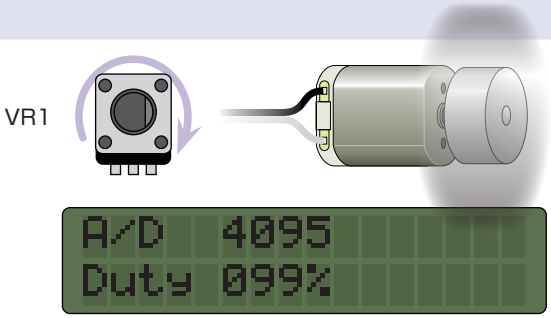
LCD 表示

学習内容

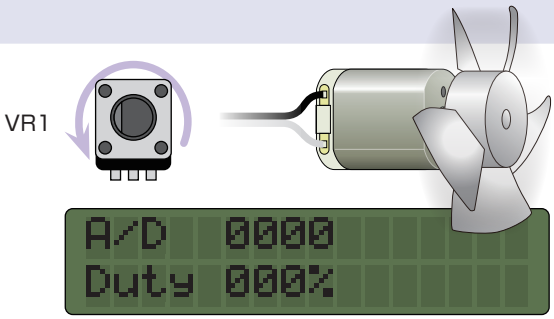
LCD に文字を表示する方法を学習しましょう。
文字表示ができるようになるとデバッグ時にも有効です。

課題 11-1

DC モータを以下のように制御し，A/D 変換値をデューティ比を LCD に表示させてみましょう。
また，システムタイマを用いた割り込みを使ってみましょう。



VR1 を右に回すと，
DC モータの回転がだんだん速くなる。
A/D 変換値とデューティ比を液晶に
表示させる。
最大値は左絵の通り。

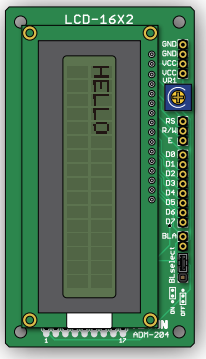


VR1 を左に回すと，
DC モータの回転がだんだん遅くなる。
A/D 変換値とデューティ比を液晶に
表示させる。
最小値は左絵の通り。

LCD 表示

配線 11-1

今回はマイコンボードと LCD ボード，ボリュームボード，DC モータボードを使用します。これらのボードはベースボード上で以下のように接続されています。LCD ボードの配線や制御方法について詳しくは，巻末に掲載しているエレモの取扱説明書をご覧ください。

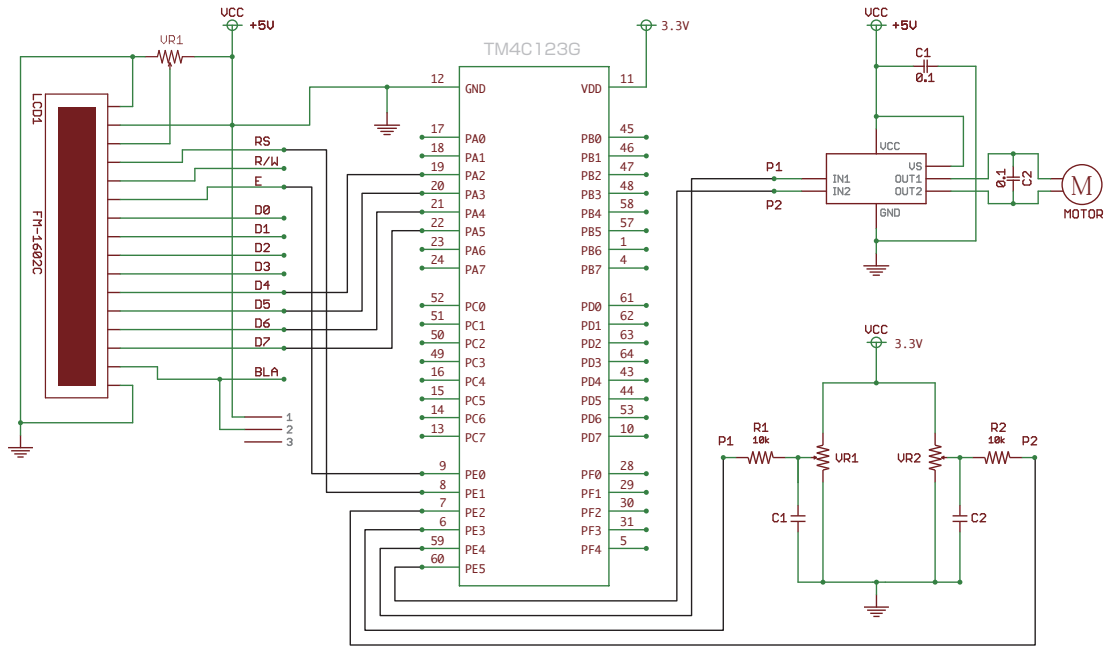


LCD ボード	ボリュームボード	マイコンボード		DC モータボード
RS		3.07	PE1	
R/W				
E		2.03	PE0	
D4		2.10	PA2	
D5		2.09	PA3	
D6		2.08	PA4	
D7		1.08	PA5	
	P1 - VR1	3.09	PE3（AIN0）	
	P2 - VR2	3.08	PE2（AIN1）	
		1.05	PE4（M0PWM4）	P1 - IN1
		1.06	PE5（M0PWM5）	P2 - IN2
	VCC	1.01	+3.3V	
GND	GND	3.02	GND	GND
VCC		3.01	VBUS（+5V）	VCC

マイコンボードと各ボードをベースボードから取り外し，ジャンパワイヤ（別売）で直接接続することも可能です。時間に余裕のある場合は，STEP 01 のピンアサインに注意しつつ，お好みのピンで課題を実現してみましょう。GND はマイコンボードの 2.01 ピン，3.02 ピンのどちらでもかまいません。

LCD 表示

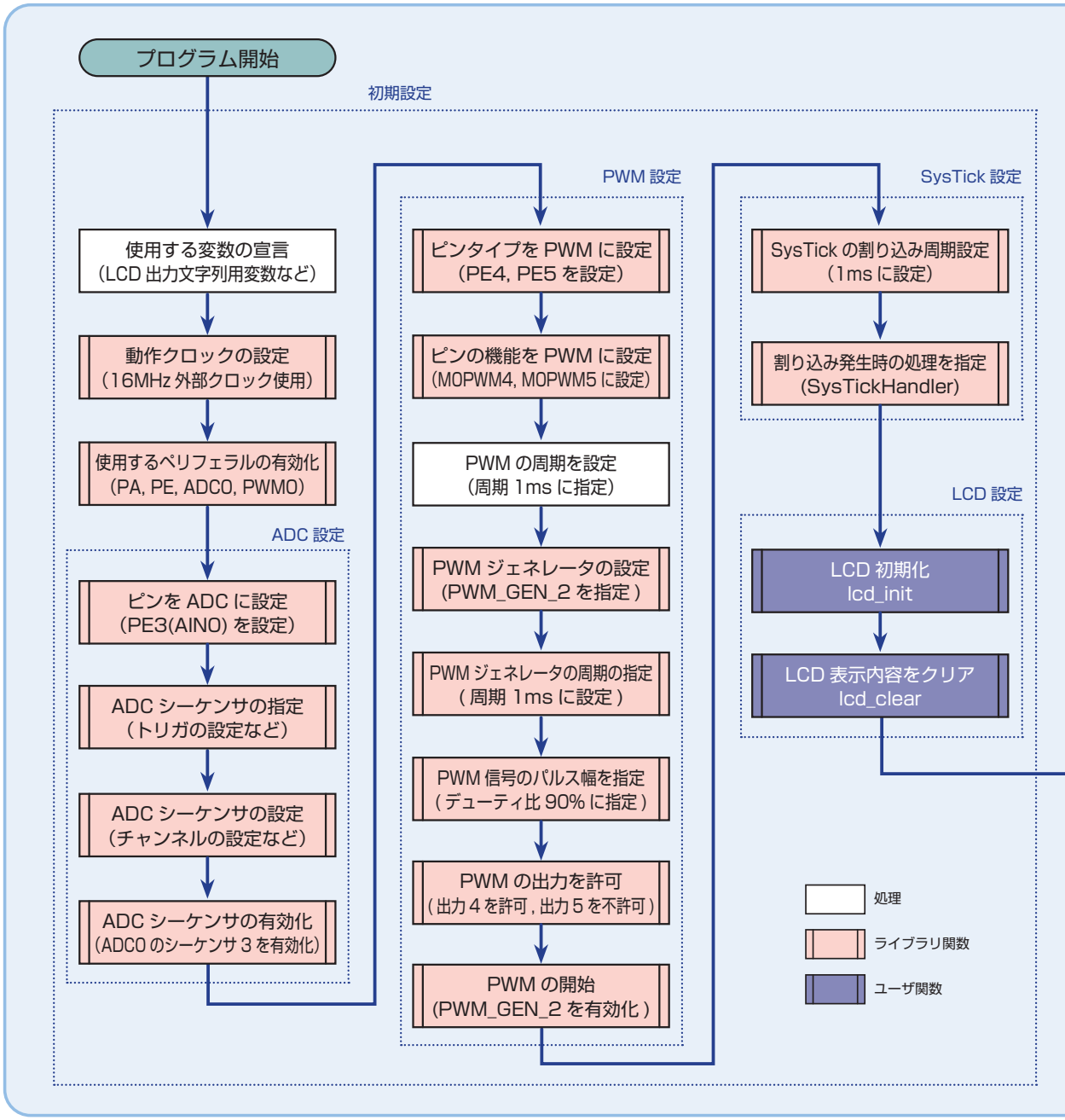
回路図


VCC
+5V
VR1
LCD1
FM-1602C
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
BLA
TM4C123G
3.3V
GND
VDD
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PE0
PE1
PE2
PE3
PE4
PE5
PF0
PF1
PF2
PF3
PF4
C1
0.1
P1
P2
IN1
IN2
VS
OUT1
OUT2
C2
MOTOR
R1
10k
R2
VR1
VR2

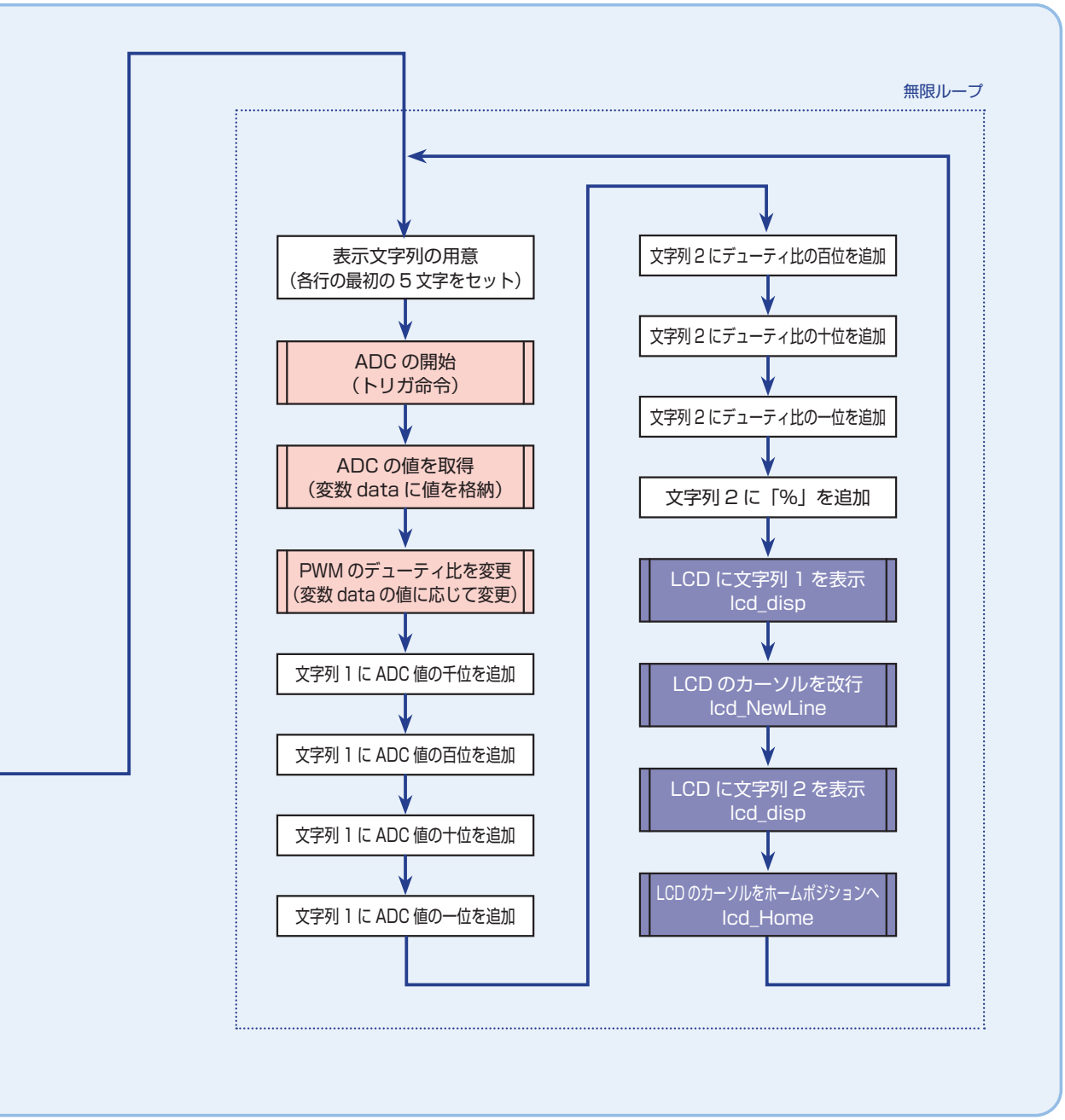
LCD 表示

フローチャート 11-1

フローチャートは以下のようになります。

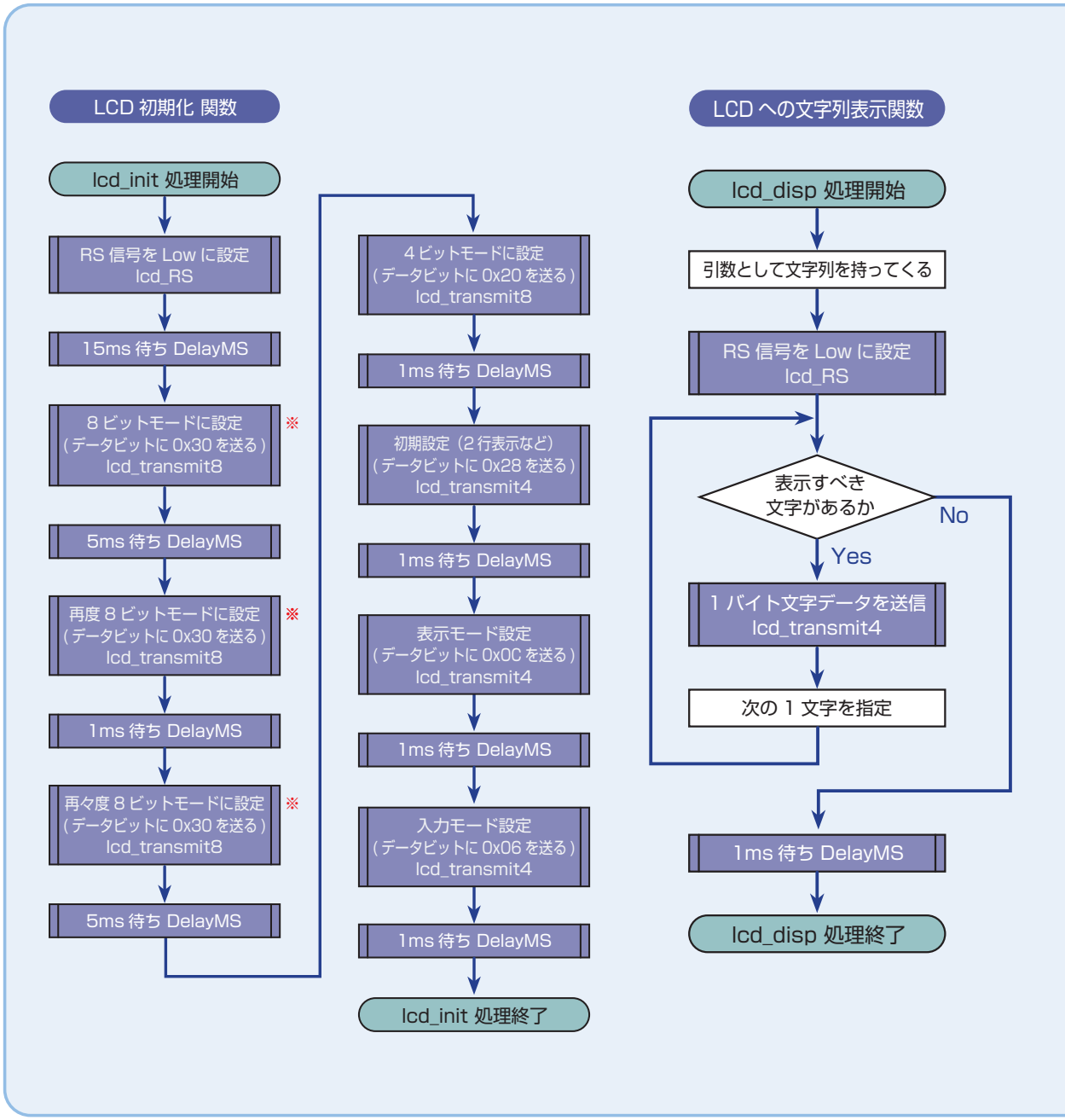


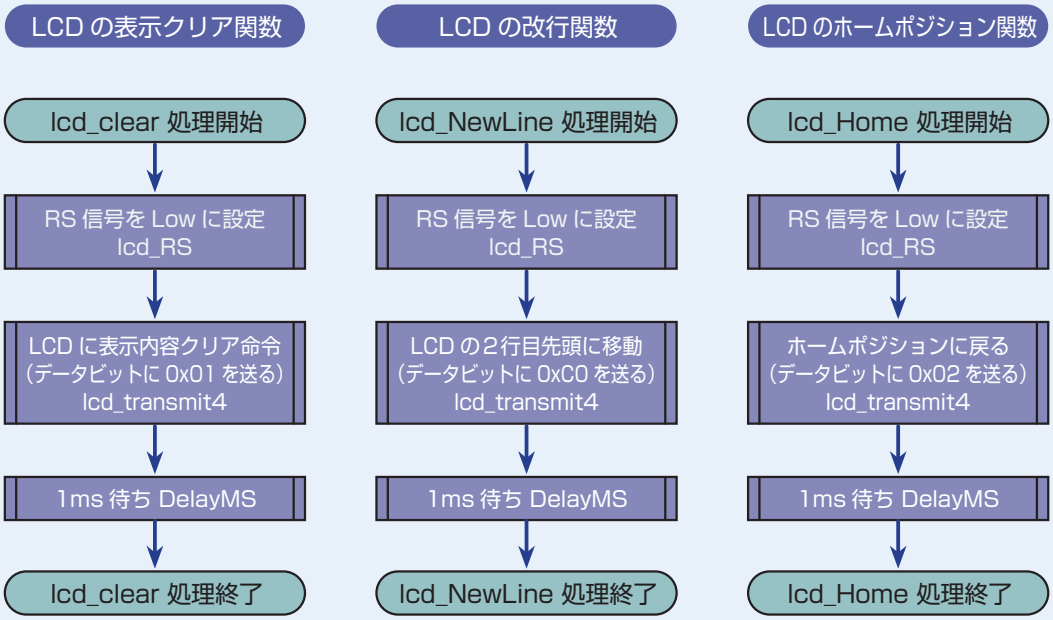
LCD 表示


無限ループ
表示文字列の用意
（各行の最初の 5 文字をセット）
ADC の開始
（トリガ命令）
ADC の値を取得
（変数 data に値を格納）
PWM のデューティ比を変更
（変数 data の値に応じて変更）
文字列 1 に ADC 値の千位を追加
文字列 1 に ADC 値の百位を追加
文字列 1 に ADC 値の十位を追加
文字列 1 に ADC 値の一位を追加
文字列 2 にデューティ比の百位を追加
文字列 2 にデューティ比の十位を追加
文字列 2 にデューティ比の一位を追加
文字列 2 に「%」を追加
LCD に文字列 1 を表示
lcd_disp
LCD のカーソルを改行
lcd_NewLine
LCD に文字列 2 を表示
lcd_disp
LCD のカーソルをホームポジションへ
lcd_Home

LCD 表示

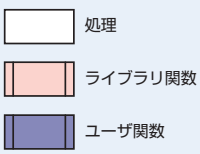
フローチャート 11-1





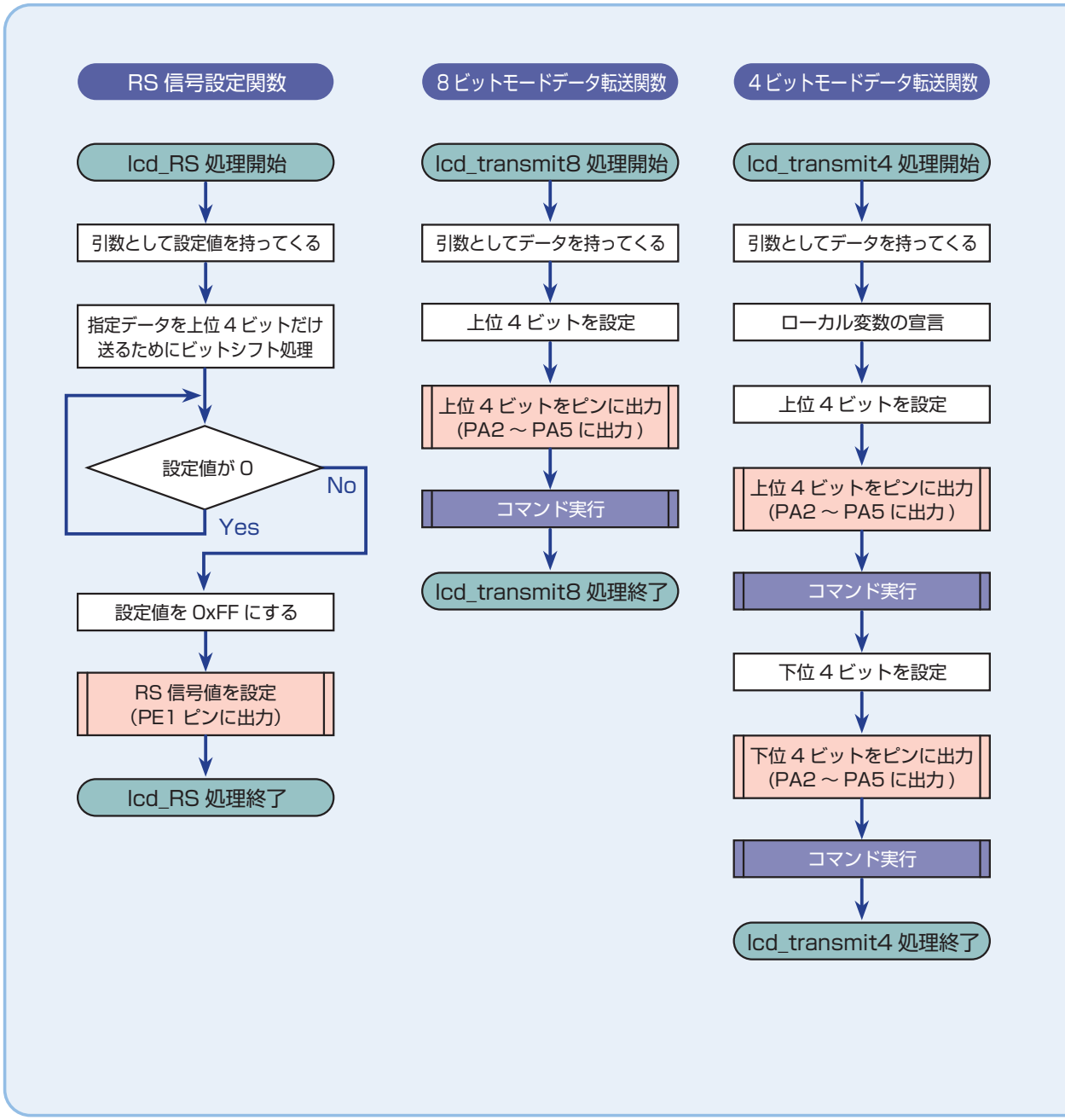
※

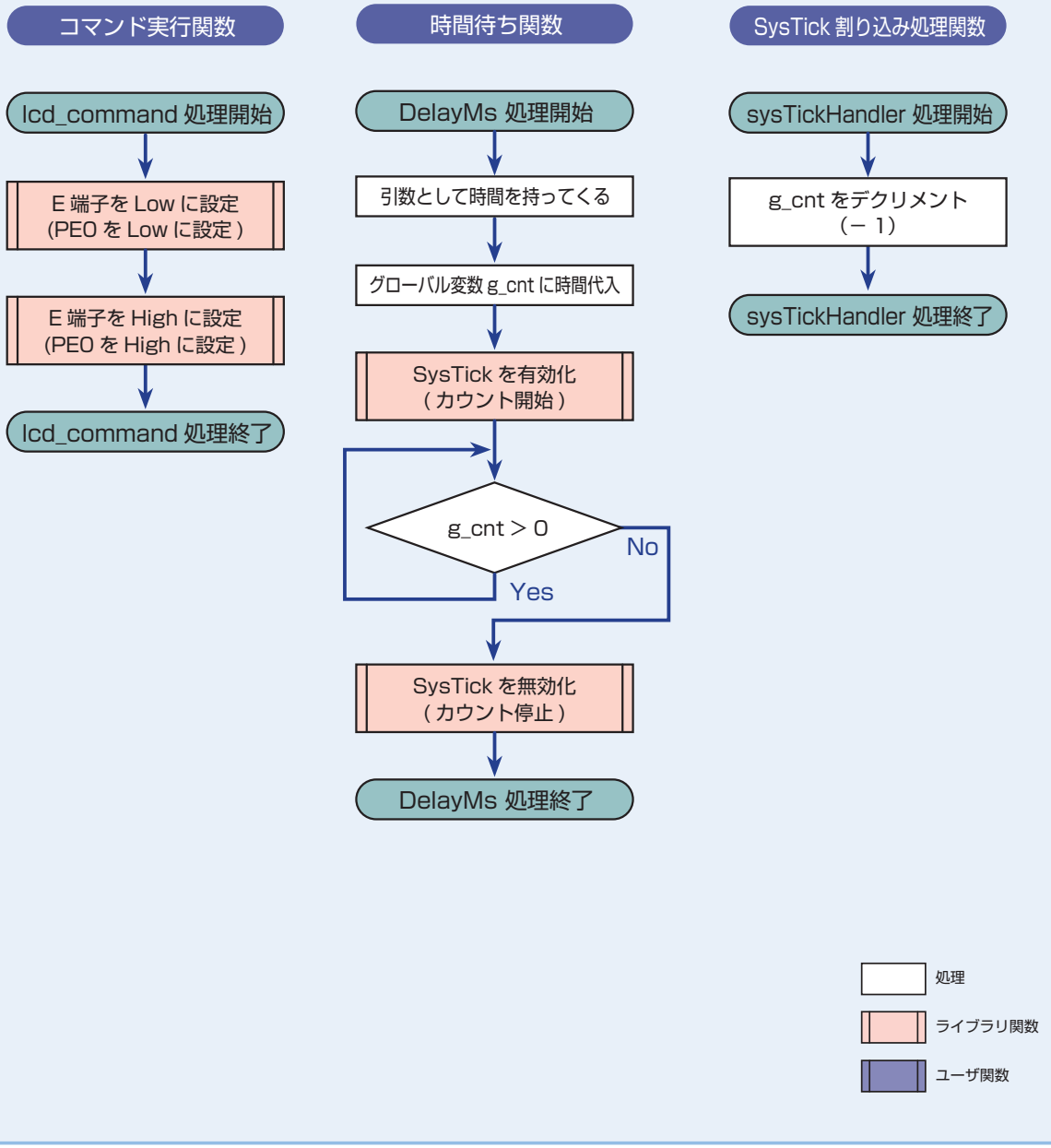
0x30 を（ある時間間隔を空けて）3 回設定するのは、LCD を初期化（リセット）するためであり、本キットで使用する LCD がそのような仕様になっています。
リセット後に LCD は 8 ビット転送モードになりますが、0x30 を送る直接の目的は LCD のリセットのためです。
テキストにある「8 ビットモードに設定」という言葉は、やや誤解を招きやすいすかもしれませんが、単に 0x30 を送ってリセットしているとご理解ください。



LCD 表示

フローチャート 11-1




コマンド実行関数
lcd_command 処理開始
E 端子を Low に設定
(PE0 を Low に設定)
E 端子を High に設定
(PE0 を High に設定)
lcd_command 処理終了
時間待ち関数
DelayMs 処理開始
引数として時間を持ってくる
グローバル変数 g_cnt に時間代入
SysTick を有効化
(カウント開始)
g_cnt > 0
No
Yes
SysTick を無効化
(カウント停止)
DelayMs 処理終了
SysTick 割り込み処理関数
sysTickHandler 処理開始
g_cnt をデクリメント
(− 1)
sysTickHandler 処理終了
処理
ライブラリ関数
ユーザ関数

LCD 表示

インクルードファイル 11-1

STEP 11-1 で使用するインクルードファイルを解説します。

標準ヘッダファイル string.h

```
#include <string.h>
```

C 言語の標準ヘッダファイルのひとつで，文字列の操作のための関数が宣言されています。解答ソース例では，strcpy() と strcat() を用いています。

Tiva ヘッダファイル driverlib/systick.h

```
#include "driverlib/systick.h"
```

Tiva のヘッダファイルで，システムタイマ（Systick）のための関数が宣言されています。本 STEP で示すライブラリ関数は，いずれもここで宣言されています。

ライブラリ関数　11-1

STEP 08 では汎用タイマによる割り込みを用いましたが，今回はより簡素なシステムタイマ（SysTick）による割り込みを用いることにします。

システムタイマ（SysTick）

簡単な 24 ビットカウントタイマ。Timer0 などに比べて設定項目が少ないため使いやすい。
反面，細かい設定ができず周期しか指定できない。

タイマによる割り込みを利用する場合，汎用タイマでは

1. SysCtlPeripheralEnable() で使用する汎用タイマモジュールを有効にする
2. TimerConfigure() でタイマの使用方法を設定
3. TimerLoadSet() で割り込みの周期を指定
4. TimerIntRegister() で呼び出す関数を指定
5. IntEnable() で割り込みを許可する
6. TimerIntEnable() で割り込み条件を有効にする
7. TimerEnable() でタイマを有効にする

という手順が必要でしたが，システムタイマでは

1. SysTickPeriodSet() で割り込みの周期を指定
2. SysTickIntRegister() で呼び出す関数を指定
3. SysTickEnable() でシステムタイマを有効にする

という手順だけで利用できます。なお，システムタイマは 24 ビットなので，

$2^{24} = 16777216$

より，動作クロックが 16MHz の場合は，最長で 1 秒強の計測が可能です。

LCD 表示

ライブラリ関数 11-1

STEP11 で使用するライブラリ関数を解説します。
関数の引数は ui32 ～が「符号無し 32 ビット整数」，*pfn ～が「関数（のポインタ）」です。ライブラリ関数は TivaWare で提供されています。

SysTick 周期を設定する SysTickPeriodSet(ui32Period)

System Timer (SysTick) の周期を設定する。

```
SysTickPeriodSet(SysCtlClockGet()/1000);
```

設定項目 例

- SysCtlClockGet()/1000 ：割り込みが発生する周期の指定。
  指定方法は Timer0 と同様。p.58 参照。

割り込み時に呼び出す関数を指定する SysTickIntRegister(*pfnHandler)

割り込み時に呼び出す関数を指定する。

```
SysTickIntRegister(SysTickIntHandler);
```

設定項目 例

- SysTickIntHandler ：割り込み時に呼び出す関数名を指定。

SysTick を有効にする SysTickEnable( )

System Timer (SysTick) を有効にする。

```
SysTickEnable();
```

SysTick を無効にする SysTickDisable( )

System Timer (SysTick) を無効にする。

```
SysTickDisable();
```

コーディング 11-1

フローチャートを元に，ソースを記述してください。ソースが完成したら，実行して動作を確認しましょう。
以下に解答例ソースを示します。解答例やサンプルソースを参考に，皆さんで工夫してみてください。

step11-1.c　CD-ROM の「サンプルソース」フォルダに、各ステップの c ファイルを収録しています

```
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "inc/hw_ints.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "driverlib/interrupt.h"
#include "driverlib/adc.h"
#include "driverlib/pin_map.h"
#include "driverlib/pwm.h"

// 待ち時間用
volatile long g_cnt = 0;

// 割り込み処理
void SysTickHandler(void) {
        // g_cnt をデクリメント（-1）
        g_cnt--;
}

// 時間待ち
void DelayMs(unsigned long ms) {
        // カウント（時間）セット
        g_cnt = ms;
        // SysTick を有効（カウント開始）
        SysTickEnable();
        // g_cnt が 0 より小さくなるまでループ（割り込みルーチンで g_cnt をデクリメント）
        while (g_cnt > 0)
                ;
        // SysTick を停止（カウント停止）
        SysTickDisable();
}

// RS 信号の設定
void lcd_RS(unsigned char value) {
        if (value != 0) value = 0xFF;
        GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_1, value);
}

```

LCD 表示

コーディング　11-1

```
// 設定したコマンドを実行（E 端子を L → H → L と切り替え）
void lcd_command() {
        GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_0, 0xFF);
        GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_0, 0x00);
}

// データ転送（8 ビットモード）
void lcd_transmit8(unsigned char data) {
        // 上位 4 ビットを設定（下位 4 ビットの端子はオープン）
        data = data >> 2;
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5, data);
        lcd_command();
}

// データ転送（4 ビットモード）
void lcd_transmit4(unsigned char data) {
        unsigned char temp;
        // 上位 4 ビットを設定
        temp = data >> 2;
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5, temp);
        lcd_command();
        // 下位 4 ビットを設定
        temp = data << 2;
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5, temp);
        lcd_command();
}

// LCD の初期化命令
void lcd_init(void) {
        // RS 信号を Low に設定
        lcd_RS(0);
        // 15 ミリ秒待ち
        DelayMs(15);
        // 8 ビットモードに設定（データビットに 0011 **** = 0x30 を送る）
        lcd_transmit8(0x30);
        // 5 ミリ秒待ち
        DelayMs(5);
        // 再度 8 ビットモードに設定（データビットに 0011 **** = 0x30 を送る）
        lcd_transmit8(0x30);
        // 1 ミリ秒待ち
        DelayMs(1);
        // 再々度 8 ビットモードに設定（データビットに 0011 **** = 0x30 を送る）
        lcd_transmit8(0x30);
        // 5 ミリ秒待ち
        DelayMs(5);
        // 4 ビットモードに設定（データビットに 0010 **** = 0x20 を送る）
```

```
89          lcd_transmit8(0x20);
90          // 1 ミリ秒待ち
91          DelayMs(1);
92          // 初期設定（4 ビット・バス，2 行表示。データビットに 0010 1000 = 0x28 を送る）
93          lcd_transmit4(0x28);
94          // 1 ミリ秒待ち
95          DelayMs(1);
96
97          // 表示モードの設定
98          //（文字表示 ON，カーソルなし，ブリンクなし。データビットに 0000 1100 = 0x0C を送る）
99          lcd_transmit4(0x0C);
100         // 1 ミリ秒待ち
101         DelayMs(1);
102         // 入力モードの設定（アドレス +1，シフトなし。データビットに 0000 0110 = 0x06 を送る）
103         lcd_transmit4(0x06);
104         // 1 ミリ秒待ち
105         DelayMs(1);
106 }
107
108 // LCD の表示をクリア
109 void lcd_clear(void) {
110         // RS 信号を Low に設定
111         lcd_RS(0);
112         // クリア命令
113         lcd_transmit4(0x01);
114         // 1 ミリ秒待ち
115         DelayMs(1);
116 }
117
118 // LCD に文字列を表示
119 void lcd_disp(char* s) {
120         // RS 信号を High に設定
121         lcd_RS(1);
122         // 文字データ出力
123         while (*s) {
124                 // 1 バイト文字データを送信
125                 lcd_transmit4(*s++);
126                 // 1 ミリ秒待ち
127                 DelayMs(1);
128         }
129 }
130
131 // 1 行目から 2 行目に改行
132 void lcd_NewLine(void) {
133         // RS 信号を Low に設定
134         lcd_RS(0);
```

LCD 表示

コーディング　11-1

```
135         // カーソルを 2 行目先頭に移動（0x80 + 0x40 = 0xC0）
136         lcd_transmit4(0xC0);
137         // 1 ミリ秒待ち
138         DelayMs(1);
139 }
140
141 // LCD のカーソルをホームポジションへ
142 void lcd_Home(void) {
143         // RS 信号を Low に設定
144         lcd_RS(0);
145         // カーソルをホームポジションに戻す
146         lcd_transmit4(0x02);
147         // 1 ミリ秒待ち
148         DelayMs(1);
149 }
150
151 // ボリューム入力による DC モータ速度制御＋ LCD に A/D 値とデューティ比を表示させる
152 void main(void) {
153         // A/D 変換値格納用
154         uint32_t data;
155         // 周期設定用変数
156         unsigned long ulPeriod;
157         // 待ち時間用
158         volatile unsigned long Loop;
159         // 数値変換用（文字列の最後が常に \0 になるようにする）
160         char temp[2] = " ";
161         // LCD への出力文字列
162         char text_1[17];
163         char text_2[17];
164         // 動作クロックの設定
165         SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
166
167         // 使用するペリフェラルの有効化
168         // : ポート A，ポート E（LCD 用）
169         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
170         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
171         // : A/D 変換モジュール 0
172         SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
173         // : PWM モジュール 0
174         SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
175
176         // PA2~PA5，PE0，PE1 を出力タイプに設定
177         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);
178         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_3);
179         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_4);
180         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);
```

```
181         GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_0);
182         GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_1);
183
184         // ADC の設定
185         // : PE3(AIN0) を ADC に設定
186         GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
187         // : 使用する ADC シーケンサの指定
188         ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
189         // : ADC シーケンサの設定
190         ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH0 | ADC_CTL_IE | ADC_CTL_END);
191         // : ADC シーケンサを有効にする
192         ADCSequenceEnable(ADC0_BASE, 3);
193
194         // PWM の設定
195         // : PWM 機能で使うポートの指定
196         GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_4);
197         GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_5);
198         // : PE4 を M0PWM4 に，PE5 を M0PWM5 に設定
199         GPIOPinConfigure(GPIO_PE4_M0PWM4);
200         GPIOPinConfigure(GPIO_PE5_M0PWM5);
201         // : PWM の周期を指定　周期： 1ms
202         ulPeriod = SysCtlClockGet() / 1000;
203         // : PWM ジェネレータの設定
204         PWMGenConfigure(PWM0_BASE, PWM_GEN_2, PWM_GEN_MODE_UP_DOWN | PWM_GEN_MODE_NO_SYNC);
205         // : PWM ジェネレータの周期を設定
206         PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, ulPeriod);
207         // : PWM のデューティ比の指定
208         PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, ulPeriod * 9 / 10);
209         PWMPulseWidthSet(PWM0_BASE, PWM_OUT_5, ulPeriod * 9 / 10);
210         // : M0PWM4 の出力を許可，M0PWM5 の出力を不許可
211         PWMOutputState(PWM0_BASE, PWM_OUT_4_BIT, true);
212         PWMOutputState(PWM0_BASE, PWM_OUT_5_BIT, false);
213         // : PWM 開始
214         PWMGenEnable(PWM0_BASE, PWM_GEN_2);
215
216         // SysTick の割り込みが発生する間隔を設定
217         SysTickPeriodSet(SysCtlClockGet() / 1000);
218         // 割り込み時に呼び出す処理（関数）を指定
219         SysTickIntRegister(SysTickHandler);
220
221         // LCD 初期化
222         lcd_init();
223         // LCD 表示内容クリア
224         lcd_clear();
225
226         while (1) {
```

LCD 表示

コーディング　11-1

```
227                     // 表示文字列の用意
228                     strcpy(text_1, "A/D ");
229                     strcpy(text_2, "Duty ");
230                     // A/D 変換開始
231                     ADCProcessorTrigger(ADC0_BASE, 3);
232                     // A/D 変換値の取得
233                     ADCSequenceDataGet(ADC0_BASE, 3, &data);
234                     // A/D 変換の値に応じてデューティ比を変更
235                     PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, ulPeriod * data / 4096);
236                     // 文字列 1 に A/D 値の千位を追加
237                     temp[0] = 0x30 + (data / 1000);
238                     strcat(text_1, temp);
239                     // 文字列 1 に A/D 値の百位を追加
240                     temp[0] = 0x30 + ((data % 1000) / 100);
241                     strcat(text_1, temp);
242                     // 文字列 1 に A/D 値の十位を追加
243                     temp[0] = 0x30 + ((data % 100) / 10);
244                     strcat(text_1, temp);
245                     // 文字列 1 に A/D 値の一位を追加
246                     temp[0] = 0x30 + (data % 10);
247                     strcat(text_1, temp);
248                     // 文字列 2 にデューティ比の百位を追加
249                     temp[0] = 0x30 + ((data * 100 / 4096) / 100);
250                     strcat(text_2, temp);
251                     // 文字列 2 にデューティ比の十位を追加
252                     temp[0] = 0x30 + (((data * 100 / 4096) % 100) / 10);
253                     strcat(text_2, temp);
254                     // 文字列 2 にデューティ比の一位を追加
255                     temp[0] = 0x30 + ((data * 100 / 4096) % 10);
256                     strcat(text_2, temp);
257                     // 文字列 2 に「%」を追加
258                     temp[0] = '%' ;
259                     strcat(text_2, temp);
260                     // LCD に文字列 1 を表示
261                     lcd_disp(text_1);
262                     // 改行
263                     lcd_NewLine();
264                     // LCD に文字列 2 を表示
265                     lcd_disp(text_2);
266                     // カーソルをホームへ
267                     lcd_Home();
268             }
269 }
```

課題 11-2

この課題は応用課題です。時間に余裕のある場合はチャレンジしてみてください。

課題 10-1 で表示した LED の色を，デューティ比を参考に 16 進数の色コード（「#」に続いて赤，緑，青を2桁の 16 進数で表す）で LCD に表示してみましょう。

表示例

解答は，巻末の解答例集参照