リスト5-3-7　task5.cの追記部

```
38    // キューハンドル定義
39    QueueHandle_t  LOGQueue;
40    QueueHandle_t  LCDQueue;
41    /* Application Data*/
42    APP_DATA appData;
43    #define SDCARD_MOUNT_NAME     "/mnt/mydrive"
44    #define SDCARD_DEV_NAME       "/dev/mmcblk1"
45
46    /** グローバル変数定義 ***/
47    uint8_t WriteBufM[] = "M,xxxx.x,xx.xx,xx.xx¥r¥n";
48    uint8_t WriteBufV[] = "V,x.xx,x.xx¥r¥n";
49    uint8_t LogFlag;                          // ログ中フラグ
50    /** キュー受信用バッファ定義 ***/
51    union {
52        uint8_t Buf[20];
53        struct {
54            uint8_t kind;
55            float data1;
56            float data2;
57            float data3;
58        }mes;
59    }LogData;
60
61    /** LCDメッセージ用データ定義 ****/
62    union {
63        uint8_t Buf[20];
64        struct{
65            uint8_t kind;
66            uint8_t msg[13];    // メッセージデータ
67        }mes;
68    }LogMsg;
69    /** LCDメッセージ漢字データ定義 ***/
70    uint8_t StartLog[]  = {0x81,0x40,0x83,0x8D,0x83,0x4D,0x83,0x93,0x83,0x4F,0x8A,0x4A,0x8E,0x6E,0x81,0x40,0x00,0x00};     //ロギング開始
71    uint8_t EndLog[]    = {0x81,0x40,0x83,0x8D,0x83,0x4D,0x83,0x93,0x83,0x4F,0x8F,0x49,0x97,0xB9,0x81,0x40,0x00,0x00};     //ロギング終了
72    uint8_t WriteErr[]  = {0x81,0x40,0x8F,0x91,0x8D,0x9E,0x82,0xDD,0x83,0x43,0x83,0x89,0x81,0x5B,0x81,0x40,0x00,0x00};     //書込みエラー
73    /** 関数プロト ***/
74    void ftostring(int seisu, int shousu, float data, uint8_t *buffer);
```

ハンドル定義

マウント情報

ログキュー受信バッファ定義

表示キュー送信バッファ定義

表示用漢字メッセージ定義

task5.hの追記部

```
63    typedef enum
64    {
65        /* Application's state machine's initial state. */
66        TASK5_STATE_INIT=0,
67        TASK5_STATE_SERVICE_TASKS,
68        /* TODO: Define states used by the application state machine. */
69
70    } TASK5_STATES;
71    typedef enum
72    {
73        APP_WAIT_SWITCH_PRESS = 0,      // スイッチオン待ち
74        APP_MOUNT_DISK,                 // SDマウント
75        APP_WAIT_QUEUE,                 // キュー待ち
76        APP_OPEN_FILE,                  // ファイルオープン
77        APP_WRITE_FILE,                 // ファイル書き込み
78        APP_CLOSE_FILE,                 // ファイルクローズ
79        APP_UNMOUNT,                    // ドライブアンマウント
80        APP_ERROR                       // ファイルアクセスエラー
81    } APP_STATES;
```

アプリ用ステートの定義

リスト5-3-8　task5.cの追記部

```
109   void TASK5_Initialize ( void )
110   {
111       /* Place the App state machine in its initial stat
112       task5Data.state = TASK5_STATE_INIT;
113
114       appData.APstate = APP_WAIT_SWITCH_PRESS;
115
116       /* TODO: Initialize your application's state machi
117        * parameters.
118        */
119   }
```

ステート初期化

```
150   case TASK5_STATE_SERVICE_TASKS:
151   {
152
153       switch ( appData.APstate )
154       {
155           /** S1スイッチオン待ち **/
156           case APP_WAIT_SWITCH_PRESS:
157               if(S1_Get() == 0){                          // S1オン
158                   LogFlag = 1;                            // ログ中フラグオン
159                   LogMsg.mes.kind = 'S';                  // 種類
160                   strcpy((char *)LogMsg.mes.msg, (char *)StartLog);
161                   /** キューへ送信 **/
162                   xQueueSend(LCDQueue, LogMsg.Buf, 0);
163                   appData.APstate = APP_MOUNT_DISK;
164               }
165               break;
166           /** SDカードのマウント **/
167           case APP_MOUNT_DISK:
168               if(SYS_FS_Mount(SDCARD_DEV_NAME, SDCARD_MOUNT_NAME, FAT, 0, NULL) != 0){
169                   appData.APstate = APP_MOUNT_DISK;       // 失敗ならマウント繰り返し
170               }
171               else{
172                   appData.APstate = APP_OPEN_FILE;        // 成功ならファイルオープンへ
173               }
174               break;
175           /* ファイルオープン実行 */
176           case APP_OPEN_FILE:
177               appData.fileHandle = SYS_FS_FileOpen("LogFile.txt", (SYS_FS_FILE_OPEN_APPEND));
178               if(appData.fileHandle == SYS_FS_HANDLE_INVALID)   // ハンドル無効の場合
179                   appData.APstate = APP_ERROR;            // エラーへ
180               else{
181                   appData.APstate = APP_WAIT_QUEUE;       // 成功なら書き込みへ
182               }
183               break;
184           /** キューからデータ取り出し **/
185           case APP_WAIT_QUEUE:                            // キュー待ち
186               if(S2_Get() == 0)                           // S2がオンの時
```

S1オンで開始

開始メッセージ
表示キューへ送信

SDマウント

ファイルオープン

キューから取り出し

```
194           /** 書き込み実行 ***/
195           case APP_WRITE_FILE:
196               if(LogData.mes.kind == 'M'){                    // センサデータの場合
197                   Red_Set();
198                   ftostring(4, 1, LogData.mes.data1, WriteBufM+2);    // データ文字変換
199                   ftostring(2, 2, LogData.mes.data2, WriteBufM+9);
200                   ftostring(2, 2, LogData.mes.data3, WriteBufM+15);
201
202                   if(SYS_FS_FileWrite(appData.fileHandle, (const void *)WriteBufM, 22) == -1)
203                   {   /* エラーの場合 */
204                       SYS_FS_FileClose(appData.fileHandle);       // ファイルクローズ
205                       appData.APstate = APP_ERROR;                // エラーへ
206                   }
207                   else{
208                      appData.APstate = APP_WAIT_QUEUE;            // キュー待ちへ戻る
209                      Red_Clear();
210                   }
211               }
212               else if(LogData.mes.kind == 'V'){                   // 電圧データの場合
213                   Green_Set();
214                   ftostring(1, 2, LogData.mes.data1, WriteBufV+2);    // データ文字変換
215                   ftostring(1, 2, LogData.mes.data2, WriteBufV+7);
216                   if(SYS_FS_FileWrite(appData.fileHandle, (const void *)WriteBufV, 13) == -1)
217                   {   /* エラーの場合 */
218                       SYS_FS_FileClose(appData.fileHandle);       // ファイルクローズ
219                       appData.APstate = APP_ERROR;                // エラーへ
220                   }
221                   else{
222                       appData.APstate = APP_WAIT_QUEUE;           // キュー待ちへ戻る
223                       Green_Clear();
224                   }
225               }
226               break;
227           /** ファイルクローズ処理 ***/
228           case APP_CLOSE_FILE:
229               SYS_FS_FileClose(appData.fileHandle);               // ファイルクローズ
230               appData.APstate = APP_UNMOUNT;                      // アンマウントへ
231               LogFlag = 0;                                        // ログ中フラグオフ
232               break;
233           /*** ドライブアンマウント実行 ***/
234           case APP_UNMOUNT:
235               if(SYS_FS_Unmount(SDCARD_MOUNT_NAME) != SYS_FS_RES_SUCCESS)
236                   appData.APstate = APP_UNMOUNT;                  // 失敗なら繰り返し
237               else{
238                   LogMsg.mes.kind = 'S';
239                   strcpy((char *)LogMsg.mes.msg, (char *)EndLog);
240                   /** キューへ送信 **/
241                   xQueueSend(LCDQueue, LogMsg.Buf, 0);
242                   appData.APstate = APP_WAIT_SWITCH_PRESS;        // 次の開始待ちへ
243               }
244               break;
245           case APP_ERROR:
246               /** エラー処理なし **/
```

種別ごとに編集

書き込み

終了メッセージ表
示キューへ送信