 f	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ	ド
n	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3

図 1　n と音階 f の関係

2.6 LCD (Liquid Crystal Display)

キャラクタ液晶ディスプレイを指す．電圧をかけると分子の向きが変わる液晶の性質を利用し，光の透過率を制御することによって表示を行う．

3 実験内容

3.1 課題 1 仮想シリアルポートの利用

blinky_main.c において，UARTprintf() のコメントアウトを解除し，仮想シリアルポートへの出力を CCS(Code Composer Studio) の Terminal で確認した．

3.2 課題 2 プッシュスイッチの利用

GPIO 割り込みを用いてプッシュスイッチ (SW1) を押すたびに，LED の色が青→赤→緑→白→青と変化するようにプログラムを書き換えた．

3.3 課題 3 ブザーの制御

GPIO 割り込みを用いてプッシュスイッチ (SW1) を押すたびに，ブザーの音階がド→消音→レ→消音→ミ→消音→ファ→消音→ソ→消音→ド→消音→... となるようにプログラムを書き換えた.

3.4 課題 4 LCD 表示関数の実装

LCD の文字の表示位置を指定する setAddressLCD 関数と，LCD に文字を表示する writeTextLCD 関数をそれぞれ実装し，LCD に正しく表示されるようにプログラムを一部書き換えた.

3.5 課題 5 割り込みによる LCD の表示変更

GPIO 割り込みを用いてプッシュスイッチ (SW1) を押すたびに，LCD に押した回数を表示するように処理を書き加えた.

4 実験結果

4.1 課題 1 仮想シリアルポートの利用

UARTprintf(”Hello,world!\n”); の Terminal における出力結果は以下のようになった.

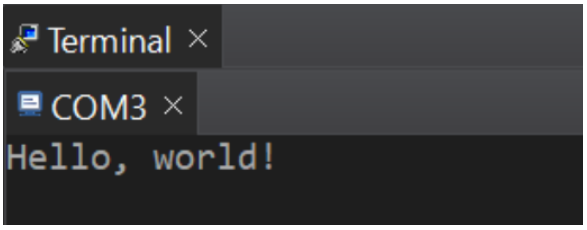

図 2 【課題 1】仮想シリアルポートへの出力

4.2 課題 2 プッシュスイッチの利用

initInterruptPins 関数と SW1PinIntHandler 関数はそれぞれ以下のようになった.

```
1   void initInterruptPins(void) {
2
3     // You can write your own code here.
4
5     // Clear Interrupt
6     GPIOIntClear(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
7
8     // Register a handler function
9     GPIOIntRegister(GPIO_PORTF_BASE,SW1PinIntHandler);
10
11    // Set type of interrupt to falling edge
12    GPIOIntTypeSet(GPIO_PORTF_BASE,INT_ALL_BUTTONS,
          GPIO_FALLING_EDGE);
13  }
```

ソースコード 1　【課題 2】blinky_main.c における initInterruptPins 関数

```
1   void SW1PinIntHandler(void) {
2
3   //static int cnt=0;
4     cnt++;
5
6     GPIOIntDisable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
7
8     GPIOIntClear(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
9
10    GPIOPinWrite(GPIO_PORTF_BASE,base_led_color,0); // turn off all
          LEDs
11
12    switch(cnt%4){
13
14        case 1:base_led_color = LED_RED;
15        break;
16
17        case 2:base_led_color = LED_GREEN;
18        break;
19
20        case 3:base_led_color = LED_WHITE;
21        break;
22
23        case 0:base_led_color = LED_BLUE;
24        break;
25
26     }
27
28    GPIOIntEnable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
29
30   }
```

ソースコード 2　【課題 2】blinky_main.c における SW1PinIntHandler 関数

main 関数に以下の一行を書き加えて実行すると，無事プッシュスイッチ (SW1) を押すたびに，LED の色が青→赤→緑→白→青と変化するようになった．

```
1    GPIOIntEnable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
```

ソースコード 3　【課題 2】blinky_main.c において main 関数に書き加えたコード

4.3　課題 3 ブザーの制御

toneBuzzer 関数と restBuzzer 関数は以下のようになった．

```
1    void toneBuzzer(int tone) {
2
3      PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, tone);
4      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, tone>>1);
5      PWMOutputState(PWM0_BASE, PWM_OUT_3_BIT, true);
6
7    }
8
9    void restBuzzer() {
10
11     PWMOutputState(PWM0_BASE, PWM_OUT_3_BIT, false);
12
13   }
```

ソースコード 4　【課題 3】buzzer.c における toneBuzzer 関数と restBuzzer 関数

また，SW1PinIntHandler 関数は以下のようになった．

```
1   void SW1PinIntHandler(void) {
2
3     //static int cnt = 0;
4     cnt++;
5
6
7       GPIOIntDisable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
8
9       GPIOIntClear(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
10
11
12      switch(cnt%10){
13
14          case 1:toneBuzzer(O4C);
15          break;
16
17          case 3:toneBuzzer(O4D);
18          break;
19
20          case 5:toneBuzzer(O4E);
21          break;
22
23          case 7:toneBuzzer(O4F);
24          break;
25
26          case 9:toneBuzzer(O4G);
27          break;
28
29          default:restBuzzer();
30          break;
31
32      }
33
34      GPIOIntEnable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
35  }
```

ソースコード 5　【課題 3】buzzer_main.c における SW1PinIntHandler 関数

これらを実行することで，プッシュスイッチ (SW1) を押すたびに，ブザーの音階がド→消音→レ→消音→ミ→消音→ファ→消音→ソ→消音→ド→消音→. . . とブザーを鳴らすことに成功した．

4.4 課題 4 LCD 表示関数の実装

setAddressLCD 関数と writeTextLCD 関数は以下のようになった.

```
1
2               inline void setAddressLCD(uint8_t x, uint8_t y) {
3
4                   /* write your code */
5
6                   uint8_t current = 0x00;
7
8                   if(y==0){
9                       current = current + x;
10                  }else{
11                      current = 0x40 + x;
12                  }
13
14                  uint8_t cmd[] = {SB1602_COMMAND_SINGLE,0x80 | current};
15
16                  writeDataI2C(I2C3_BASE,SB1602_SLAVE_ADDRESS,cmd,2);
17
18              }
19
20              inline void writeTextLCD(uint8_t *text, uint8_t length) {
21
22                  /* write your code */
23
24                  uint8_t cmd[] = {SB1602_DATA_BURST};
25
26                  uint8_t i;
27
28                  for(i=1;i<=length;i++){
29
30                      cmd[i]=text[i-1];
31
32                  }
33                  writeDataI2C(I2C3_BASE,SB1602_SLAVE_ADDRESS,cmd,length+1);
34
35              }
```

ソースコード 6　【課題 4】lcd_SB1602.c における setAddressLCD 関数と writeTextLCD 関数

上記 2 つの関数を実装した後，lcd_main.c の一部を以下のように書き換え，実行することで LCD 上に「シャキーン」と「ショボーン」を出力させた．

```
1       uint8_t face1[]={'(',0xF3,0xA5,'w',0xA5,0xF4,')'};
2       uint8_t face2[]={'(',0xF4,0xA5,'w',0xA5,0xF3,')'};
3       uint8_t msg1[]= {0xBC,0xAC,0xB7,0xB0,0xDD,'!'};
4       uint8_t msg2[]= {0xBC,0xAE,0xCE,0xDE,'—',0xDD};
```

ソースコード 7　【課題 4】lcd_main.c におけるシャキーン!とショボーンのパーツを格納する配列

```
1
2       setAddressLCD(4,0);
3       if (face == 0) {
4           writeTextLCD(face1,7);
5       } else {
6           writeTextLCD(face2,7);
7       }
8       setAddressLCD(5,1);
9       if (face == 0) {
10          writeTextLCD(msg1,6);
11      } else {
12          writeTextLCD(msg2,6);
13      }
14      face = 1 − face;
```

ソースコード 8　【課題 4】lcd_main.c におけるシャキーン!とショボーンの表示

4.5 課題 5 割り込みによる LCD の表示変更

SW1PinIntHandler 関数は以下のようになった.

```
1   void SW1PinIntHandler(void){
2
3     //static int count = 0;
4
5     count++;
6     int length = 2;
7     if(count>=100){
8       length = 3;
9     }
10
11
12
13
14    GPIOIntDisable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
15
16    GPIOIntClear(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
17
18    setAddressLCD(5,0);
19    writeTextLCD(itoa(count,length),3);
20
21
22    GPIOIntEnable(GPIO_PORTF_BASE,INT_ALL_BUTTONS);
23
24  }
```

ソースコード 9　【課題 5】lcd_main.c における SW1PinIntHandler 関数

実行すると，プッシュスイッチ (SW1) を押すたびに，LCD に押した回数が表示されるようになった.

5 考察

5.1 実験および課題に対する考察

実験においてGPIO割り込みを用いてブザーやLEDを制御する際，ボタンを速く連打すると音がずれてしまったり，LEDの色がおかしくなったりしたことがあった．(もちろん本レポートに記載しているコードはそのような問題はクリアしている) 単純に割り込みの順番が間違っていた可能性が高いが，もっと速く (機械などで) 連打した場合はディレイを別途関数で調整する必要があると考えられる．

5.2 考察 1

Systick割り込みは「時間変化」を，GPIO割り込みは「スイッチが押されたかどうか」を，LEDの点滅やLCDの表示などに反映させる役割を持つ．

5.3 考察 2

I^2C は信号線がSDAとSCLの2本で済むため配線の簡単化を可能にするとともに，この技術を用いることで，複数機器との接続，master/slave構造により，異なるデバイスを協調して動作させ，複雑な処理を実現できるという利点が存在する．

5.4 考察 3

割り込み駆動のアプローチは、イベントが発生した瞬間にセンサデータを読み込むため、システムとしてリアルタイム性の向上が見込めるとともに，CPUが常にセンサを監視する必要がなくなるため，処理負荷の軽減，他タスクへのリソース分配が見込めるという利点がある．

6 問題回答

6.1 問 1

RGBそれぞれに対して，ONとOFFの2通り．3色ともOFFの時は考慮しないので，表現できる色は $2^3 - 1 = 7$(通り)．よって7色表現できる．

6.2 問 2

問1と同じように考えて，$(2^8)^3 - 1 = 16777215$(通り)．よって16777215色表現できる．

6.3 問 3

「1 オクターブ高くする」には周波数を 2 倍にすることが必要である．よって，周波数の逆数である周期は 1/2 倍されるべきである．

6.4 問 4

プルアップしていない場合，スイッチがオフの状態時に入力端子がどこにも繋がっていない，浮いた状態となり外部のノイズなどの影響を受けやすくなってしまう．それを防ぐため．

6.5 問 5

本実験で用いた I^2C のアドレススペースは 7bit である．したがって理論上は最大 $2^7 = 128$ 個の機器に接続可能である．

6.6 問 6

delay_ms 関数

ミリ秒単位の時間遅延 (処理待機時間) を導入するための関数．

itoh 関数

32 ビットの符号なし整数を 16 進数の ASCII 文字列に変換するための関数．ASCII 文字列へのポインタを返す．

itoa 関数

32 ビットの符号付き整数を文字列に変換するための関数．文字列へのポインタに加え，出力文字列の長さも返す．

参考文献

1. 『PWM とは』 東芝デバイス&ストレージ株式会社
   https://toshiba.semicon-storage.com/jp/semiconductor/knowledge/e-learning/brushless-motor/chapter3/what-pwm.html
2. 『I2C(Inter-Integrated Circuit) について』 東阪電子機器株式会社
   https://tohan-denshi.co.jp/technical-information/1835/
3. 『用語集 オクターブ』 Allisone
   http://www.allisone.co.jp/html/Notes/glossary/
4. 『LCD (Liquid Crystal Display)』 EIZO 株式会社
   https://www.eizo.co.jp/support/glossary/a/lcd_i/