

# SD カードI/O モジュール 製品仕様書

Rev.6  
2005/6/3

## 改版履歴

Rev.	日付	改版内容	改版者
1	2004/11/22	新規作成。	One Light 南
2	2004/11/25	1) CPU ボード依存の記述を削除。 2) コネクタ変更。 3) 外形図追加。 4) 回路図の用紙サイズ変更。 5) 部品表追加。	One Light 南
3	2004/11/30	1) 型名追加。	One Light 南
4	2005/2/10	1) 前面修正	One Light 南
5	2005/2/16	1) ピンアサイン変更。	One Light 南
6	2005/6/3	1) ピンアサイン変更。	One Light 南

## 目次

1. 概要 .....	3
2. 特徴 .....	3
3. 電源、消費電流 .....	3
4. 外形図 .....	3
5. ピン配置 .....	4
5.1. CN2 .....	4
5.2. CN3 .....	4
6. SD カード I/O ドライバー .....	5
6.1. 提供ファイル一覧 .....	5
6.2. API 一覧 .....	5
6.3. API 仕様 .....	5
6.3.1. sdInit .....	5
6.3.2. sdDriver .....	6
6.3.3. sdSuspend .....	6
6.3.4. sdRead .....	7
6.3.5. sdSectorRead .....	7
6.3.6. sdWrite .....	8
6.3.7. sdSectorWrite .....	8
6.4. sddrv.h .....	9
6.5. カスタマイズ .....	10
7. お問い合わせ .....	11

## 1. 概要

SD カードI/O モジュール (以降、SDIOM) は、専用 ASIC とその周辺回路、SD カードコネクタで構成されるボードです。

SDIOM は、ホストCPU からの要求に応じて SD カードをセクター単位での読み込み、書き込みを行う機能を持ち、ホストCPU と8 ビットバスで接続されます。

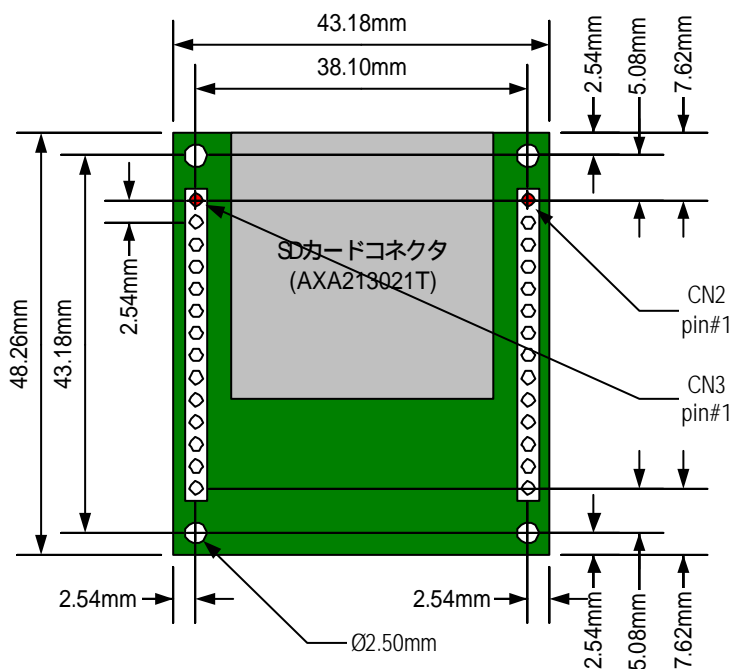
## 2. 特徴

- ? SD カードI/O エンジンとして高性能な専用 ASIC を採用しています。
- ? SD バスモードに対応し、4 ビット幅でのカードI/O を行い、最大約 8M バイト/秒の READ/WRITE 性能を持ちます。
- ? 4GB までの MMC、SD カードに対応しています。
- ? ファイルシステム等のソフトウェア製品と組み合わせることにより、組み込みシステムでのSD カード利用が容易に行え、PC とのデータ交換にも便利です。
- ? SD カードI/O ドライバー標準添付です。

## 3. 電源、消費電流

- ? 3.3V、5.0V の両電源に対応しています。
- ? 消費電流は 200mA 以内です。

## 4. 外形図



## 5. ピン配置

### 5.1. CN2

2.54mm ピッチ 1 列 x14

ピン#	信号名	説明
1	GND	グラウンド
2	GND	グラウンド
3	-RES	リセット(L=リセット)
4	-SUSPEND	省電力モード制御(L=省電力モード)
5	A0	アドレス0
6	A1	アドレス1
7	A2	アドレス2
8	A3	アドレス3
9	A4	アドレス4
10	-CS	チップセレクト
11	-RD	リード
12	-WR	ライト
13	(N.C.)	未接続
14	(N.C.)	未接続

### 5.2. CN3

2.54mm ピッチ 1 列 x14

ピン#	信号名	説明
1	D0	データ0
2	D1	データ1
3	D2	データ2
4	D3	データ3
5	D4	データ4
6	D5	データ5
7	D6	データ6
8	D7	データ7
9	(N.C.)	未接続
10	(N.C.)	未接続
11	VCC	SDIOM 電源 (5V または 3.3V)
12	VCC	SDIOM 電源 (5V または 3.3V)
13	GND	グラウンド
14	GND	グラウンド

## 6. SD カードI/O ドライバー

SD カードI/O ドライバーは、SDIOM を利用するためのソフトウェアです。ソースコードで提供されます。

### 6.1. 提供ファイル一覧

関数名	概要
sddrv.c	SD カードI/O ドライバ本体
sddrv.h	SD カードI/O ドライバヘッダー
sumo.c	SD カードコントローラ制御部
sumo.h	SD カードコントローラ制御部ヘッダー
main.c	デモ用サンプルコード

### 6.2. API 一覧

関数名	概要
sdInit	SD カードI/O ドライバーを初期化します。SDIOM を利用する際に最初に一度だけ呼び出してください。
sdDriver	SD カードの装着状態を調べ、必要に応じて SD カードの初期化を行います。メインループの中で10ミリ秒周期で呼び出してください。
sdRead	指定されたセクターからデータを読み込みます。
sdSectorRead	指定されたセクターからデータを読み込みます (1セクター)。
SdWrite	指定されたセクターへ指定されたデータを書き込みます。
sdSectorWrite	指定されたセクターへ指定されたデータを書き込みます (1セクター)。
sdSuspend	SDIOM を省電力モードへ移行、又は復帰を行います。

### 6.3. API 仕様

#### 6.3.1. sdInit

**【処理内容】** SD カードI/O ドライバーを初期化します。SDIOM を利用する際に最初に一度だけ呼び出してください。SDIOM をリセットする目的で呼び出すことも可能です。

**【関数型】** int sdInit( void \*reg);

**【引数】** reg SD I/O ASIC レジスタの先頭アドレスを指定してください。

**【戻値】** 初期化に成功した場合、SD\_OK を返します。失敗した場合、SD\_OK 以外を返します。

**【使用例】**

```
#include "sddrv.h"
:
/* SDIOM の初期化 */
if( sdInit() != SD_OK){
    /* エラー処理 */
}
```

### 6.3.2. sdDriver

**【処理内容】** SD カードの装着状態を調べ、必要に応じて SD カードの初期化を行います。メインループの中で 10ms 周期で呼び出してください。

**【関数型】** int sdDriver( PSDINFO sdinfo);

**【引数】** sdinfo SD カード状態が格納されます。

**【戻値】** 常に SD\_OK を返します。

**【使用例】**

```
#include "sddrv.h"
:
SDINFO sdinfo; /* SD 状態格納領域 */
:
void main( void)
{
:
/* メインループ */
while( 1){
:
/* SD カードI/O ドライバーの呼び出し */
sdDriver( &sdinfo);
:
}
}
```

### 6.3.3. sdSuspend

**【処理内容】** SDIOM を省電力モードへ移行、又は復帰を行います。

**【関数型】** int sdSuspend( int mode);

**【引数】** mode SD\_SUSPEND を指定するとサスペンドモードへ移行します。  
SD\_RESUME を指定するとサスペンドモードから復帰します。

**【戻値】** 常に SD\_OK を返します。

**【使用例】**

```
#include "sddrv.h"
:
/* サスペンドモードへ移行 */
sdSuspend( SD_SUSPEND);
:
/* サスペンドモードから復帰 */
sdSuspend( SD_RESUME);
:
```

### 6.3.4. sdRead

- 【処理内容】** 指定されたセクターからデータを読み込みます。
- 【関数型】** int sdRead( void \*buf, int len, unsigned long lba);
- 【引数】** buf SD カードから読み取ったデータの格納先を指定します。  
len SD カードから読み取るセクター数を指定します。  
1セクターは512バイトです。  
lba SD カードのセクターアドレスを指定します。  
セクターアドレスは0から始まります。
- 【戻値】** 正常に読み取りした場合 SD\_OK を返します。その他の場合は SD\_OK 以外を返します。
- 【使用例】**
- ```
#include "sddrv.h"
:
/* SD カード読み取り */
{
    unsigned char buf[512*2];
    if( sdRead( buf, 2, 0) != SD_OK){
        /* エラー処理 */
    }
}
```

### 6.3.5. sdSectorRead

- 【処理内容】** 指定されたセクターから1セクター分のデータを読み込みます。
- 【関数型】** int sdSectorRead( void \*buf, unsigned long lba);
- 【引数】** buf SD カードから読み取ったデータの格納先を指定します。  
1セクターは512バイトです。  
lba SD カードのセクターアドレスを指定します。  
セクターアドレスは0から始まります。
- 【戻値】** 正常に読み取りした場合 SD\_OK を返します。その他の場合は SD\_OK 以外を返します。
- 【使用例】**
- ```
#include "sddrv.h"
:
/* SD カード読み取り */
{
    unsigned char buf[512];
    if( sdSectorRead( buf, 0) != SD_OK){
        /* エラー処理 */
    }
}
```

### 6.3.6. sdWrite

- 【処理内容】** 指定されたセクターからデータを読み込みます。
- 【関数型】** int sdWrite( void \*buf, int len, unsigned long lba);
- 【引数】** buf SD カードへ書き込むデータを指定します。  
len SD カードへ書き込むセクター数を指定します。  
1セクターは512バイトです。  
lba SD カードの書き込み先 (セクターアドレス)を指定します。  
セクターアドレスは0から始まります。
- 【戻値】** 正常に書き込みした場合 SD\_OK を返します。その他の場合は SD\_OK 以外を返します。
- 【使用例】** #include "sddrv.h"  
:  
/\* SD カード書き込み \*/  
{  
    unsigned char buf[512\*2];  
    memset( buf, 0x55, 512\*2);  
    if( sdWrite( buf, 2, 0) != SD\_OK){  
        /\* エラー処理 \*/  
    }  
}

### 6.3.7. sdSectorWrite

- 【処理内容】** 指定されたセクターから1セクター分のデータを読み込みます。
- 【関数型】** int sdWrite( void \*buf, int len, unsigned long lba);
- 【引数】** buf SD カードへ書き込むデータを指定します。  
1セクターは512バイトです。  
lba SD カードの書き込み先 (セクターアドレス)を指定します。  
セクターアドレスは0から始まります。
- 【戻値】** 正常に書き込みした場合 SD\_OK を返します。その他の場合は SD\_OK 以外を返します。
- 【使用例】** #include "sddrv.h"  
:  
/\* SD カード書き込み \*/  
{  
    unsigned char buf[512];  
    memset( buf, 0x55, 512);  
    if( sdSectorWrite( buf, 0) != SD\_OK){  
        /\* エラー処理 \*/  
    }  
}

## 6.4. sddrv.h

sddrv.h は、SD カードI/O ドライバーを利用するためのヘッダーファイルです。SD カードI/O ドライバーを利用するプログラムよりインクルードしてください。

```
//SD 情報
typedef struct{
    unsigned ready      :1;           //カード利用可能
    unsigned wp         :1;           //書き込み禁止(1=禁止)
    unsigned wide       :1;           //現在のバス幅(1=4 ビット)
    unsigned mmc        :1;           //MMC 初期化中
    unsigned invalid    :1;           //不正カード(初期化不能)
    unsigned suspend    :1;           //サスペンド中
    unsigned long capacity;           //カード容量 (セクター数)
    unsigned long last_lba;           //最終 LBA

    unsigned char cid[16];            //CID
    unsigned char csd[16];            //CSD
    unsigned char scr[8];             //SCR(SD カードのみ)
    unsigned long rca;                //RCA
    unsigned char freq;               //CLK 周波数コード
}SDINFO, *PSDINFO;

//SD ドライバ API 戻り値
#define SD_OK             (0)         //正常終了
#define SD_ERR           (-1)        //異常終了
#define SD_OUT_OF_RANGE  (-2)        //アドレス範囲外
#define SD_WRITE_PROTECTED (-3)     //書き込み禁止状態
#define SD_SUSPENDED     (-4)        //サスペンド中
#define SD_NO_CARD       (-5)        //カードなし

//SD ドライバ API
int sdInit( void);
int sdDriver( PSDINFO sdinfo);
int sdRead( void *buf, int len, unsigned long lba);
int sdSectorRead( void *buf, unsigned long lba);
int sdWrite( void *buf, int len, unsigned long lba);
int sdSectorWrite( void *buf, unsigned long lba);
int sdSuspend( int suspend);
```

## 6.5. カスタマイズ

ハードウェア依存部分のカスタマイズは、sumo.h を修正することにより行います。以下、sumo.h の抜粋です。

```
#define SUMO_BASE      0xC00000L    //SD カードコントローラ IC レジスタアドレス
#define SUMO_CLK       40          //CLK クロック周波数(40 か 50)
#define SD_WIDE_BUS    1          //ワイドバス(4-bit モード)を使用する
#define SUMO_SRAM_TEST 1          //SD カードコントローラ IC 内蔵 SRAM のテスト
#define sdclnitSuspend() (*(BR*)(0xfffe3e)) |= 0x01 //PF0 方向
#define sdcLeaveSuspend() (*(BR*)(0xffff0e)) |= 0x01 //PF0 = 1
#define sdcEnterSuspend() (*(BR*)(0xffff0e)) &= ~0x01 //PF0 = 0
:
:
```

ラベル	意味
SUMO_BASE	SD カードコントローラIC に割り当てられた CPU アドレスを指定してください。
SUMO_CLK	SD カードコントローラIC の動作周波数を設定してください。 40MHz の場合は 40、50MHZ の場合は 50 を指定してください。
SD_WIDE_BUS	ワイドバスを使用するかどうかを指定します。 1 を指定すると、ワイドバスを使用します。0 を指定するとワイドバスを使用しません。
SUMO_SRAM_TEST	1 を指定すると、sdlnit 内で SD カードコントローラIC 内蔵の SRAM をテストします。0 を指定するとテストを行いません。
SdclnitSuspend()	CPU のサスペンド制御端子を出力モードに設定するマクロを記述してください。
sdcLeaveSuspend()	CPU のサスペンド制御端子へ HIGH 出力するマクロを記述してください。
sdcEnterSuspend()	CPU のサスペンド制御端子へ LOW 出力するマクロを記述してください。

## 7. お問い合わせ

SD カードI/O モジュール、SD カードI/O ASIC についてのお問い合わせは有限会社ワンライトへ。

有限会社ワンライト (<http://www.one-light.jp>)  
東京都町田市南成瀬 6-5-21 ブランシュJ-1  
Tel: 042-710-4433 Fax: 042-710-4713  
カスタム仕様、応用製品の開発協力承ります。