microSD カードの使用

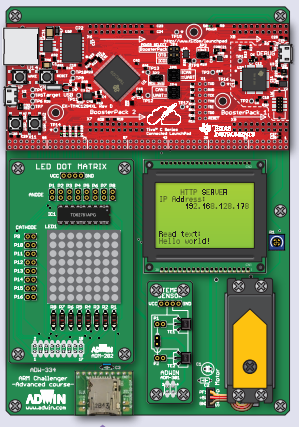
学習内容

本 STEP では microSD カード内のデータの読み込み方法を学習します。

課題 09-1

1. パソコンで microSD カード内に「read.txt」ファイルを保存しておきます。「read.txt」の内容は任意の半角文字列（例は”Hello world!”）とします。

2. マイコンから microSD カードの「read.txt」を読み込むことに成功したら、GLCD に「read.txt」の内容を保存します。ファイルの読み込みに失敗した場合は、エラーメッセージを表示します。

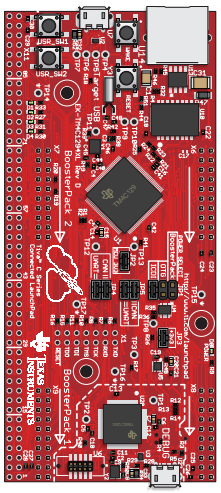


付属の microSD カードに「read.txt」を保存し、マイコンボードのカードスロットにセットする

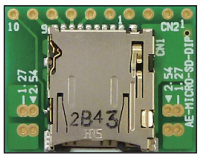
microSD カードの使用

配線 09-1

今回は microSD モジュールを追加で使います。マイコンボードと microSD モジュール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、microSD モジュールの SW-A と SW-B は、microSD カード挿入時にショートします。

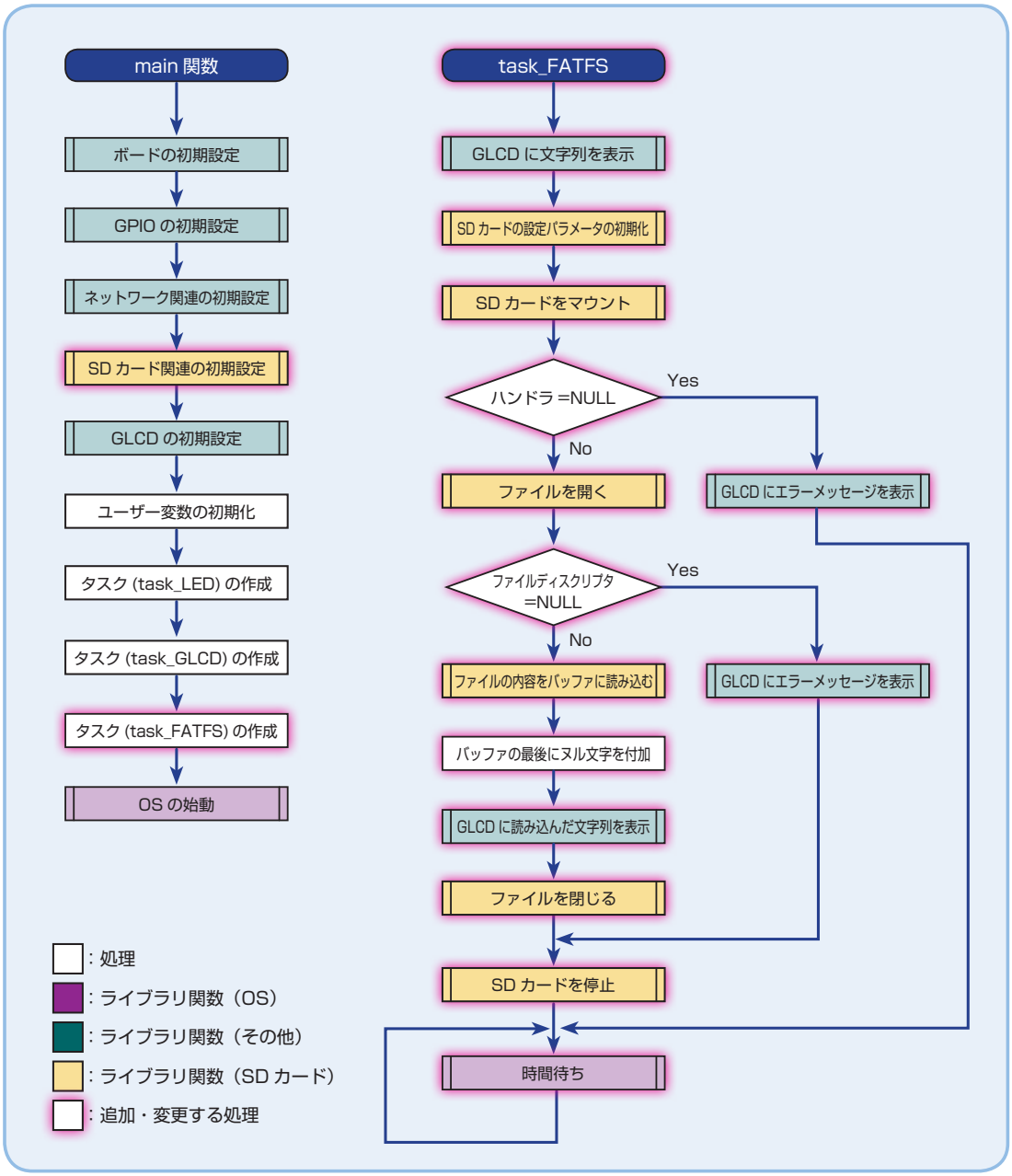


マイコンボード		microSD モジュール	
		CN2-1	NC
PC1	⟷	CN2-2	CS
PD1	⟷	CN2-3	DI
+5V	⟷	CN2-4	V_{DD}
PD3	⟷	CN2-5	SCLK
GND	⟷	CN2-6	V_{SS}
PD1	⟷	CN2-7	DO
		CN2-8	NC
PP2	⟷	CN2-9	SW-B
	⟷	CN2-10	SW-A



フローチャート 09-1

以下は、課題 09-1 を実現するためのフローチャート例です。その他のタスクは「フローチャート 08-2」と同じです。



microSD カードの使用

初期設定 09-1

microSD を使うには、OS の FatFS モジュールと xdc.runtime.Timestamp モジュールを有効化する必要があります。なお、xdc.runtime.Timestamp モジュールの有効化は、GUI での設定には対応しておらず、cfg ファイルに直接書き込む必要があります。

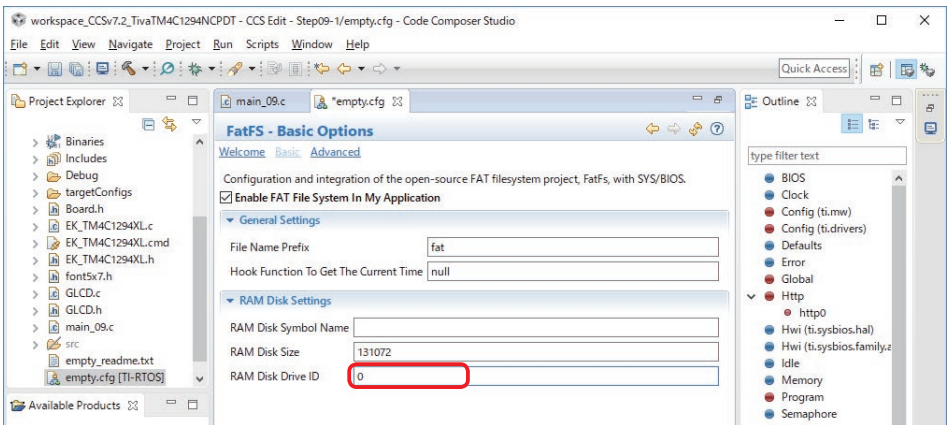
FatFS モジュールの有効化と設定

❶ Project Explorer 上で「empty.cfg」を右クリックし、「Open With」→「XGCONF」を選択します。

❷「System Overview」を開き、「Middleware & Drivers」内の「FatFS」を右クリックして「Use FatFS」を選択します。

❸「FatFS welcome」が開いたら、ページ上の「Basic」を開きます。

❹「RAM Disk Settings」内の「RAM Disk Drive ID」を「0」に設定します。

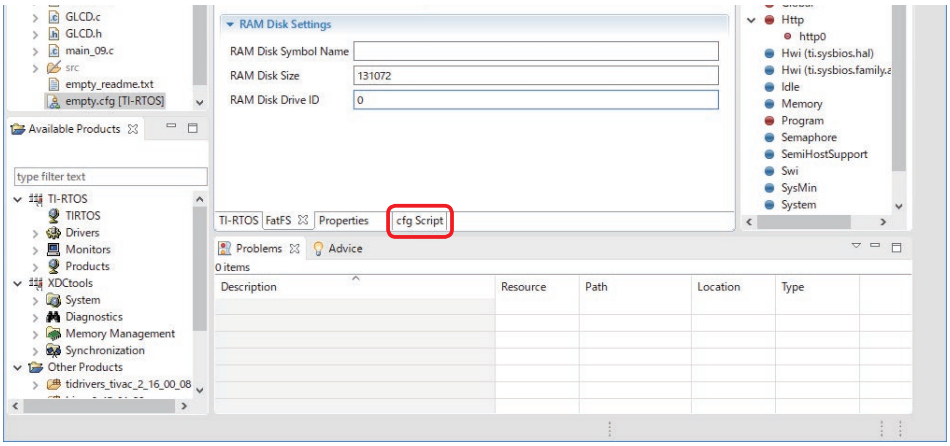


以上の操作によって、「empty.cfg」ファイルには以下の行が追加されることになります。

```
var FatFS = xdc.useModule('ti.mw.fatfs.FatFS');
FatFS.ramdiskDriveId = 0;
```

xdc.runtime.Timestamp モジュールの有効化

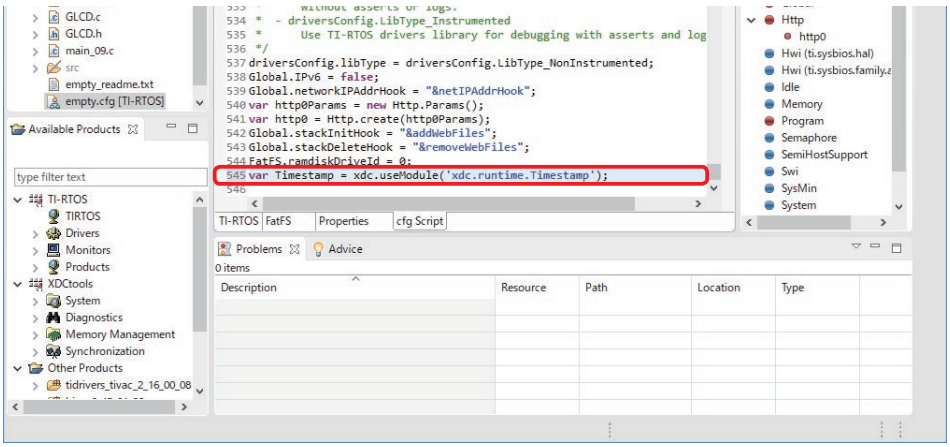
❺ 画面下の「cfg Script」タブをクリックし、cfg ファイルのテキスト編集モードに入ります。



❻ ファイルの任意の場所（特に事情が無ければ一番下）に

```
var Timestamp = xdc.useModule('xdc.runtime.Timestamp');
```

という行を加えます。



以上の設定が完了したら、「Save」アイコン をクリックして cfg ファイルを保存しましょう。

microSD カードの使用

インクルードファイル 09-1

ドライバヘッダファイル　ti/driver/SDSPI.h

SD カードを扱うための定数や型・関数が定義されています。本課題では、以下の型・関数を使います。

- SDSPI_Handle
- SDSPI_Params
- SDSPI_Params_init()
- SDSPI_open()
- SDSPI_close()

C 言語ヘッダファイル　stdio.h

C 言語の入出力標準ライブラリです。

型 09-1

ドライバ関係

本課題で新しく使うドライバ関係の型は、いずれも「ti/driver/SDSPI.h」で定義されています。

「ti/driver/SDSPI.h」で定義されている型の詳細については、「Help Contents」の TI-RTOS for TivaC * → Documentation Links → TI-RTOS Driver Runtime APIs (doxygen) → SDSPI.h をご覧ください。

SDSPI_Handle

マウントされた SD カードを操作するためのハンドラです。中身を直接操作することはありません。

SDSPI_Params

SD カードの操作に関するパラメータを格納する構造体です。本コースでは中身を直接操作することはありませんが、以下のメンバ変数を持ちます。

- bitRate ：SD カードを操作する SPI のビットレート
- custom ：ドライバによるカスタム引数

C 言語標準

FILE

stdio.h で定義されている C 言語標準のファイルディスクリプタです。中身を直接操作することはありません。

microSD カードの使用

ライブラリ関数 09-1

ボード関係

ボードの SD カード関連の初期設定　Board_initSDSPI()

SD カードを扱うための初期設定を行います。具体的な処理内容は、「EX_TM4C1294XL.c」中の「EK_TM4C1294XL_initSDSPI」に書かれています。

ドライバ関係

本課題で新しく使うドライバ関係の関数は、いずれも「ti/driver/SDSPI.h」で定義されています。

「ti/driver/SDSPI.h」で定義されている関数の詳細については、CCS の「Help」→「Help Contents」より「TI-RTOS for TivaC *」→「Documentation Links」→「Drivers Documentation」→「TI-RTOS DriverRuntime APIs (doxygen)」→「SDSPI.h」をご覧ください。

SD カードを扱うパラメータの初期化　void SDSPI_Params_init(SDSPI_Params *params)

SD カードを扱うパラメータの構造体 *params に初期値を代入します。以下は SDSPI_Params 型変数 sdspiParams の初期化例です。

```
SDSPI_Params_init(&sdspiParams);
```

引数の設定例

- `&sdspiParams`　：SDSPI_Params 型変数のポインタを設定する。

SD カードをマウントする
SDSPI_Handle SDSPI_open(unsigned int index,unsigned char drv, SDSPI_Params *params)

FAT ファイルシステムの SD カードをマウントし、ハンドラを返します。失敗した場合は NULL(=0) を返します。以下は使用例です。

```
sdspiHandle = SDSPI_open(Board_SDSPI0, 0, &sdspiParams);
```

引数の設定例

- `Board_SDSPI0`：マイコンボードのどの端子を使うかを指定する。本コースでは「Board_SDSPI0」の端子に microSD カードモジュールが接続されており、「Board_SDSPI0」を指定。
- `0`　：SD カードの FAT ファイルシステムのドライバ番号を指定する。ここでは 0 を指定。
- `&sdspiParams`：設定を行った SDSPI_Params 型変数のポインタを指定する。

SD カードをアンマウントする　void SDSPI_close(SDSPI_Handle handle)

マウントされている SD カードをアンマウントします。以下は使用例です。

```
SDSPI_close(sdspiHandle);
```

引数の設定例

- sdspiHandle　：SDSPI_open 関数が返したハンドラを指定する。

C 言語標準

本課題で新しく使う C 言語標準の関数は、いずれも「stdio.h」で定義されています。

ファイルを開く　FILE *fopen(const char *_fname, const char *_mode)

ファイルを開き、ファイルディスクリプタを返します。以下は、SD カード内のファイル「read.txt」を読み込み専用で開き、ファイルディスクリプタを fd に代入する例です。失敗した場合は NULL（=0）を返します。

```
fd = fopen("fat:0:read.txt", "r");
```

引数の設定例

- "fat:0:read.txt"　：読み込むファイルを指定する。
  今の場合、ファイル名の前に「fat:0:」と書く必要がある。
- "r"　：ファイルを開くモードを指定する。
  読み込み専用の時は「"r"」、書き込み専用の時は「"w"」、追記専用の時は「"a"」を指定する。

microSD カードの使用

ライブラリ関数 09-1

ファイルを読む　size_t fread(void *_ptr, size_t _size, size_t _count, FILE *_fp)

ファイルディスクリプタ *_fp のファイルの内容を、_size バイトずつ _count 個読み込み、*_ptr で指定された配列に格納します。戻り値は、読み込んだデータ個数です。以下は、ファイルディスクリプタ fd のファイルの中身を readBuffer に読み込み、読み込んだバイト数を bytesRead に代入する例です。

```
bytesRead = fread(readBuffer, 1, sizeof(readBuffer)-1, fd);
```

引数の設定例

- readBuffer：読み込んだデータを保存するバッファ（配列）を指定する。
- 1　　：読み込むデータの大きさを指定する。

以下のように使うと、ファイルの文字列を読み込んで最後にヌル文字をつけることができます（1 バイトずつ読むように fread を指定しているので、fread の戻り値は読み込んだバイト数に一致します。また、最後にヌル文字を別途つけるので読み取りの最大サイズはバッファサイズより 1 小さくしています）。

```
bytesRead = fread(readBuffer, 1, sizeof(readBuffer)-1, fd);
readBuffer[bytesRead] = '¥0';
```

ファイルを閉じる　int fclose(FILE *_fp)

ファイルディスクリプタ *_fp のファイルを閉じます。以下は使用例です。

```
fclose(fd);
```

引数の設定例

- fd　　：　閉じたいファイルのファイルディスクリプタ（fopen が返したもの）を指定する。

> **8.3 形式**
> TI-RTOS の FatFS モジュールでは、ファイル名は拡張子以外（ドットの前）が 8 文字以内、拡張子が 3 文字の 8.3 形式にする必要があります。また、大文字と小文字は区別されません（小文字も大文字として扱われます）。
> 8.3 形式でないファイルは、マイコンから認識できない場合があるのでご注意ください。

コーディング main.c 09-1

以下は、main.c 08-2 のソースコードを元にしたコーディング例です。■は main.c 08-2 から追加した部分です。

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>

/* C 言語ヘッダファイル */
#include <stdio.h>

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

/* NDK ヘッダファイル */
#include <ti/ndk/inc/netmain.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
#define TASK_FATFS_PRIO 10

/*
 * ユーザースタックのサイズ
 */
#define TASK_LED_STACK 512
#define TASK_GLCD_STACK 512
#define TASK_FATFS_STACK 1024

/*
```

microSD カードの使用

コーディング 09-1

```
45  * ユーザータスクの構造体
46  */
47 Task_Struct taskLEDStruct;
48 Task_Struct taskGLCDStruct;
49 Task_Struct taskFATFSStruct;
50
51 /*
52  * ユーザータスクのスタック
53  */
54 Char taskLEDStack[TASK_LED_STACK];
55 Char taskGLCDStack[TASK_GLCD_STACK];
56 Char taskFATFSStack[TASK_FATFS_STACK];
```

この間の行に変更はありません

```
114 /*
115  * FATFS タスク
116  */
117 Void task_FATFS(UArg arg0, UArg arg1)
118 {
119     SDSPI_Handle sdspiHandle; // SD カードマウントのハンドラ
120     SDSPI_Params sdspiParams; // SD カードマウントのパラメータ
121
122     FILE *fd; // ファイルディスクリプタ
123
124     char readBuffer[128]; // ファイル読み込みバッファ
125     unsigned int bytesRead; // ファイル読み込みサイズ
126
127     CONSOLE("FATFS タスクの開始 \n");
128
129     // GLCD に文字列を表示
130     GLCD_str(6, 0, "Read text:");
131
132     // SD カードの設定パラメータの初期化
133     SDSPI_Params_init(&sdspiParams);
134
135     // SD カードをマウント
136     sdspiHandle = SDSPI_open(Board_SDSPI0, 0, &sdspiParams);
137
138     // エラーチェック
139     if (sdspiHandle == NULL)
140     {
141         CONSOLE(" マウントに失敗しました！ \n");
142         // GLCD にエラーメッセージを表示
143         GLCD_str(7, 0, "Mount error");
144     }
145     else
146     {
147         // ファイルを開く
148         fd = fopen("fat:0:read.txt", "r");
```

```
149
150         // エラーチェック
151         if (fd == NULL)
152         {
153             CONSOLE(" ファイルを開けませんでした！ \n");
154             // GLCD にエラーメッセージを表示
155             GLCD_str(7, 0, "File open error");
156         }
157         else
158         {
159             // ファイルから文字列を読み込み
160             bytesRead = fread(readBuffer, 1, sizeof(readBuffer) -1, fd);
161             readBuffer[bytesRead] = '\0';
162
163             // GLCD に読み込んだ文字列を表示
164             GLCD_str(7, 0, readBuffer);
165
166             // ファイルを閉じる
167             fclose(fd);
168         }
169
170         // SD カードを停止
171         SDSPI_close(sdspiHandle);
172     }
173
174     while (1)
175     {
176         // 時間待ち
177         SLEEP(1000);
178     }
179 }
180
181 /*
182  * フック関数
183  */
```

この間の行に変更はありません

```
216 /*
217  * メイン
218  */
219
220 int main(void)
221 {
222     Task_Params taskParams;
223
224     // ボードの初期設定
225     CONSOLE(" ボードの初期設定 \n");
226     Board_initGeneral();
227     Board_initGPIO();
```

microSD カードの使用

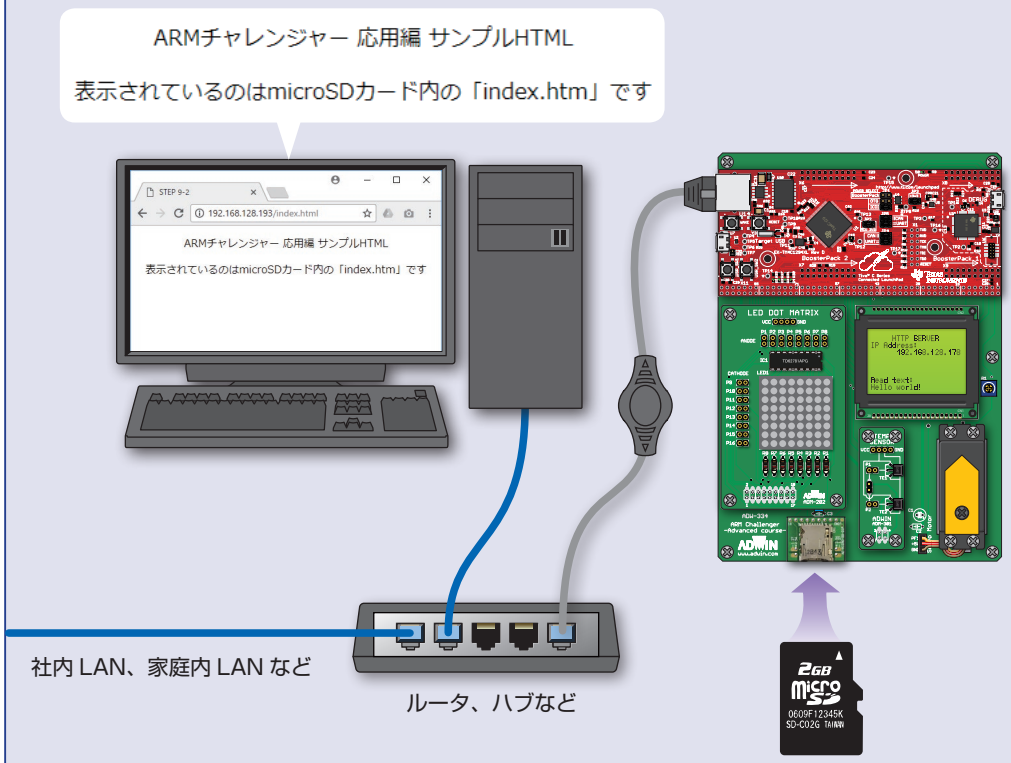
コーディング 09-1

```
228     Board_initEMAC();
229     Board_initSDSPI();
230
231     // GLCD の初期化
232     GLCD_init();
233
234     // ユーザー変数の初期値の設定
235     IP = 0;
236
237     // ユーザータスクの作成
238     CONSOLE(" ユーザータスクの作成 \n");
239     Task_Params_init(&taskParams);
240     taskParams.stackSize = TASK_LED_STACK;
241     taskParams.stack = &taskLEDStack;
242     taskParams.priority = TASK_LED_PRIO;
243     Task_construct(&taskLEDStruct, (Task_FuncPtr) task_LED, &taskParams, NULL);
244
245     Task_Params_init(&taskParams);
246     taskParams.stackSize = TASK_GLCD_STACK;
247     taskParams.stack = &taskGLCDStack;
248     taskParams.priority = TASK_GLCD_PRIO;
249     Task_construct(&taskGLCDStruct, (Task_FuncPtr) task_GLCD, &taskParams, NULL);
250
251     Task_Params_init(&taskParams);
252     taskParams.stackSize = TASK_FATFS_STACK;
253     taskParams.stack = &taskFATFSStack;
254     taskParams.priority = TASK_FATFS_PRIO;
255     Task_construct(&taskFATFSStruct, (Task_FuncPtr) task_FATFS, &taskParams, NULL);
256
257     // OS の起動
258     CONSOLE("OS の起動 \n");
259     BIOS_start();
260
261     return (0);
262 }
263
```

microSD カードの使用

課題 09-2

1. パソコンで microSD カード内に「index.htm」ファイルを保存しておきます。「index.htm」の内容は任意の HTML ソースコードとします。
2. パソコンのブラウザからマイコンにリクエストを送信します。
   例）http://192.168.128.178/
3. SD カード内の「index.htm」の内容を返信し、ブラウザに表示させましょう。

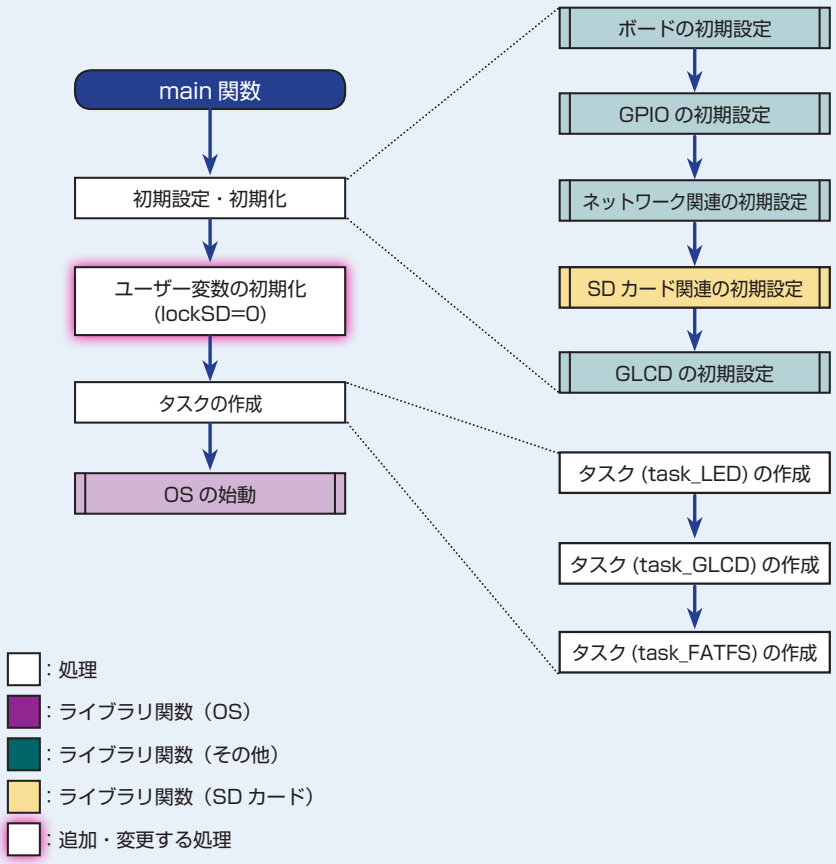


> ⚠ microSD カードに保存するファイルの拡張子は、8.3 形式に従って「.htm」とする必要がありますが、Web サーバ上のファイル名は「index.html」のように拡張子を「.html」とするものとします。

microSD カードの使用

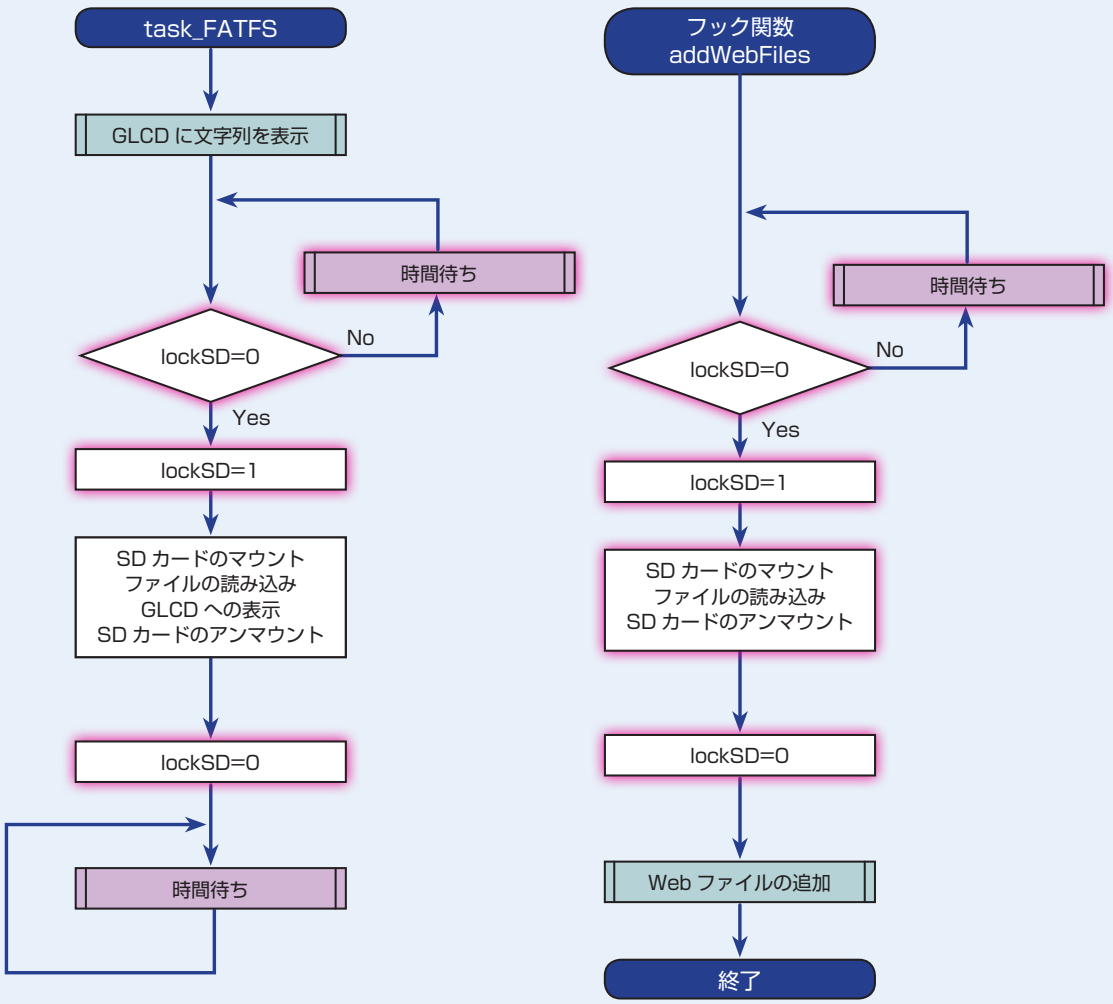
フローチャート 09-2

以下は、課題 09-1 を実現するためのフローチャート例です。ここでは、SD カードの排他制御用に変数「lockSD」を用意しています。その他のタスクは「フローチャート 08-2」と同じです。



この STEP 以降では、フローチャートが煩雑になるのを避けるため、変更のない似通った処理はまとめて描くことにします。

microSD カードの使用



microSD カードの使用

コーディング main.c 09-2

以下は、main.c 09-1 のソースコードを元にしたコーディング例です。■は main.c 09-1 から追加・変更した部分です。

main.c

これ以前の行に変更はありません

```
58 /*
59  * ユーザー変数
60  */
61 uint32_t IP; // IP アドレス格納用変数
62 char lockSD; // SD カード排他制御用変数
63
64 /*
65  * ユーザータスク
66  */
```

この間の行に変更はありません

```
115 /*
116  * FATFS タスク
117  */
118 Void task_FATFS(UArg arg0, UArg arg1)
119 {
120     SDSPI_Handle sdspiHandle; // SD カードマウントのハンドラ
121     SDSPI_Params sdspiParams; // SD カードマウントのパラメータ
122
123     FILE *fd; // ファイルディスクリプタ
124
125     char readBuffer[128]; // ファイル読み込みバッファ
126     unsigned int bytesRead; // ファイル読み込みサイズ
127
128     CONSOLE("FATFS タスクの開始 \n");
129
130     // GLCD に文字列を表示
131     GLCD_str(6, 0, "Read text:");
132
133     // 他のタスクが SD カードを使っている場合は使い終わるまで待つ
134     while (lockSD != 0) SLEEP(100);
135
136     // SD カードの使用を開始
137     lockSD = 1;
138
139     // SD カードの設定パラメータの初期化
140     SDSPI_Params_init(&sdspiParams);
141
142     // SD カードをマウント
143     sdspiHandle = SDSPI_open(Board_SDSPI0, 0, &sdspiParams);
```

```
144
145     // エラーチェック
146     if (sdspiHandle == NULL)
147     {
148         CONSOLE(" マウントに失敗しました！ \n");
149         // GLCD にエラーメッセージを表示
150         GLCD_str(7, 0, "Mount error");
151     }
152     else
153     {
154         // ファイルを開く
155         fd = fopen("fat:0:read.txt", "r");
156
157         // エラーチェック
158         if (fd == NULL)
159         {
160             CONSOLE(" ファイルを開けませんでした！ \n");
161             // GLCD にエラーメッセージを表示
162             GLCD_str(7, 0, "File open error");
163         }
164         else
165         {
166             // ファイルから文字列を読み込み
167             bytesRead = fread(readBuffer, 1, sizeof(readBuffer) -1, fd);
168             readBuffer[bytesRead] = '\0';
169
170             // GLCD に読み込んだ文字列を表示
171             GLCD_str(7, 0, readBuffer);
172
173             // ファイルを閉じる
174             fclose(fd);
175         }
176
177         // SD カードを停止
178         SDSPI_close(sdspiHandle);
179     }
180
181     // SD カードの使用を終了
182     lockSD = 0;
183
184     while (1)
185     {
186         // 時間待ち
187         SLEEP(1000);
188     }
189 }
190
191 /*
192  * フック関数
193  */
```

microSD カードの使用

コーディング main.c 09-2

```
194
195 Void netIPAddrHook(IPN IPAddr, uint IfIdx, uint fAdd)
196 {
197     /* IP アドレスの取得 */
198     IP = IPAddr;
199     CONSOLE("IP アドレス : %x\n", IP);
200 }
201
202 Void addWebFiles()
203 {
204     SDSPI_Handle sdspiHandle; // SD カードマウントのハンドラ
205     SDSPI_Params sdspiParams; // SD カードマウントのパラメータ
206
207     FILE *fd; // ファイルディスクリプタ
208
209     static char page[2048]; // ページデータの読み込みバッファ
210     unsigned int bytesRead; // ファイル読み込みサイズ
211
212     // 他のタスクが SD カードを使っている場合は使い終わるまで待つ
213     while (lockSD != 0) SLEEP(100);
214
215     // SD カードの使用を開始
216     lockSD = 1;
217
218     // SD カードの設定パラメータの初期化
219     SDSPI_Params_init(&sdspiParams);
220
221     // SD カードをマウント
222     sdspiHandle = SDSPI_open(Board_SDSPI0, 0, &sdspiParams);
223
224     // エラーチェック
225     if (sdspiHandle == NULL)
226     {
227         CONSOLE(" マウントに失敗しました！ \n");
228         // GLCD にエラーメッセージを表示
229         GLCD_str(7, 0, "Mount error");
230     }
231     else
232     {
233         // ファイルを開く
234         fd = fopen("fat:0:index.htm", "r");
235         // エラーチェック
236         if (fd == NULL)
237         {
238             CONSOLE(" ファイルを開けませんでした！ \n");
239             // GLCD にエラーメッセージを表示
240             GLCD_str(7, 0, "File open error");
241         }
242         else
243         {
```

```
244             // ファイルから文字列を読み込み
245             bytesRead = fread(page, 1, sizeof(page) - 1, fd);
246             page[bytesRead] = '\0';
247             CONSOLE(" 読み込んだバイト数 : %d/%d\n", bytesRead, sizeof(page) - 1);
248 
249             // ファイルを閉じる
250             fclose(fd);
251         }
252         // SD カードを停止
253         SDSPI_close(sdspiHandle);
254     }
255     // SD カードの使用を終了
256     lockSD = 0;
257 
258     /* Web ファイルの追加 */
259     efs_createfile("index.html", strlen(page), (UINT8 *) page);
260 }
261 
262 Void removeWebFiles()
263 {
264     /* Web ファイルの削除 */
265     efs_destroyfile("index.html");
266 }
267 
268 /*
269  * メイン
270  */
271 
272 int main(void)
273 {
274     Task_Params taskParams;
275 
276     // ボードの初期設定
277     CONSOLE(" ボードの初期設定 \n");
278     Board_initGeneral();
279     Board_initGPIO();
280     Board_initEMAC();
281     Board_initSDSPI();
282 
283     // GLCD の初期化
284     GLCD_init();
285 
286     // ユーザー変数の初期値の設定
287     IP = 0;
288     lockSD = 0;
```

これ以降の行に変更はありません