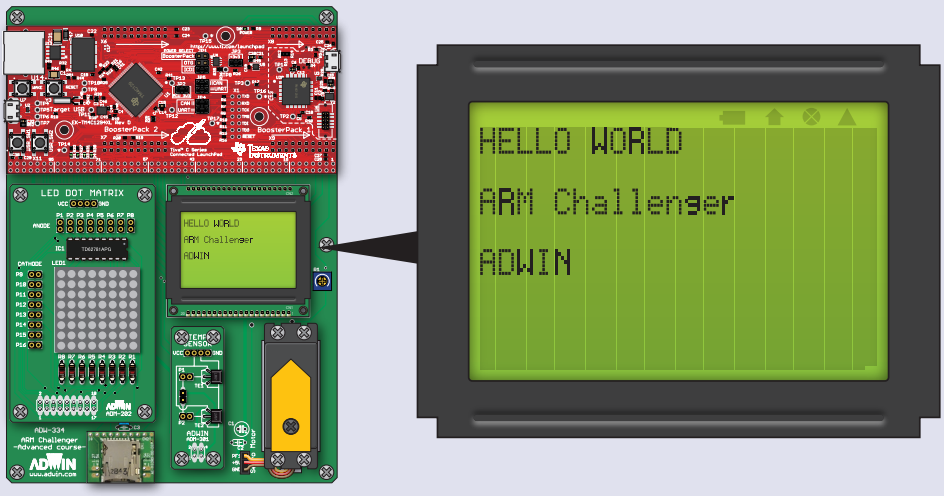
グラフィック LCD の使用

学習内容

グラフィック LCD に文字を表示させてみましょう。
以後、グラフィック LCD はステータス確認やデバッグに有効に使えます。

課題 07

グラフィック LCD に以下のように文字を表示させてみましょう。

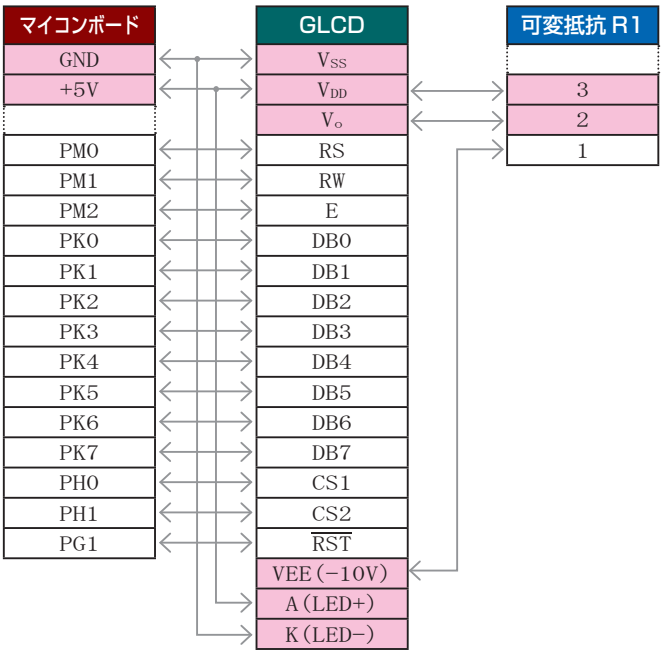


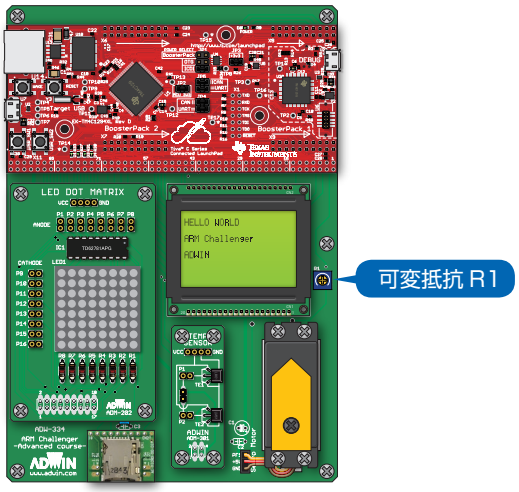
グラフィック LCD は、以後 GLCD と表記します。

GLCD の使用

配線 07

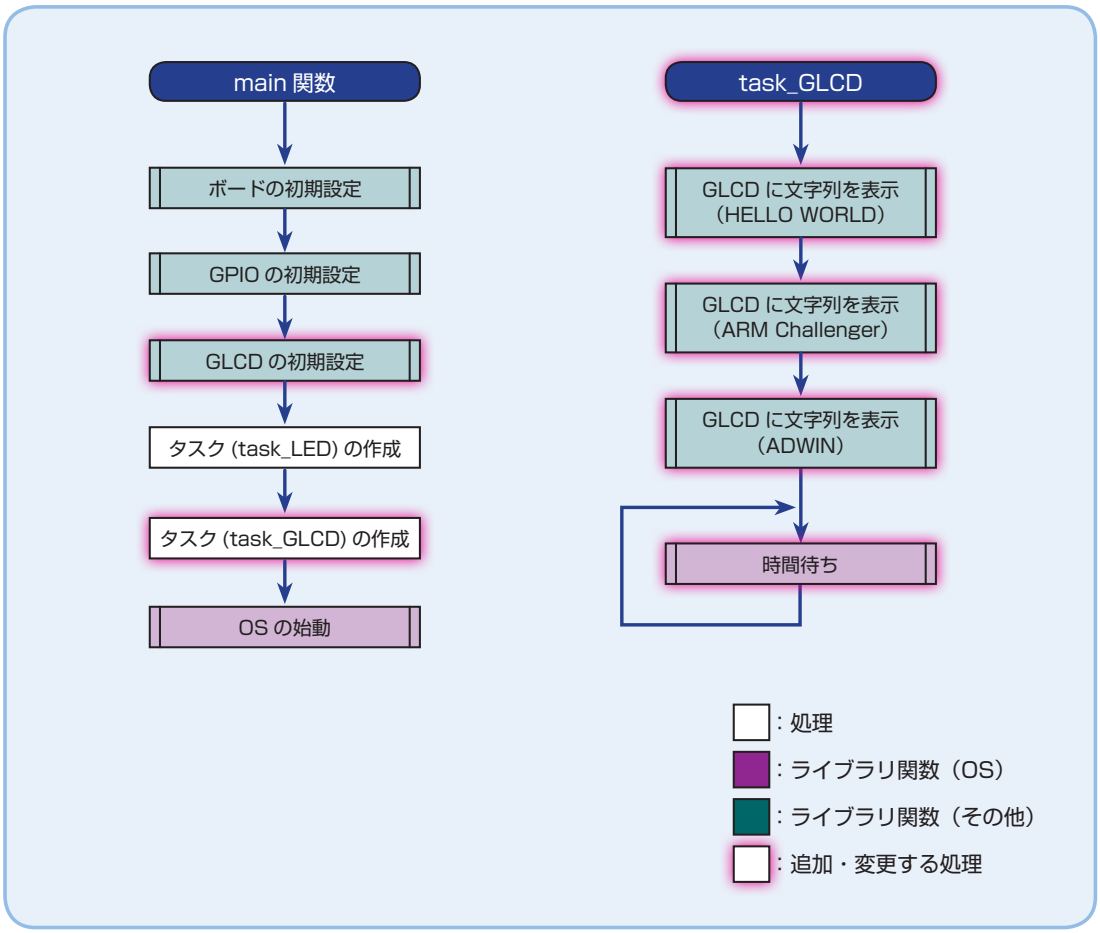
今回はマイコンボードと GLCD モジュールを使います。これらはベースボード上で以下のように配線されています。なお、可変抵抗 R1 は GLCD ディスプレイのコントラスト調整用です。





フローチャート 07

フローチャートは以下のようになります。その他のタスクは「フローチャート 06」と同じです。



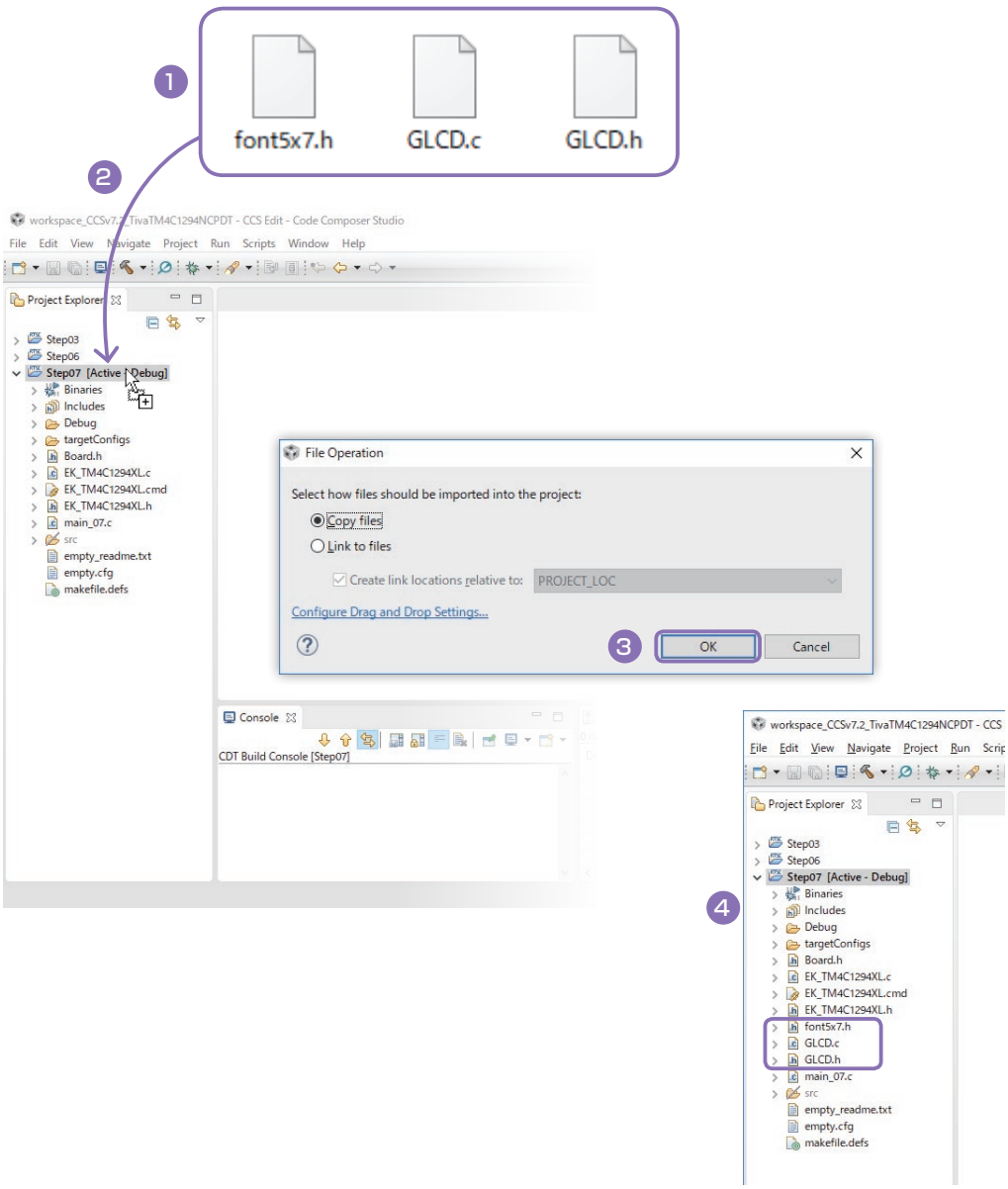
プロジェクトのコピー

本 STEP のプロジェクトは、STEP 06 のプロジェクトを複製して使うと良いでしょう。
プロジェクトのコピー方法は STEP 03 の p.21 に解説してあります。
以後、プロジェクトのコピーについて明記しませんが、必要に応じて行ってください。

GLCD の使用

GLCD ライブラリの追加

GLCD を使うためのライブラリは弊社から提供しています。
付属 CD または、弊社サイトからダウンロードした GLCD ライブラリファイルの「GLCD.c」、「GLCD.h」、「font5x7.h」をプロジェクトフォルダにドラッグ＆コピーしてください。
プロジェクトにこれらのファイルが追加されていることを確認しましょう。



インクルードファイル 07

課題 07 で使用するインクルードファイルを解説します。

なお、インクルードされているファイルは、テキストカーソルをファイル名部分に合わせて「F3」キーを押すことで開くことができます。

GLCD ヘッダファイル　GLCD.h

GLCD を使用するための、以下の関数が宣言されています。

- GLCD_init()
- GLCD_str()

GLCD の使用

ライブラリ関数 07

課題 07 で使用するライブラリ関数を解説します。

なお、テキストカーソルをソースコード中の関数に合わせると、その関数に関する情報がポップアップされます。さらに「F3」キーを押すと、その関数が定義されているファイルを開くことができます。

GLCD 関係

GLCD の初期化　void GLCD_init()

```
GLCD_init();
```

マイコンのペリフェラルのセットアップおよび GLCD の初期化を行います。文字列を表示する前に、まずこの関数を実行しておく必要があります。

さらに詳しくは、プロジェクトフォルダにコピーした GLCD.c をご覧ください。

GLDC に文字列を表示　void GLCD_str(int y, int x, char *str)

```
GLCD_str(0, 0, "HELLO WORLD");
```

引数の設定例

- `0` ：行のオフセット（1 行は 8 ドット）。1 行目に文字列を表示させる場合は 0 を指定。
- `0` ：文字（列）のオフセット（1 文字の幅は 6 ドット）。1 文字目から文字列を表示させる場合は 0 を指定。
- `"HELLO WORLD"` ：表示させる文字列。
  表示させる文字は ASCII 文字で、最後がヌル文字の char 型（8 ビット）配列でなければならない※。

任意の図形を表示させる場合については、GLCD モジュール（Vatronix 社製「TG12864E」）のデータシートをご覧ください。

※　「" ～ "」で指定した文字列は、最後にヌル文字が挿入された char 型配列に自動的に変換されます。その他の場合は、文字列の最後にヌル文字を指定する必要があり、配列の大きさもヌル文字を含めた（表示させる文字数より 1 つ増やした）大きさにする必要があります。

GLCD の使用方法および注意事項

1. GLCD の初期化をする

GLCD を使用するときは、文字列を表示する前にまず GLCD_init() を実行します。

2. 表示する文字列を準備する

変数の値を GLCD に表示する場合には、数値を文字列に変換する必要があります（表示したい数値を 1 桁ずつ文字に変換し、配列に格納します）。

3. 文字列を表示させる

表示したい文字列を直接指定したり、文字列を格納した配列を指定して表示させます。表示される文字列は 1 文字あたり 6 × 8 ドットで表示されます。

4. コントラストの調整

コントラストが適切に調整されていないと、GLCD に表示されている文字列が見えません。文字列が表示されない場合や、見えにくい場合は、GLCD 横の可変抵抗 R1 を回し、コントラストを調整してみてください。

本コースで使う GLCD は、128x64 ドットのディスプレイに任意の図形を表示することができます。本コースで用いる GLCD ライブラリは、このディスプレイを 8 行 22 文字（22 文字目は後ろ 3 ドットが欠けます）に分割し、任意の行数・文字数から ASCII 文字を入力します。

任意の図形を表示させる場合については、GLCD モジュール（Vatronix 社製「TG12864E」）のデータシートをご覧ください。

GLCD の使用

新しいタスクの作成手順 【CCS】

新しいタスクを追加する際の作成手順と注意事項です。以下の手順に従ってソースファイルに記述してください。

1 新しく作成するタスクの優先度を決める

優先度は、1 ～ 15 の整数で指定し、数値が大きいほど優先度が高くなります。なお、同じ優先度を指定した場合は、タスクの作成順に実行されることになります。
優先度はマクロ（#define）で定義しておくと良いでしょう。以下は定義例です。

```
例 : #define TASK_GLCD_PRIO 9
```

2 スタックのサイズを決める

スタックのサイズは 512 を基本とし、状況に応じてより大きなスタックサイズを指定します。
スタックのサイズも、マクロ（#define）で定義しておくと良いでしょう。以下は定義例です。

```
例 : #define TASK_GLCD_STACK 512
```

3 タスクの構造を保持する変数を宣言する

タスクの構造を保持する変数は、Task_Struct 型構造体で宣言します。
以下は、Task_Struct 型変数の宣言例です。

```
例 : Task_Struct taskGLCDStruct;
```

4 タスクのスタックを宣言する

タスクのスタックは、Char 型の配列として宣言します。配列の大きさは、先ほど決めたスタックのサイズになります。
以下はスタックの宣言例です。

```
例 : Char taskGLCDStack[TASK_GLCD_STACK];
```

5 タスクを関数の形で記述する

タスクは、UArg 型の引数を 2 つ持つ、戻り値が Void 型の関数で記述します。なお、2 つの引数はタスク生成時のパラメータとして指定することができます（p.42 ～ 43 のソースコード empty.c 03 参照）。

```
例 : Void task_GLCD(UArg arg0, UArg arg1)
    {
    // 実際のコード
    }
```

6 Task_construct 関数でタスクを作成する

Task_construct 関数にタスクの構造体、タスクの関数、タスクのパラメータ（優先度等）を渡すことで、タスクが作成されます。

スタックサイズ、スタック、優先度といったパラメータは、Task_Params 型構造体で宣言した変数のメンバ変数で指定することになります（詳細は p.47 の型・構造体 03 をご覧ください）。なお、Task_Params 型変数はタスク作成時にパラメータを渡すだけなので、タスクごとに用意する必要はありません。

以下は、Task_construct 関数でのタスクの作成例です。

```
Task_Params taskParams;

Task_Params_init(&taskParams);
taskParams.stackSize = TASK_GLCD_STACK;
taskParams.stack = &taskGLCDStack;
taskParams.priority = TASK_GLCD_PRIO;
Task_construct(&taskGLCDStruct, (Task_FuncPtr)task_GLCD, &taskParams, NULL);
```

GLCD の使用

コーディング main.c 07

以下は、main.c 06のソースコードを元にしたコーディング例です。■はmain.c 06から新たに追加した部分です。

main.c

```
/* XDCtools ヘッダファイル */
#include <xdc/std.h>
#include <xdc/runtime/System.h>

/* BIOS ヘッダファイル */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Clock.h>

/* ドライバヘッダファイル */
#include <ti/drivers/GPIO.h>

/* ボードヘッダファイル */
#include "Board.h"

/* GLCD ヘッダファイル */
#include "GLCD.h"

/* ユーザー定義マクロ */
#define CONSOLE(...) do { System_printf(__VA_ARGS__); System_flush(); } while(0)
#define SLEEP(X) Task_sleep((X)*1000/Clock_tickPeriod)

/*
 * ユーザータスクの優先度
 */
#define TASK_LED_PRIO 1
#define TASK_GLCD_PRIO 9

/*
 * ユーザースタックのサイズ
 */
#define TASK_LED_STACK 512
#define TASK_GLCD_STACK 512

/*
 * ユーザータスクの構造体
 */
Task_Struct taskLEDStruct;
Task_Struct taskGLCDStruct;

/*
 * ユーザータスクのスタック
 */
Char taskLEDStack[TASK_LED_STACK];
Char taskGLCDStack[TASK_GLCD_STACK];
```

```
46
47  /*
48   * ユーザータスク
49   */
50
51  /*
52   * LED 点滅タスク
53   */
54  Void task_LED(UArg arg0, UArg arg1)
55  {
56      CONSOLE("LED 点滅タスクの開始 \n");
57
58      // LED1 を点灯
59      GPIO_write(Board_LED0, Board_LED_ON);
60
61      while (1)
62      {
63          // LED を 0.5 秒おきにトグル
64          SLEEP(500);
65          GPIO_toggle(Board_LED0);
66      }
67  }
68
69  /*
70   * GLCD 表示タスク
71   */
72  Void task_GLCD(UArg arg0, UArg arg1)
73  {
74      // 表示する文字列を配列形式で宣言
75      char str_1[] = "ARM Challenger";
76      char str_2[] = { 'A', 'D', 'W', 'I', 'N', '\0' };
77
78      CONSOLE("GLCD 表\示タスクの開始 \n");
79
80      // 各文字列を GLCD に表示
81      GLCD_str(0, 0, "HELLO WORLD");
82      GLCD_str(2, 0, str_1);
83      GLCD_str(4, 0, str_2);
84
85      while (1)
86      {
87          // 時間待ち
88          SLEEP(1000);
89      }
90  }
91
92  /*
93   * メイン
94   */
95
```

Shift_JIS の「ダメ文字」回避

GLCD の使用

コーディング main.c 07

```
 96 int main(void)
 97 {
 98     Task_Params taskParams;
 99
100     // ボードの初期設定
101     CONSOLE(" ボードの初期設定 \n");
102     Board_initGeneral();
103     Board_initGPIO();
104
105     // GLCD の初期化
106     GLCD_init();
107
108     // ユーザータスクの作成
109     CONSOLE(" ユーザータスクの作成 \n");
110     Task_Params_init(&taskParams);
111     taskParams.stackSize = TASK_LED_STACK;
112     taskParams.stack = &taskLEDStack;
113     taskParams.priority = TASK_LED_PRIO;
114     Task_construct(&taskLEDStruct, (Task_FuncPtr) task_LED, &taskParams, NULL);
115
116     Task_Params_init(&taskParams);
117     taskParams.stackSize = TASK_GLCD_STACK;
118     taskParams.stack = &taskGLCDStack;
119     taskParams.priority = TASK_GLCD_PRIO;
120     Task_construct(&taskGLCDStruct, (Task_FuncPtr) task_GLCD, &taskParams, NULL);
121
122     // OS の起動
123     CONSOLE("OS の起動 \n");
124     BIOS_start();
125
126     return (0);
127 }
128
```

Shift_JIS の「ダメ文字」回避

コード中に、Shift_JIS の「ダメ文字」があると、文字化けを起こしてしまいます。
例えば、「表示」は「侮ヲ」となります。
以下のように「ダメ文字」である「表」の後にエスケープ記号「 \ (¥)」をつけると回避できます。

```
CONSOLE("GLCD 表\示タスクの開始 \n");
```