V850E2/ML4

マルチメディアカード SPI モード対応デバイスドライバ : 導入ガイド

R01AN1026JJ0101
Rev.1.01
2012.07.23

要旨

本アプリケーションノートでは、V850E2/ML4 用マルチメディアカードデバイスドライバと、その使用方法を説明します。

動作確認デバイス

V850E2/ML4 (μ PD70F4022)

目次

1. 概要

1.1 目的

V850E2/ML4 によりマルチメディアカード（以下、MMC と略す。）を SPI モードで制御する例を示すことを目的としています。

本アプリケーションノートでは、アプリケーションを作成する例を提供します。

1.2 機能概要

本デバイスドライバ（以下、MMC ドライバと略す。）は、MMC との通信を V850E2/ML4 で実現するためのソフトウェアです。

本ソフトウェアは、FAT ファイルシステムを実装しています。

本ソフトウェアは、V850E2/ML4 に内蔵されている通信機能のうち、クロック同期式シリアル・インタフェース（以下、CSIH と略す。）を使い、SPI モードでの MMC へのアクセスを実現しています。

MMC ドライバの仕様

- MMC 規格 Ver.3.x
- MMC の SPI モードで動作します。
- 1 セクタ=512Byte とするブロック型デバイスドライバです。
  READ_MULTIPLE_BLOCK コマンドと WRITE_MULTIPLE_BLOCK コマンドを使用しています。
  上記 2 つの MULTIPLE_BLOCK コマンド非対応カードの場合は、READ_SINGLE_BLOCK コマンドと WRITE_SINGLE_BLOCK コマンドで対応します。
  マルチプルブロックアクセス非対応のカードに対しては、READ_SINGLE_BLOCK コマンドと WRITE_SINGLE_BLOCK コマンドでアクセスします。
- OS 非依存です。

TFAT ライブラリの仕様【注】

- FAT タイプは FAT12、FAT16 です。
- ファイル名は 8.3 形式（8 文字のファイル名および 3 文字の拡張子）です。
- ドライブ数は 1 つです。
- セクタサイズは 512byte です。
- TFAT ライブラリは、V850E2M コア専用です。他のコア（V850, V850ES 等）には対応していません。
  【注】TFAT ライブラリは更新することがあります。ルネサス エレクトロニクスホームページより最新版を入手ください。（関連アプリケーションノート[1]参照）

1.3 ファイルの構成

ディレクトリおよびファイルの構成を示します。

ディレクトリ構成、 ディレクトリ名／ファイル名	備考
¥doc <DIR>	ドキュメントディレクトリ
r01an1026jj1011_v850e2ml4.pdf	アプリケーションノート（本書）
¥source <DIR>	MMC ドライバソースプログラム
¥common <DIR>	共通関数のディレクトリ
r_mtl_com.c	共通関数（ログ記録）
r_mtl_com2.h	共通関数の各種定義
r_mtl_endi.c	共通関数（エンディアン関連）
r_mtl_mem.c	共通関数（標準ライブラリ関数）
r_mtl_str.c	共通関数（標準ライブラリ関数）
r_mtl_tim.c	共通関数（ソフトウェアループタイマ）
r_mtl_tim.h	共通関数（ソフトウェアループタイマ）の各種定義
r_mtl_com.h	V850E2/ML4 動作用定義ヘッダファイル
¥mmc_driver <DIR>	MMC 用デバイスドライバのディレクトリ
r_mmc.h	MMC ドライバ共通定義
r_mmc_io.c r_mmc_io.h	MMC ドライバ SPI モード I/O モジュール
r_mmc_spi.c	MMC ドライバ SPI モード SPI を用いた通信モジュール
r_mmc_mmc.c	MMC ドライバ SPI モード MMC モジュール
r_mmc_sub.c r_mmc_sub.h	MMC ドライバ SPI モード・サブモジュール
r_mmc_usr.c	MMC ドライバ SPI モード・API ソースプログラム
¥V850E2_ML4 <DIR>	V850E2_ML4 SPI 用 SFR 定義ディレクトリ
r_mmc_sfr.h	MMC ドライバ SFR 定義
r_mmc_user_config.h	MMC ユーザ定義ヘッダファイル
¥code <DIR>	CSIH モジュール用のディレクトリ
CG_main.c	main 関数
CG_port.c CG_port.h	CSIH ポート初期化モジュール
CG_serial.c CG_serial.h	CSIH 動作制御モジュール
CG_serial_user.c	CSIH ユーザ設定モジュール
CG_macrodriver.h	状態制御定義ヘッダファイル
¥tfat_sample <DIR>	TFAT テスト用サンプルプログラムのディレクトリ
r_sample_tfat.c	MMC ドライバ動作検証用のサンプルプログラム
r_tfat_drv_if.c r_tfat_drv_if.h	TFAT ドライバソースファイル
r_data_file.h	書き込みデータ定義ヘッダファイル
r_target_io.h	周辺 I/O 定義ヘッダファイル
¥Library <DIR>	ライブラリのディレクトリ
r_tfat_lib.h	TFAT ファイルシステムヘッダファイル
libtfat_v850e2m.lib	MMC ドライバ用 TFAT ファイルシステムライブラリ
¥sample <DIR>	サンプルプロジェクトディレクトリ
TFAT_MMC_sample_for_V850E2ML4.mtpj	プロジェクトファイル

2. プログラム型定義

本プログラムの整数型の定義は以下のとおりとします。

Datatype	Typedef
unsigned char	uint8_t
unsigned short	uint16_t
unsigned long	uint32_t
signed char	int8_t
signed short	int16_t
signed long	int32_t

3. デバイスドライバ

3.1 デバイスドライバ関数概要

初期化関数

関数名	機能概要
R_mmc_Init_Driver ()	MMC ドライバ初期化処理

デバイス操作関数

関数名	機能概要
R_mmc_Init_Slot()	MMC スロット初期化処理
R_mmc_Detach()	MMC スロット停止処理
R_mmc_Chk_Detect()	MMC 挿入チェック処理

データ・アクセス操作関数

関数名	機能概要
R_mmc_Read_Data()	データ読み出し処理
R_mmc_Write_Data()	データ書き込み処理
R_mmc_Get_MmcInfo()	MMC 情報取得処理

内部使用コマンド一覧

コマンドインデックス	コマンド名
CMD0	GO_IDLE_STATE
CMD1	SEND_OP_COND
CMD9	SEND_CSD
CMD10	SEND_CID
CMD12	STOP_TRANSMISSION
CMD13	SEND_STATUS
CMD17	READ_SINGLE_BLOCK
CMD18	READ_MULTIPLE_BLOCK
CMD24	WRITE_BLOCK
CMD25	WRITE_MULTIPLE_BLOCK
CMD58	READ_OCR
CMD59	CRC_ON_OFF

【注】 未サポート・コマンドに対しては、ユーザ側で対応してください。

3.2 関数詳細

3.2.1 MMC ドライバ初期化処理(R_mmc_Init_Driver)

項目	内容
プロトタイプ	void R_mmc_Init_Driver(void)
引数	なし
説明	MMC ドライバの初期化を行います。 MMC 制御用 SFR の初期化、スロット毎の制御ポートと RAM を初期化します。 本関数はシステム起動時に一度だけ呼び出してください。
戻り値	なし

3.2.2 MMC スロット初期化処理(R_mmc_Init_Slot)

項目	内容
プロトタイプ	int16_t R_mmc_Init_Slot(uint8_t SlotNo)
引数	uint8_t SlotNo ：スロット番号
説明	引数で指定されたスロットに対する RAM およびポートの初期化を行います。 また、MMC に対して初期化処理を行います。 本関数はカード挿入検出時に呼び出してください。
戻り値	初期化結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー MMC_ERR_HARD : ハードウェアエラー MMC_ERR_CRC : CRC エラー MMC_ERR_IDLE : Idle state エラー MMC_ERR_OTHER : その他エラー

3.2.3 MMC スロット停止処理(R_mmc_Detach)

項目	内容
プロトタイプ	int16_t R_mmc_Detach(uint8_t SlotNo)
引数	uint8_t SlotNo ：スロット番号
説明	指定スロット MMC 取り外しの処理を行います。 MMC 制御用 SFR の初期化、制御 port の開放、制御用 RAM の初期化を行います。 本関数はカード引抜き検出時に呼び出してください。
戻り値	取り外し処理の結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー

3.2.4 MMC 挿入チェック処理(R_mmc_Chk_Detect)

項目	内容
プロトタイプ	int16_t R_mmc_Chk_Detect(uint8_t SlotNo, uint8_t *pDetSts)
引数	uint8_t SlotNo : スロット番号 uint8_t *pDetSts : MMC 挿入状態格納先バッファポインタ
説明	引数で指定されたスロットに対して MMC 挿入状態のチェックを行います。 戻り値が MMC_OK の場合、MMC 挿入状態格納先バッファ(pDetSts)には MMC 挿入検出端子状態が格納されます。 • MMC_TRUE ：MMC 挿入検出端子 Active • MMC_FALSE：MMC 挿入検出端子 Non Active この処理内でチャタリングの除去は行いません。 上位にて必要なチャタリングの除去を行ってください。 定周期でのポーリングによるメディアの挿入確認を推奨します。
戻り値	チェック結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー

3.2.5 データ読み出し処理(R_mmc_Read_Data)

項目	内容
プロトタイプ	int16_t R_mmc_Read_Data(uint8_t SlotNo, uint32_t BlkNo, uint32_t BlkCnt, uint8_t *pData, uint8_t Mode)
引数	uint8_t SlotNo : スロット番号 uint32_t BlkNo : 読み出し開始ブロック番号 uint32_t BlkCnt : 読み出しブロック数 uint8_t *pData : 読み出しデータ格納バッファポインタ uint8_t Mode : 読み出しデータ転送モード
説明	MMC からブロック(512byte)単位でデータの読み出しを行います。 指定ブロックから指定ブロック数分、データを読み出します。 転送モード(Mode)は MMC_MODE_NORMAL（データを引数のデータ格納バッファ(pData)に転送するモード）を選択してください。 MMC からの読出しは、R_mmc_Get_MmcInfo()関数から渡される MMC 情報のうち、カード種別(MmcInfo.Card)が‘ MMC_CARD_UNDETECT ’でない場合のみ可能です。 最大ブロック番号は、R_mmc_Get_MmcInfo()関数から渡される’pMmcInfo.MaxBlkNum’です。 最大ブロック数は、’pMmcInfo.MaxBlkNum’+1 です。
戻り値	読み出し結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー MMC_ERR_HARD : ハードウェアエラー MMC_ERR_CRC : CRC エラー MMC_ERR_OTHER : その他エラー

3.2.6 データ書き込み処理(R_mmc_Write_Data)

項目	内容
プロトタイプ	int16_t R_mmc_Write_Data(uint8_t SlotNo, uint32_t BlkNo, uint32_t BlkCnt, uint8_t *pData, uint8_t Mode)
引数	uint8_t SlotNo : スロット番号 uint32_t BlkNo : 書き込み開始ブロック番号 uint32_t BlkCnt : 書き込みブロック数 uint8_t *pData : 書き込みデータ格納バッファポインタ uint8_t Mode : 書き込みデータ転送モード
説明	MMC へブロック(512byte)単位でデータの書き込みを行います。 指定ブロックから指定ブロック数分、データを書き込みます。 転送モード(Mode)は MMC_MODE_NORMAL（データを引数のデータ格納バッファ(pData)から転送するモード）を選択してください。 MMC への書き込みは、R_mmc_Get_MmcInfo()関数から渡される MMC 情報のうち、カード種別(MmcInfo.Card)が ‘ MMC_CARD_UNDETECT ’ でない場合のみ可能です。 最大ブロック番号は、R_mmc_Get_MmcInfo()関数から渡される’pMmcInfo.MaxBlkNum’です。 最大ブロック数は、’pMmcInfo.MaxBlkNum’+1 です。
戻り値	MMC 情報取得結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー MMC_ERR_HARD : ハードウェアエラー MMC_ERR_WP : ライトプロテクトエラー MMC_ERR_OTHER : その他エラー

3.2.7 MMC 情報取得処理(R_mmc_Get_MmcInfo)

項目	内容
プロトタイプ	int16_t R_mmc_Get_MmcInfo(uint8_t SlotNo, MMC_INFO* pMmcInfo)
引数	uint8_t SlotNo : スロット番号 MMC_INFO* pMmcInfo : MMC 情報格納バッファポインタ
説明	MMC 情報を返します。 MMC 情報格納バッファ(pMmcInfo)には MMC 情報が格納されます。 • pMmcInfo.Card : カード種別 — MMC_CARD_UNDETECT : カード未検出 — MMC_CARD_MMC : MMC — MMC_CARD_OTHER : その他カード pMmcInfo.WProtect : ライトプロテクト状態 — MMC_NO_PROTECT : ライトプロテクト解除 — MMC_W_PROTECT_SOFT : ソフトライトプロテクト • pMmcInfo.MemSize : カード容量(byte) • pMmcInfo.MaxBlkNum : 最大ブロック番号（メディアの最大ブロック番号） ‘pMmcInfo.MemSize’が 0xFFFFFFFF の場合は、’pMmcInfo.MaxBlkNum‘+1 が、 メディアの持つブロック数（1 ブロック=512 バイト）を示します。 カード容量の計算が必要な場合、(’pMmcInfo.MaxBlkNum‘+1)×512 で計算してください。
戻り値	書き込み結果を返します。 MMC_OK : 正常終了 MMC_ERR_PARAM : パラメータエラー MMC_ERR_OTHER : その他エラー

3.3 データ構造体

データ構造体を以下に示します。

MMC 情報構造体定義

```
typedef struct {
  uint8_t    Card;             /* Card type                      */
  uint8_t    WProtect;         /* Write-protection status        */
  uint32_t   MemSize;          /* Card capacity                  */
  uint32_t   MaxBlkNum;        /* The number of the max blocks   */
} MMC_INFO;                    /* total 12byte                   */
```

3.4 マクロ定義

マクロ定義を以下に示します。

```
/*----- Definitions of return value -----*/
#define MMC_OK              (int16_t)( 0)        /* Successful operation     */
#define MMC_ERR_PARAM       (int16_t)(-1)        /* Parameter error          */
#define MMC_ERR_HARD        (int16_t)(-2)        /* Hardware error           */
#define MMC_ERR_CRC         (int16_t)(-3)        /* CRC error                */
#define MMC_ERR_WP          (int16_t)(-4)        /* Write-protection error   */
#define MMC_ERR_MBLKCMD     (int16_t)(-5)        /* Multi-block command error*/
#define MMC_ERR_IDLE        (int16_t)(-6)        /* Idle state error         */
#define MMC_ERR_OTHER       (int16_t)(-7)        /* Other error              */

/*----- Definitions of log type -----*/
#define MMC_LOG_ERR         1                    /* Log type : Error         */

/*----- Definitions of flag -----*/
#define MMC_TRUE            (uint8_t)0x01        /* Flag "ON"                */
#define MMC_FALSE           (uint8_t)0x00        /* Flag "OFF"               */

/*----- Definition of card type -----*/
#define MMC_CARD_UNDETECT   (uint8_t)0x00        /* Card is not found        */
#define MMC_CARD_MMC        (uint8_t)0x01        /* MMC                      */
#define MMC_CARD_OTHER      (uint8_t)0xFF        /* Other card               */

/*----- Definitions of write-protection status -----*/
#define MMC_NO_PROTECT      (uint8_t)0x00        /* None setting             */
#define MMC_W_PROTECT_SOFT  (uint8_t)0x02        /* Software write-protection*/
```

4. 設定例

4.1 共通関数 r_mtl_XXX の可変データ設定例

各システムのリソースに合わせて設定をする部分です。

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

ファイル毎に抜粋を示し、詳細な解説を加えます。

4.1.1 r_mtl_com.h

共通で使用される共通関数のヘッダです。

r_mtl_com.h は MCU およびシステムクロック別に個々に用意する必要があります。

ご使用の環境に合わせて r_mtl_com.h をインクルードしてください。

異なる MCU やシステムクロックで動作させる場合は、ユーザでご用意ください。

使用 MCU – システムクロック	インクルードディレクトリ（source ディレクトリ以降）
V850E2/ML4 - 50MHz	¥common¥V850E2_50MHz

(1) ループタイマの定義

— ソフトウェア・ループタイマを使用する場合、r_mtl_tim.h をインクルードします。
主にデバイスドライバが待ち時間を確保するために、使用します。
ソフトウェア・ループタイマを使用しない場合は、下記インクルードをコメントにしてください。
下記の例は、ソフトウェア・ループタイマを使用する場合の例です。
本ドライバ使用時は、インクルードしてください。また、r_mtl_tim.h からシステムクロックに合った
マクロを定義してください。V850E2 シリーズの MCU を 50MHz で動作する場合は
"MTL_TIM_V850E2__50_0MHz"を定義します。

```
/* When not using the loop timer, put the following 'include' as comments. */
#define MTL_TIM_V850E2__50_0MHz
#include "r_mtl_tim.h"
```

(2) エンディアンタイプ定義

— リトルエンディアン／ビッグエンディアンのどちらかの指定を行います。

```
#if ( (defined(__LIT)) || (!defined(__BIG)) )
#define MTL_MCU_LITTLE              /* Little Endian           */ /** SET **/
#endif
```

- (3) 使用する標準ライブラリのタイプの定義

— 使用する標準ライブラリのタイプを定義してください。
下記に示す処理を標準ライブラリで使用する場合は、下記マクロ定義をコメントにしてください。
下記の例は、コンパイラ添付のライブラリを使用しない場合の例です。

```
/* Specify the type of user standard library.                          *//** SET **/
/* When using the compiler-bundled library for the following processes,   *//** SET **/
/* put the following 'define' as comments.                                 *//** SET
**/
/* memcmp()/memmove()/memcpy()/memset()/strcat()/strcmp()/strcpy()/strlen()*//** SET **/
#define MTL_USER_LIB                /* use optimized library            */ /** SET **/
```

4.1.2 r_mtl_tim.h

r_mtl_com.h にて、ループタイマを定義した場合に、インクルードされます。

使用する MCU やクロック、ウェイトに依存します。

環境に合うものがない場合は、ユーザにて定義を作成してください。

```
/* Define the counter value for the timer.                                  */
/* Specify according to the user MCU, clock and wait requirements.          */
/* Set the reference value to 10% more than the actual calculated value.    */
/*==========================================================================*/
/* Macro definitions                                                        */
/*==========================================================================*/
#ifdef MTL_TIM_V850E2__50_0MHz
/* Setting for 50.0MHz no wait (Compile Option "-optimize=2 -size")          */
#define MTL_T_250NS         2       /* loop times of 250ns     */ /** SET **/
#define MTL_T_500NS         5       /* loop times of 500ns     */ /** SET **/
#define MTL_T_1US          11       /* loop times of   1us     */ /** SET **/
#define MTL_T_2US          24       /* loop times of   2us     */ /** SET **/
#define MTL_T_4US          49       /* loop times of   4us     */ /** SET **/
#define MTL_T_5US          61       /* loop times of   5us     */ /** SET **/
#define MTL_T_10US        124       /* loop times of  10us     */ /** SET **/
#define MTL_T_20US        249       /* loop times of  20us     */ /** SET **/
#define MTL_T_30US        374       /* loop times of  30us     */ /** SET **/
#define MTL_T_50US        624       /* loop times of  50us     */ /** SET **/
#define MTL_T_100US      1249       /* loop times of 100us     */ /** SET **/
#define MTL_T_200US      2499       /* loop times of 200us     */ /** SET **/
#define MTL_T_300US      3749       /* loop times of 300us     */ /** SET **/
#define MTL_T_400US      4999       /* loop times of 400us     */ /** SET **/
#define MTL_T_1MS       12499       /* loop times of   1ms     */ /** SET **/
#endif
```

4.2 MMC ドライバ可変データ設定例

各システムのリソースに合わせて設定をする部分です。

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

ファイル毎に抜粋を示し、詳細な解説を加えます。

4.2.1 r_mmc.h（ドライバ共通定義）

- (1) デバイス数、デバイス番号の定義
  - メモリカードのスロット数を定義してください。
  - V850E2/ML4 では、MMC スロットが 1 つのみを想定しているため、複数デバイスをサポートしていません。

```
/* Define number of required card slots. (1-N slots)                              */
/* Define slot number in accordance with the number of card slots to be connected.   */
/* Define number of slots (devices). */
#define MMC_SLOT_NUM          1                   /* 1 slots              */ /** SET **/

/* Define slot number. */
#define MMC_SLOT0             0                   /* Slot 0               */ /** SET **/
#define MMC_SLOT1             1                   /* Slot 1               */ /** SET **/
```

- (2) SPI モード マルチブロックコマンド非対応カードの場合の定義
  - デフォルトでは、MULTIPLE_BLOCK コマンド非対応カードの場合は、READ_SINGLE_BLOCK コマンドと WRITE_SINGLE_BLOCK コマンドで対応する設定のため、このままの設定を推奨します。

```
/*  When use the card which does not support a multi-block command, please define it.*/
/*  Use single block commands in the case of the card which does not support multiple block*/
/*  commands.                                                                      */
#define MMC_SBLK_CMD        /* Support single block commands    */ /** SET **/
```

- (3) 対象メディアの定義
  - MMC_SUPPORT_MMC を定義してください。

```
/*  Please define the media to support.                                           */
#define MMC_SUPPORT_MMC                     /* MMC             */ /** SET **/
```

4.2.2 r_mmc_user_setting.h（MCU 個別定義）

(1) 使用 MCU の選択

r_mmc_user_setting.h は MCU 依存します。

ご使用の環境に合わせて下記の表に示すディレクトリにある mmc_user_setting.h をインクルードしてください。

環境に合うものがない場合は、ユーザにて定義を作成してください。

使用 MCU - 通信モジュール	インクルードディレクトリ（source ディレクトリ以降）
V850E2/ML4	¥mmc_driver¥V850E2_ML4

(2) 通信モジュールのチャネル番号の定義

MMC_CSIH_CHANNEL マクロ（CSIH 使用時）、で使用する通信モジュールの番号を定義してください。

[V850E2/ML4 - CSIH 使用時]

```
/*----- CSIH Channel Select (CSIHx / x = 0 or 1) -----*/
#define MMC_CSIH_CHANNEL    1                       /** SET **/
```

(3)使用制御ポートの定義

— DETECT（カード検出）信号および CS（カードセレクト）信号を接続回路に合わせて定義してください。
下記に設定例を示します。

```
/*----- Define the control port. -----*/
#define MMC_CS_PORT         P2          /* CS Port Setting */       /** SET **/
#define MMC_CS_BIT          0x0020U     /* CS Bit Select   */       /** SET **/

#define MMC_DETECT_PORT     PPR7【注】    /* DETECT Port Setting */   /** SET **/
#define MMC_DETECT_BIT      0x0001U     /* DETECT Bit Select   */   /** SET **/
```

【注】本レジスタは I/O ポートが入力の際に用いる、読み出し専用レジスタです。詳細は V850E2/ML4 ハードウェアマニュアルをご覧下さい。

(4) 通信タイムアウト検出処理の定義

— 通信中のタイムアウト検出処理を省略することが出来ます。
省略する場合、MMC_NOCHK_TIMEOUT マクロを定義してください。定義した場合は処理速度が向上するメリットがありますが、通信機能に異常が発生した場合にプログラムが停止する可能性があるデメリットもあります。
省略しない場合は、タイムアウト時間を設定してください。

- MMC_T_SPI_WAIT で時間の単位を設定します。設定するマクロは r_mtl_tim.h から選択してください。
- MMC_SPI_TX_WAIT(送信時)のマクロでデータ送信時のタイムアウト時間を定義してください。
- MMC_SPI_RX_WAIT(受信時)のマクロでデータ受信時のタイムアウト時間を設定してください。
- 各タイムアウト時間マクロの設定値は（タイムアウト時間 / 単位）となります。

```
/*-----------------------------------------------------------------------*/
/* Macro "MMC_NOCHK_TIMEOUT" omits detecting timeout during communication.  */
/* If user omits detecting timeout, please define this macro.              */
/* If this macro is defined, processing speed would be increased.          */
/*-----------------------------------------------------------------------*/
#define MMC_NOCHK_TIMEOUT  /* No Check Communication Timeout   */ /** SET **/

/*-----------------------------------------------------------------------*/
/* If MMC_NOCHK_TIMEOUT would be not defined, please set timeout time.
/* MMC_T_SCI_WAIT is unit of measuring timeout.
/* Please select value from "r_mtl_tim.h"
/* Please set value of (timeout time/unit) to MMC_SCI_TX_WAIT(transmitting)
/* and MMC_SCI_RX_WAIT(receiving).
/*-----------------------------------------------------------------------*/
/* SCI transmit&receive completion waiting polling time        */ /** SET **/
#define MMC_T_SCI_WAIT     (uint32_t)MTL_T_100US
/* SCI transmission completion waiting time 50 * 1ms = 50ms    */ /** SET **/
#define MMC_SCI_TX_WAIT    (uint32_t)(MTL_T_1MS*50)
/* SCI receive completion waiting time  50 * 1ms = 50ms        */ /** SET **/
#define MMC_SCI_RX_WAIT    (uint32_t)(MTL_T_1MS*50)
```

•

(5) 通信ボーレートの定義

— 通信ボーレートを定義してください。

使用する SIO の定義を修正する場合は、各々の SIO レジスタに応じた SFR 設定が必要になりますので、MCU のデータシートを参照し、システムに応じた SFR 設定を行ってください。

特に、通信速度設定に関して、Identification mode/Data Transfer mode それぞれのカード仕様書の tODLY を満たすように設定する必要があります。

さらに、カード仕様書では、Identification mode では、tOD（100kHz≦tOD≦400kHz）、Data Transfer mode では、tPP（0.1MHz≦tPP≦20MHz(*)）を満たすように設定する必要があります。

なお、tOD, tPP は、本システムにおいて、クロック周波数を示すことになります。

MCU により、サポート可能な SIO のクロック周波数が変わりますので、MCU のデータシートを参照してください。

MMC_UBRG_IDENTIFICATION マクロには Identification mode 時のクロック設定を設定してください。

MMC_UBRG_D_TRANSFER マクロには Transfer mode 時のクロック設定を定義してください。

MMC_CLK_D_TRANSFER マクロには Transfer mode 時のクロック周波数を定義してください。

本プログラムは、Identification mode 時に 312.5kHz、Transfer mode 時に 12.5MHz に設定しています。

```
/*-----------------------------------------------------------------------------------------*/
/* Define the value of the bit rate register according to a communication baud rate.      */
/* Set the frequency of CLK to 6MHz or less.                                               */
/* The possible maximum transfer frequency of CLK is depends on hardware circuit          */
/* and MCU conditions.                                                                     */
/*   Refer to MCU hardware manual/memory card specifications and specify the buad rate.    */
/* When operating card with SPI mode,                                                      */
/* specify the following two definitions of Identification mode and Data Transfer mode.    */
/* Specify the definition to meet tODLY of both Identification mode and Data Transfer mode.*/
/* In addition, meet tOD (100kHz <= tOD <= 400kHz) at Identification mode                  */
/* and tPP (0.1MHz <= tPP <= 20MHz ) at Data Transfer mode.                                */
/* The maximum frequency depends on MCU type.                                              */
/*                                                                                         */
/* BRR = PCLK / (2m * B * 2)                                                               */
/* PCLK: Operating frequency [MHz]                                                         */
/* B   : Bit rate [bit/s]                                                                  */
/* m   : Determined by the SMR settings shown in the following table.                      */
/*                                                                                         */
/*-----------------------------------------------------------------------------------------*/
/* PCLK = 50MHz, n=0 */
#define MMC_UBRG_IDENTIFICATION (uint16_t)0x0050 /* BRR identification mode setting */ /** SET **/
                                        /* ++++-------- <= 400kHz                    */ /** SET **/
#define MMC_UBRG_D_TRANSFER     (uint16_t)0x0002 /* BRR data Transfer mode setting  */ /** SET **/
                                        /* ++++-------- <= 20MHz                   */ /** SET **/
#define MMC_CLK_D_TRANSFER      (uint32_t)600000/* Data Transfer mode clock frequency *//** SET**/
```

5. MCU との接続方法と使用する MCU リソース

5.1 使用する MCU リソース

本プログラムは、以下の制御を行っています。

データの入出力を、クロック同期式インタフェース（CSIH、P バスクロックで動作）で、制御します。

CSIH を割り当てる際には、高速動作させるため CMOS 出力可能な端子割り当てと CMOS 出力設定をしてください。

送信制御は、送信バッファの空きを検出して、割り込みを使用せずに送信割り込み要求ビットを利用しています。したがって、割り込み関連を以下のように設定しています。

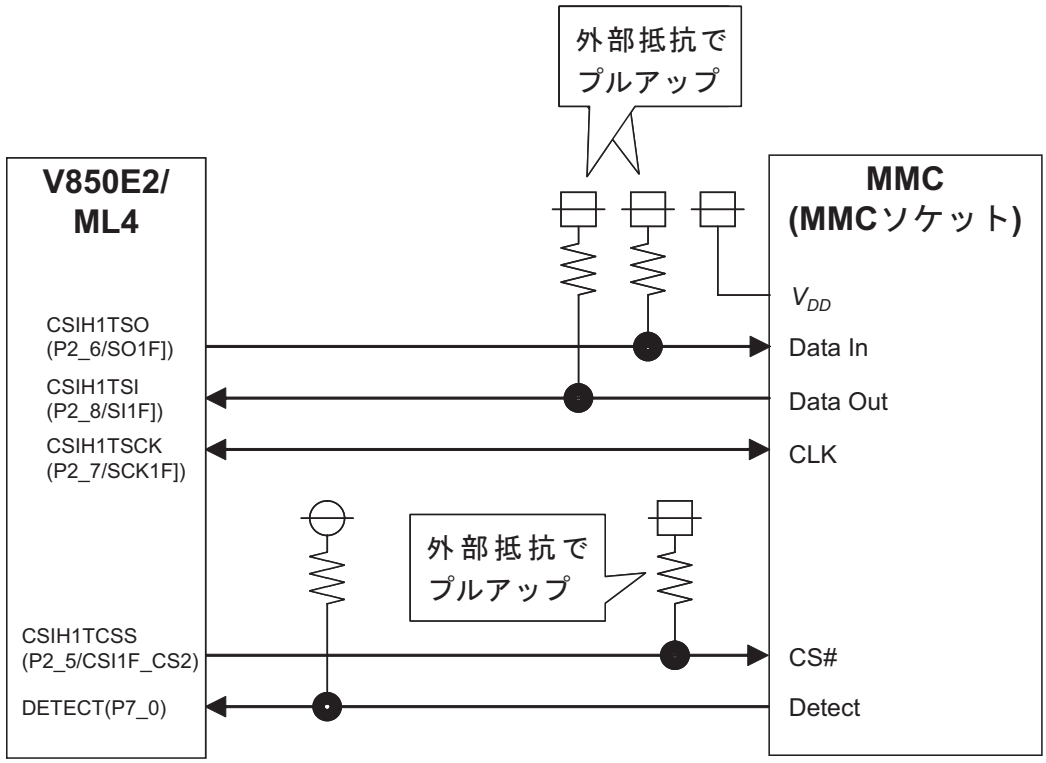
- 割り込み優先レベルを、レベル 0（割り込み禁止）に設定。
- MMC の CS#端子を MCU の Port に接続し、MCU 汎用ポート出力で制御する。

使用するリソース
CSIH
CS#用ポート　1 本/Card
カード検出用ポート　1 本/Card
電源制御用ポート　1 本/Card

5.2 MCU との接続方法

V850E2/ML4 との接続例を示します。

他の MCU の場合であっても、同様の接続となります。



活線挿抜制御のためには、カードへの電源制御回路が必要になります。
（電源制御回路は、示しておりません。）
カードの挿入によるカード検出後、カードへの電源供給を行ってください。
カードを抜く前に、カードへの電源切断を行ってください。
同一 SPI バス上で、複数のカードを使用する場合、カード毎に CS#用ポート／CardDetect ポートを用意してください。

6. アプリケーション作成時の注意事項

6.1 使用上の注意事項

- 使用にあたっては、ハードウェアに合わせてソフトウェアを設定してください。
- メモリカードを非アクティブにし、MCU-メモリカード間の信号を Hi-z にし、デバイスへの電源供給を停止させた後、メモリカードを抜いてください。動作中にデバイスを抜いた場合、デバイスが壊れる可能性があります。
- 活線挿抜対応の回路を実現していない場合、活線挿抜を行うと電源等が不安定になり MCU がリセット状態になる可能性があります。

6.2 組み込み時の注意事項

6.2.1 開発環境

[開発ホスト]

- Windows XP・Windows NT 4.0・Windows 2000・Windows Me・Windows 98・Windows 95

弊社の開発環境を以下に示します。

ユーザアプリケーション開発時は以下のバージョンより新しいものをご使用下さい。

[ソフトウェアツール]

統合開発環境

- CubeSuite+ V1.00.01

C コンパイラ

- CubeSuite+ CX コンパイラ V1.20

[デバッグツール]

エミュレータデバッガ

- V850E2M E1(JTAG)

エミュレータソフトウェア

- V850E2 Debugger V.1.00.00

[評価ボード]

- V850E2/ML4 CPU ボード (型名：R0K0F4022C000BR)

6.2.2 インクルードするファイル

MMC ドライバを組み込む場合は、r_mtl_com.h と r_mmc.h をインクルードしてください。

r_mtl_com.h を先にインクルードする必要があります。

6.3 ROM/RAM/スタックサイズ

本プログラムが使用する ROM/RAM サイズおよびスタックサイズは以下のとおりです。

ROM/RAM サイズ

分類（セクション名）	サイズ
ROM（.text .data）	約 30.8[kByte]
RAM（.sbss）	約 12.0[kByte]

スタックサイズ（MMC ドライバ）

関数名	スタックサイズ [byte]
R_mmc_Init_Driver	16
R_mmc_Init_Slot	136
R_mmc_Detach	16
R_mmc_Chk_Detect	8
R_mmc_Read_Data	124
R_mmc_Write_Data	132

スタックサイズ（TFAT ライブラリ）

関数名	スタックサイズ [byte]
R_tfat_f_mount	0
R_tfat_f_open	224
R_tfat_f_close	64
R_tfat_f_read	84
R_tfat_f_write	128
R_tfat_f_lseek	112
R_tfat_f_truncate	96
R_tfat_f_sync	56
R_tfat_f_opendir	148
R_tfat_f_readdir	80
R_tfat_f_getfree	100
R_tfat_f_stat	164
R_tfat_f_mkdir	228
R_tfat_f_unlink	172
R_tfat_f_chmod	164
R_tfat_f_utime	160
R_tfat_f_rename	236
R_tfat_f_forward	80

6.4 カードの挿抜検出に関する注意事項

カードのコネクタにあるカード検出端子を使用することで、R_mmc_Chk_Detect()を使ってカード挿抜の検出が可能です。したがって、カード検出のために一定周期でのポーリングによるカードの挿入確認を推奨します。

また、通信中においてカードが抜かれた状態になった場合、コマンドに対するレスポンス異常となり、結果的にドライバがエラーを返します。

考慮が必要な項目として、通信中に一瞬挿抜が行われた場合が挙げられます。

以下のような場合、ドライバがエラーを返さない可能性があります。

- 周期内の抜挿はドライバでは検出できず、カードからレスポンス異常が無い場合は正常動作を行います。
- 書き込み中に一瞬抜挿すると、ドライバが書き込み完了と認識してしまう可能性があります。これは、書込み Busy 信号解除は DataIn 端子の"H"で検出する仕様であるためです。（DataIn 端子はプルアップされています。）

ハードウェア割り込み制御やポーリング周期の見直し等でシステムに適した方法で解決してください。

6.5 ポートの割り当てとカードの挿抜に関するポートの Hi-z 化処理の注意事項

- カードの挿入においては、カードの CS#, DataIn, DataOut, CLK 信号を Hi-z 状態にし、カードを挿入してください。その後、カードへの電源を供給してください。
- カードの抜去においては、カードへの電源供給停止後、カードの CS#, DataIn, DataOut, CLK 信号を Hi-z 状態にしてからカードを抜いてください。
- カードの CS#, DataIn, DataOut, CLK は、MCU の SIO、ポート端子に割り当てられていますが、そのポートが他のリソースに割り当てられている場合が想定されますので、本ドライバでは Hi-z 処理を行っておりません。したがって、カードの挿抜においては上位側で MCU 端子の Hi-z 処理をお願いします。

7. TFAT ライブラリ

7.1 TFAT ライブラリ関数概要

本プログラムで使用している TFAT ライブラリ API 関数は以下の通りです。

ファイル操作関数

関数名	機能概要
R_tfat_disk_initialize ()	**外部メディア初期化処理関数**
R_tfat_f_mount ()	FAT **ファイルシステム作業領域登録処理関数**
R_tfat_f_open ()	**ファイルオープン処理関数**
R_tfat_f_read ()	**ファイルデータ読み出し処理関数**
R_tfat_f_write ()	**ファイルデータ書き出し処理関数**
R_tfat_f_close ()	**ファイルクローズ処理関数**
R_tfat_f_lseek ()	**ファイルリード/ライトポインタ移動処理関数**

7.2 関数詳細

7.2.1 メディア初期化処理(R_tfat_disk_initialize)

項目	内容
プロトタイプ	DSTATUS R_tfat_disk_initialize (void)
引数	なし
説明	MMC の初期化を行います。 MMC ドライバの MMC 挿入チェック処理関数 R_mmc_Chk_Detect、MMC スロット初期化関数 R_mmc_Init_Slot を呼び出します。 本関数は、正常終了するまで何度も呼び出されます。
戻り値	初期化結果を返します。 TFAT_RES_OK : 初期化正常終了 TFAT_STA_NODISK : MMC 非挿入 TFAT_STA_NOINIT : 未初期化状態

7.2.2 FAT ファイルシステム作業領域登録処理(R_tfat_f_mount)

項目	内容
プロトタイプ	FRESULT R_tfat_f_mount (uint8_t drv, FATFS *fs)
引数	uint8_t drv : 論理ドライブ番号 FATFS *fs : ワークエリアポインタ
説明	FAT ファイルシステム作業領域を MMC に登録します。 操作対象のドライブに対して開かれているファイルやディレクトリがあった場合、それらは全て無効になります。 本関数は、ドライブの状態に関わらず常に正常終了します。
戻り値	登録結果を返します。 TFAT_FR_OK : 正常終了

7.2.3 ファイルオープン処理(R_tfat_f_open)

項目	内容
プロトタイプ	FRESULT R_tfat_f_open (FIL *fp, const uint8_t *path, uint8_t mode)
引数	FIL *fp : オブジェクト構造体ポインタ const uint8_t *path : オープンファイル文字列ポインタ uint8_t mode : アクセス/オープンモードフラグ
説明	既存のファイルを開きます。また、新規のファイルを作成します。 本関数が成功すると、ファイル･オブジェクトが生成され、以降そのファイルに対するアクセスに使用します。ファイル・アクセスを開始する前に、ドライブに FAT ファイルシステム作業領域を与える必要があり、この初期化の後、そのドライブに対して全てのファイル関数が使えるようになります。
戻り値	結果を返します。 TFAT_FR_OK : 正常終了 TFAT_FR_NO_FILE : ファイル未検出 TFAT_FR_EXIST : 同名オブジェクト存在 TFAT_FR_DENIED : オブジェクト操作拒否 TFAT_FR_RW_ERROR : リード/ライトエラー

7.2.4 ファイルデータ読み出し処理(R_tfat_f_read)

項目	内容
プロトタイプ	FRESULT R_tfat_f_read (FIL *fp, void *buff, uint16_t btr, uint16_t *br)
引数	FIL *fp : オブジェクト構造体ポインタ void *buff : データ格納バッファポインタ uint16_t btr : 読み出しバイト数 uint16_t *br : バイト数格納変数ポインタ
説明	ファイルからデータを読み出します。 本関数は、MMC ドライバ R_tfat_disk_read()を呼びます。 読み出し開始位置は、現在のリード/ライトポインタからになります。リード/ライトポインタは、読まれたバイト数だけ進みます。関数が正常終了した後は、*br を確認することを推奨します。*br が btr よりも小さいときは、読み込み中にファイルの終端に達したことを示しています。
戻り値	結果を返します。 TFAT_FR_OK : 正常終了 TFAT_FR_DENIED : オブジェクト操作拒否 TFAT_FR_RW_ERROR : リード/ライトエラー TFAT_FR_NOT_READY : ドライブ初期化失敗

7.2.5 ファイル書き込み処理(R_tfat_f_write)

項目	内容
プロトタイプ	FRESULT R_tfat_f_write (FIL *fp, const void *buff, uint16_t btw, uint16_t *bw)
引数	FIL *fp : オブジェクト構造体ポインタ const void *buff : データ格納バッファポインタ uint16_t btw : 書き出しバイト数 uint16_t *bw : バイト数格納変数ポインタ
説明	ファイルにデータを書き込みます。 本関数は、MMC ドライバ R_tfat_disk_write()を呼びます。 書き込み開始位置は、リード/ライトポインタの位置からになります。リード/ライトポインタは実際に書き込まれたバイト数だけ進みます。関数が正常終了した後は、要求したバイト数が書き込まれたかどうか*bw を確認することを推奨します。*bw が btw よりも小さいときは、MMC がフルであることを意味します。
戻り値	結果を返します。 TFAT_FR_OK : 正常終了 TFAT_FR_DENIED : オブジェクト操作拒否 TFAT_FR_RW_ERROR : リード/ライトエラー

7.2.6 ファイルクローズ処理(R_tfat_f_close)

項目	内容
プロトタイプ	FRESULT R_tfat_f_close(FIL *fp)
引数	FIL *fp　　オブジェクト構造体ポインタ
説明	開いているファイルを閉じます。何らかの書き込みの行われたファイルの場合、キャッシュされた状態（リード/ライトバッファ上のデータ、変更された FAT）はディスクに書き戻されます。関数が正常終了すると、そのファイルオブジェクトは無効になり、そのメモリも解放できます。
戻り値	結果を返します。 TFAT_FR_OK　　: 正常終了

7.2.7 ファイルリード/ライトポインタ移動処理(R_tfat_f_lseek)

項目	内容
プロトタイプ	FRESULT R_tfat_f_lseek(FIL *fp, uint32_t ofs)
引数	FIL *fp　　オブジェクト構造体ポインタ Uint32_t ofs　　移動先オフセット
説明	ファイルのリード/ライトポインタ（次にリード/ライトされるバイトのオフセット）を移動します。オフセットの原点はファイル先頭です。書き込みモードでファイルサイズより大きな値を指定すると、そこまでファイルサイズが拡張され、拡張された部分のデータは未定義となります。データを遅延無く高速に書き込みたいときは、予めこの関数で必要なサイズまでファイルサイズを拡張しておくことを推奨します。
戻り値	結果を返します。 TFAT_FR_OK　　: 正常終了 TFAT_FR_RW_ERROR　: リード/ライトエラー

8. サンプルプログラム

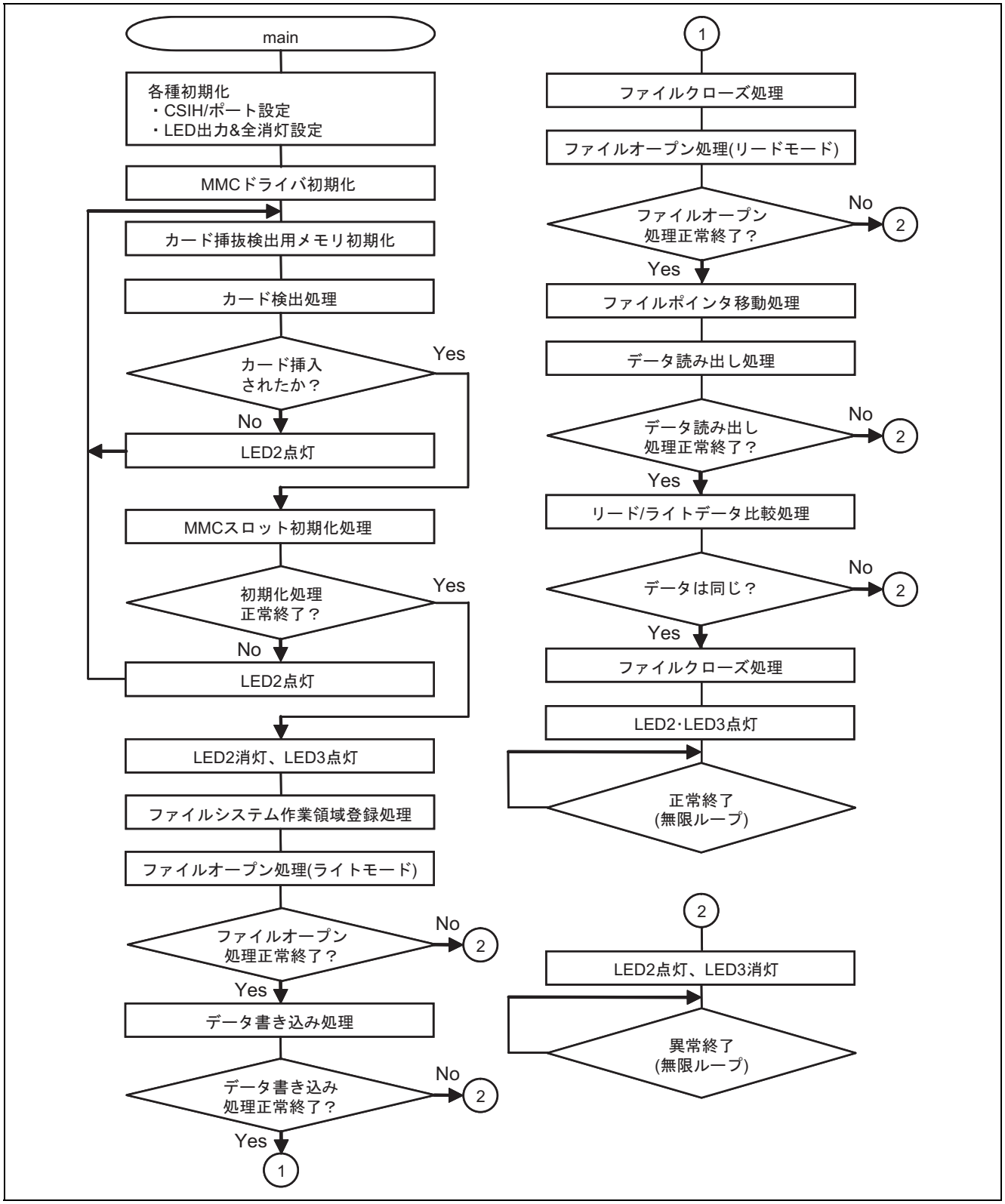
ここではサンプルプログラムについて説明します。

8.1 処理内容

- MMC を挿入すると、MMC 内にファイルを作成し、2kByte のテキストデータをファイルに書き込みます。書き込んだデータを確認するために、ファイルの内容全体を読み取り、プログラムの書き込みバッファデータと比較します。
- 上記を最終ブロックまで実行した後、カードを抜いて動作終了です。
- プログラムの進行状況および結果を LED で表示します。表示の内容は下表を参照してください。

LED3(P4_4)	LED2(P4_3)	意味
ON	OFF	**実行中**
OFF	ON	**カード未挿入、エラー終了**
ON	ON	**正常終了**

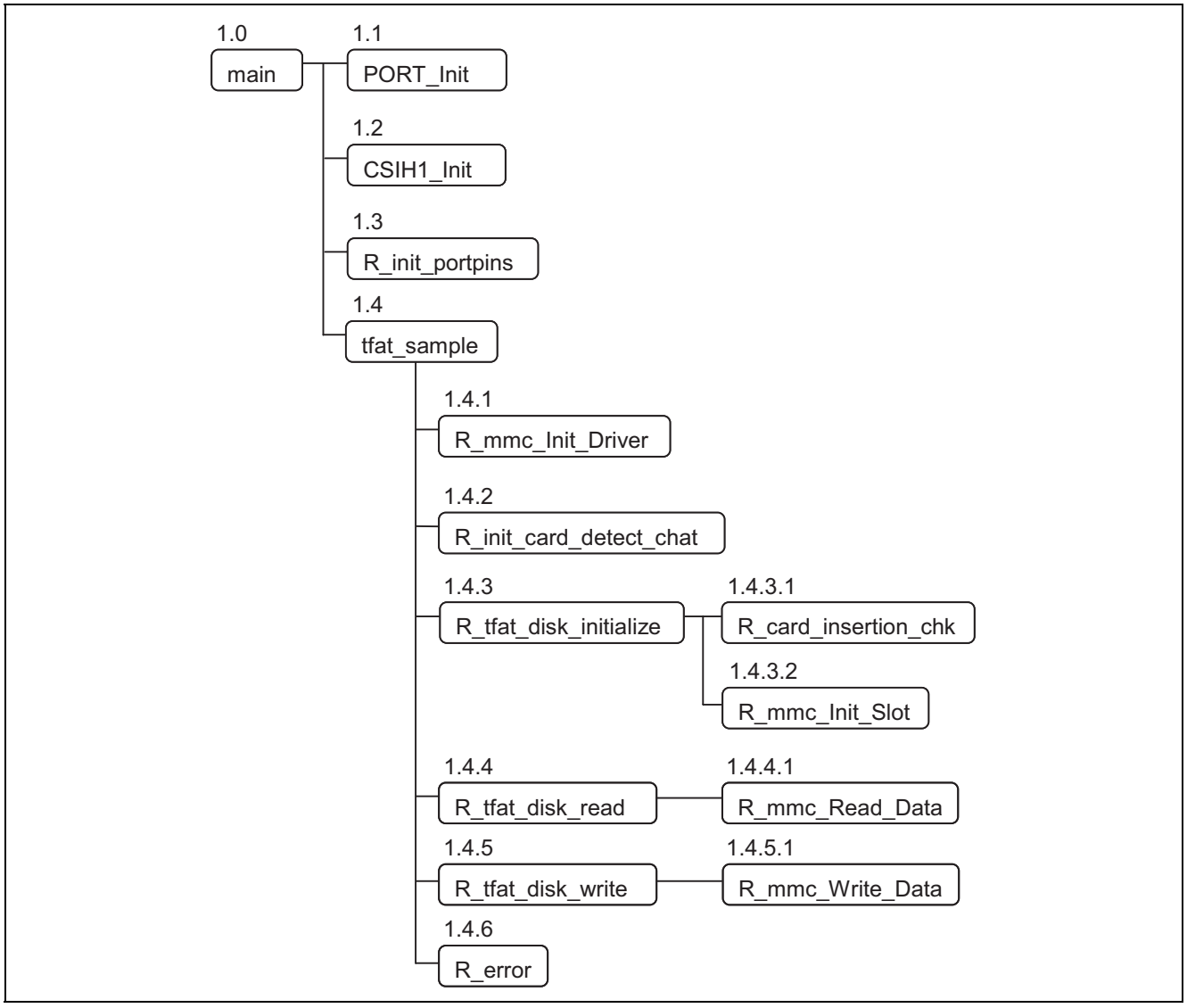
8.2 処理の流れ


main
各種初期化
・CSIH/ポート設定
・LED出力&全消灯設定
MMCドライバ初期化
カード挿抜検出用メモリ初期化
カード検出処理
カード挿入
されたか？
Yes
No
LED2点灯
MMCスロット初期化処理
初期化処理
正常終了？
Yes
No
LED2点灯
LED2消灯、LED3点灯
ファイルシステム作業領域登録処理
ファイルオープン処理(ライトモード)
ファイルオープン
処理正常終了？
No
2
Yes
データ書き込み処理
データ書き込み
処理正常終了？
No
2
Yes
1
1
ファイルクローズ処理
ファイルオープン処理(リードモード)
ファイルオープン
処理正常終了？
No
2
Yes
ファイルポインタ移動処理
データ読み出し処理
データ読み出し
処理正常終了？
No
2
Yes
リード/ライトデータ比較処理
データは同じ？
No
2
Yes
ファイルクローズ処理
LED2･LED3点灯
正常終了
(無限ループ)
2
LED2点灯、LED3消灯
異常終了
(無限ループ)

8.3 関数一覧

No	関数名	概略
1.0	main	MCU のクロックおよび CPU ボードの LED を初期化し、サンプルプログラムを実行します。
1.1	PORT_Init	CSIH ポート、DETECT 端子の初期化をします。
1.2	CSIH1_Init	CSIH モジュールの初期化をします。
1.3	R_init_portpins	LED ポートの初期化をします。
1.4	tfat_sample	TFAT ファイルシステムサンプルプログラムを実行します。
1.4.1	R_mmc_Init_Driver	MMC ドライバ処理の初期化をします。 - API 関数です。
1.4.2	R_init_card_detect_chat	カード挿抜検出用のメモリを初期化します。
1.4.3	R_tfat_disk_initialize	カードの初期化をします。 - API 関数です。
1.4.3.1	R_card_insertion_chk	カードの挿入検出をします。
1.4.3.2	R_mmc_Init_Slot	カード制御の RAM、ポートの初期化、カードの初期化、カードの状態遷移（MMC モードから SPI モード）をします。
1.4.4	R_tfat_disk_read	カードからデータを読み出します。 - API 関数です。
1.4.4.1	R_mmc_Read_Data	カードから 512Byte 単位でデータを読み出します。
1.4.5	R_tfat_disk_write	カードにデータを書き込みます。 - API 関数です。
1.4.5.1	R_mmc_Write_Data	カードに 512Byte 単位でデータを書き込みます。
1.4.6	R_error	異常を検出したときに実行される関数です。

8.4 関数チャート


1.0
main
1.1
PORT_Init
1.2
CSIH1_Init
1.3
R_init_portpins
1.4
tfat_sample
1.4.1
R_mmc_Init_Driver
1.4.2
R_init_card_detect_chat
1.4.3
R_tfat_disk_initialize
1.4.3.1
R_card_insertion_chk
1.4.3.2
R_mmc_Init_Slot
1.4.4
R_tfat_disk_read
1.4.4.1
R_mmc_Read_Data
1.4.5
R_tfat_disk_write
1.4.5.1
R_mmc_Write_Data
1.4.6
R_error

9. 参考ドキュメント

- ハードウェアマニュアル
  V850E2/ML4 ユーザーズマニュアル ハードウェアマニュアル [R01UH0262JJ]
  （最新版はルネサスエレクトロニクスのホームページから入手してください）

- ソフトウェアマニュアル
  V850E2M ユーザーズマニュアル アーキテクチャ編 [R01US0001JJ]
  （最新版はルネサスエレクトロニクスのホームページから入手してください）

- 関連アプリケーションノート
  [1] V850E2M M3S-TFAT-Tiny:FAT ファイルシステムソフトウェア導入ガイド [R01AN1028JJ0100]

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.03.27	—	初版発行
1.01	2012.07.23	—	プロジェクトオプション設定変更、ポート初期化、クロック出力制御に伴う更新

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. **未使用端子の処理**
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. **電源投入時の処置**
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. **リザーブアドレスのアクセス禁止**
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. **クロックについて**
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. **製品間の相違について**
   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認ください。
   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/