

AVR107 : 直列型メモリとAVRのインターフェース

要点

- デバイス: AT25128A/256A, AT25F1024/2048/4096
- 直列型メモリの全機能支援
- メモリ配列集中読み込み
- ページ集中読み込み
- 書き込み保護検出
- アクセス中検出
- 進行を妨げない書き込みアクセス
- アクセス状態情報

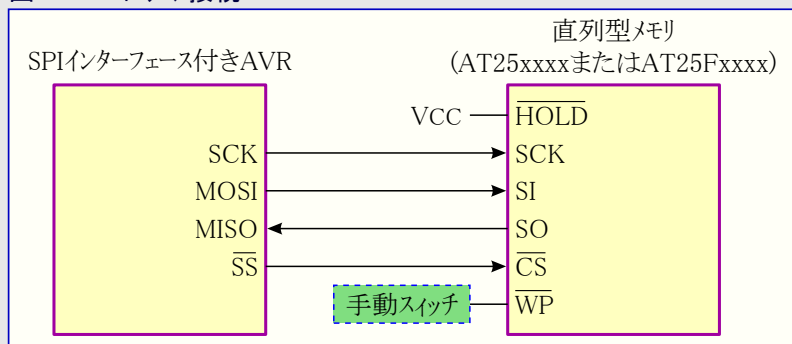
1. 序説

直列インターフェースメモリは消費者、車載、電気通信、医療、工業、PCに関連する市場の広範囲で使用されます。主に個人的な選択データや形態設定/構成設定データの格納に使用される直列メモリは今日利用可能な不揮発性メモリ(NVM)の最も柔軟な形式です。他のNVM解決策に比べて直列メモリデバイスは低電力消費だけでなく、少ないピン数、より小さな外圍器、より低い電圧も提供します。

AVRの殆どはAT25128A/256AやAT25F1024/2048/4096のような直列メモリとの接続が可能なSPIインターフェースを提供します。

お客様のAVRシステムでのSPI直列メモリの統合を簡単化して進めるため、これらを効率的にアクセスするための基本ドライバが開発されました。この応用記述は選ばれた解決策への誘導だけでなく、これらのドライバの機能と構造も記述します。提供する関数の完全な一覧と説明はDoxygen自動化資料の役割です。

図1-1. ハードウェア接続



2. 直列型メモリ

後続項はAVRへのメモリ接続、メモリアクセス形式、領域書き込み保護ビット、多忙検出を記述します。より多くの情報についてはデータシートを参照してください。

2.1. 直列型メモリのAVRへのハードウェア接続

SPIインターフェースは主装置動作に形態設定されます。SCKとMOSIのポートは出力、MISOのポートは入力です。図1-1.をご覧ください。

この応用では直列メモリのチップ選択(\overline{CS})ピンを制御するのに従装置選択(\overline{SS})信号を使用することに注意してください。これはSPIインターフェースの初期化中にいくつかの予防処置を取ることが必要です(「SPIインターフェースの初期化」をご覧ください)。チップ選択線を制御するのにPB2のような別のポートを使用することもできます。

SPIインターフェースはデータが有効でない場合にクロックを常に停止するので、 \overline{HOLD} 信号をHighに維持することができます。

使用者は書き込み保護(\overline{WP})信号の形態設定を選ぶことができます。

注: EEPROMの動作電圧範囲は1.8~5.5Vで、一方フラッシュメモリは2.7~3.6Vで動作します。



8ビット **AVR**[®]
マイクロコントローラ

応用記述

本書は一般の方々の便宜のため有志により作成されたもので、ATMEL社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

Rev. 2595A-03/05, 2595AJ2-02/14

2.2. SPI直列型メモリのアクセス形式

提唱ドライブの構造を理解するために、直列メモリへの各種アクセスを復習することは価値あることです。

全てのアクセスは次の手順に従います。

1. チップ選択線がLowに駆動されます。
2. SCKクロックに同期してデータ線上で幾らかの直列化されたバイトが送られるか、または受け取られます。
3. チップ選択線がHighに駆動されます。

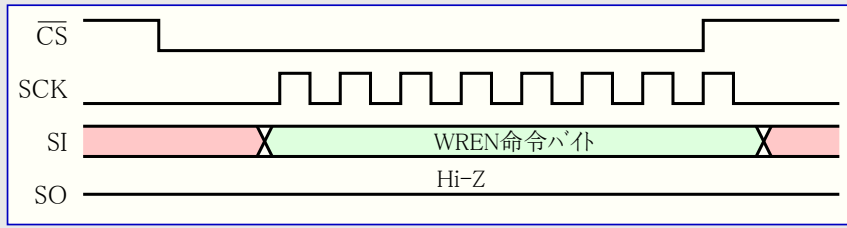
直列化されるバイトの数と値はアクセス形式に依存します。

この応用記述のSPIメモリデバイスはSPI動作種別0('00')とSPI動作種別3('11')の両方を支援します。提唱ドライブでは動作種別0が選ばれます(詳細については直列型メモリのデータシートをご覧ください)。

2.2.1. 単一バイト書き込み命令: WREN, WRDI, CHIP ERASE

これらのアクセスは1バイト長です。MOSIデータ線に命令バイト(以降は"op_code"と呼ばれます(訳注)本書では不使用)だけが送られます。

図2-1. WRENタイミング例



2.2.2. 状態レジスタ読み/書き: RDSR/WRSR

これらのアクセスは2バイト長です。命令バイトに書かれる(WRSR)または読まれる(RDSR)バイトが続きます。

注: 状態レジスタ書き込み操作はWREN命令が先行しなければなりません。

図2-2. WRSRタイミング

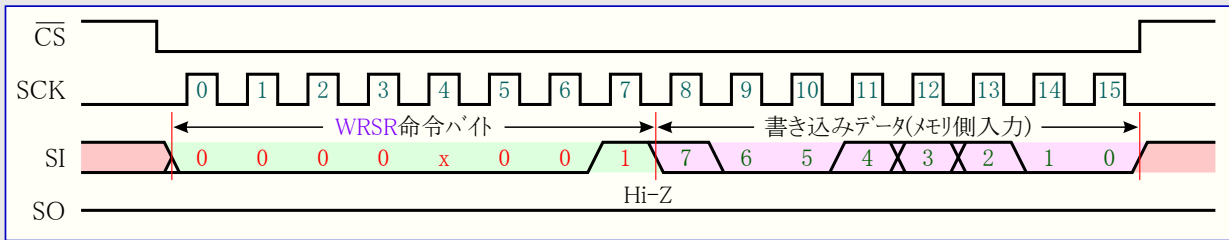
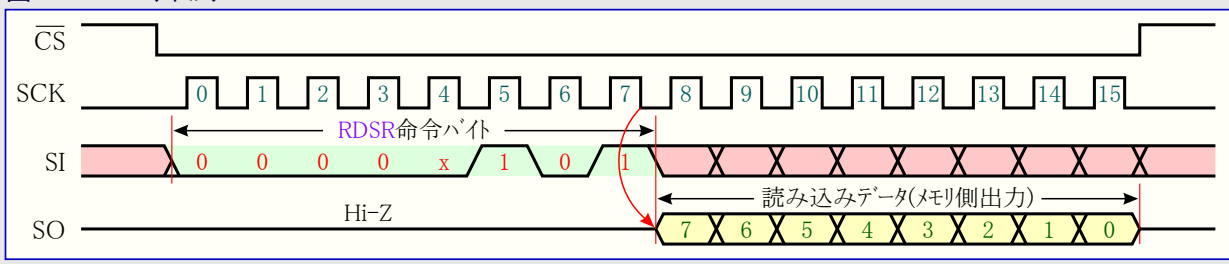


図2-3. RDSRタイミング

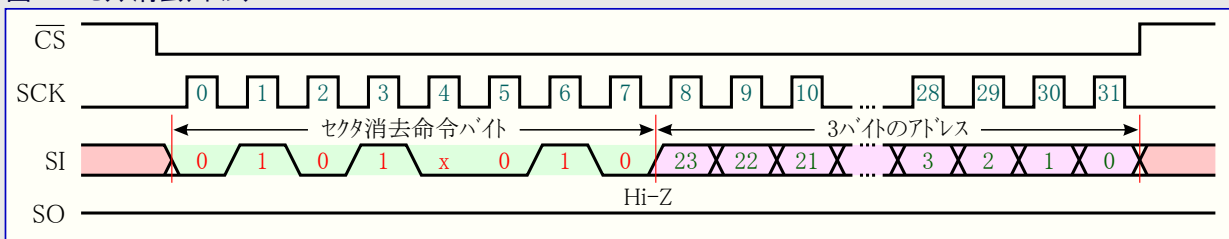


2.2.3. セクタ消去命令(AT25Fxxxxデバイスのみ)

このアクセスは1バイトの命令バイトと3つのアドレスバイトから成ります。

注: セクタ消去(SECTOR ERASE)操作はWREN命令が先行しなければなりません。

図2-4. セクタ消去タイミング



2.2.4. データ書き込みとデータ読み込みのアクセス

これらのアクセスは1バイトの命令バイトと後続するアドレス段階(AT25xxxxデバイスは2バイト長、AT25Fxxxxデバイスは3バイト長)、データ段階から成ります。データ段階の長さは読み書きされるべきバイト数に依存します。

全メモリ配列読み込みの場合、バイト数はメモリのバイト単位での容量です。

注: WRITEとPROGRAMの操作はWREN命令が先行しなければなりません。

図2-5. WRITE命令タイミング (AT25xxxx)

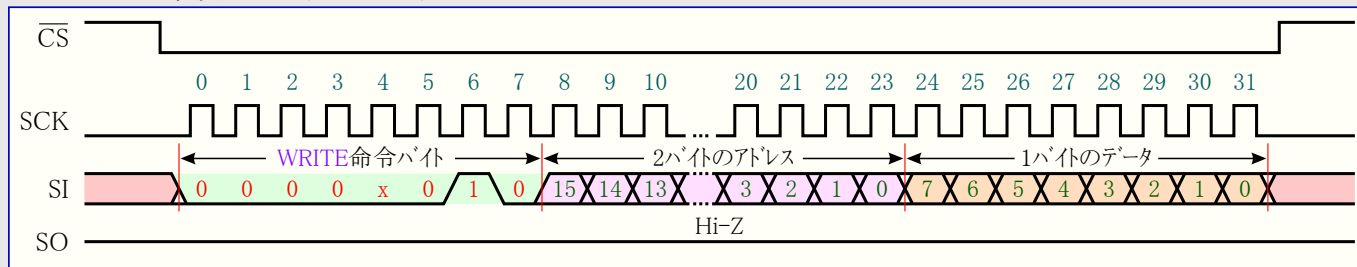
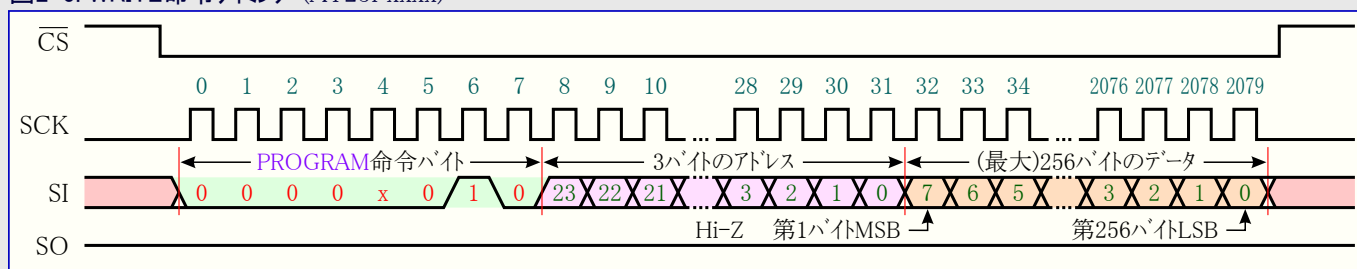


図2-6. WRITE命令タイミング (AT25Fxxxx)



2.2.5. アクセス形式要約

次表はアクセス形式とそれらに関連する長さを要約します。

表2-1. アクセス形式要約表

命令	命令バイト	アドレス段階の長さ (バイト)		データ段階の長さ (バイト)	
		AT25xxxx	AT25Fxxxx	MOSI線	MISO線
WREN	0000 x110	0	0	0	0
WRDI	0000 x100	0	0	0	0
RDSR	0000 x101	0	0	0	1
WRSR	0000 x001	0	0	1	0
READ	0000 x011	2	3	0	1~配列容量
WRITE	0000 x010	2	N/A (注)	1~ページ容量	0
PROGRAM		N/A (注)	3	1~ページ容量	1
SECTOR ERASE	0101 x110	N/A (注)	3	0	0
CHIP ERASE	0110 x110	N/A (注)	0	0	1
RDID	0001 x101	N/A (注)	0	0	2

注: この命令はこのデバイスに対して利用不可です。

2.3. SPI周辺機能の扱い

直列メモリへのアクセスを実行するにはSPIインターフェースを使用し、それが許可され、SCKに同期してSPDRレジスタに書かれたバイトをMOSI線にシフト出力し、そしてMISO線上のバイトをSPDRにシフト入力します。

SPI操作はバイト単位です。バイトが送受信された後、SPSRレジスタ内でSPIFが設定(1)されます。

SPI周辺機能を扱うための2つの操作方法が想像できます。

1. SPIFが設定(1)されたかを検知するためにSPSRをポーリングし、そして次のSPI転送を実行します。
2. SPI割り込みを許可し、SPI割り込み処理で次のSPI転送を管理します。

2.3.1. ホーリング動作形態

ホーリング動作は転送終了に対する素早い応答の利点があり、簡潔なコード量にも通じます。

決定的な欠点はコードがホーリング繰り返しの終了を待たなければならないので、ホーリング操作が閉塞構造であることです。従って次の短いアクセスに対してホーリング操作を選択します。

- WREN
- WRDI
- WRSR
- RDSR
- SECTOR ERASE
- CHIP ERASE
- RDID

主プログラムがそのコード実行を進めるのに先立ってデータを読むために待つと仮定したため、次にもホーリング操作を選択しました。

- READ

2.3.2. 割り込み動作形態

データ集中書き込みアクセスに対してホーリング動作形態は上手く適応しません。実際、一般的に主ルーチンはその実行コードで先に進む前に直列メモリ書き込み周期の終了を待つ必要がありません。

従って以下のアクセスを扱うための操作に割り込み動作を選びます。

- WRITE
- PROGRAM

この動作形態ではSPI割り込み処理が次のSPI転送を管理します。これはSPI割り込み間でそのコード実行を主ルーチンに許します。割り込みルーチンが次のSPI転送を扱うことができるように、全域変数にアクセス状況を保存しなければなりません(「[全域変数の説明](#)」をご覧ください)。これは数バイトのメモリ空間を消費します。

2.3.3. 両動作形態間の相互作用

ホーリング動作形態を使用する関数と割り込み動作を使用する関数間でSPIインターフェースでのどんな衝突も避けるため、以下の予防処置が取られなければなりません。

1. どんなホーリング操作の前にも、SPI周辺機能が送信準備可であることを(全域変数の状態を通して)調べられます。そしてSPI割り込みが禁止されます。
2. ホーリング操作形態を使用するアクセスの終了で、SPI割り込みが許可されます。

2.4. 書き込み保護された領域の検出

直列メモリデバイスはコードを書く時にも考慮されなければならない書き込み保護機能を提供します。

保護されるセクタは直列メモリの状態レジスタのBPxビットの設定に依存します。より多くの詳細については直列メモリのデータシートを参照してください。

表2-2. AT25F4096メモリ保護任意選択

段階	状態レジスタ			AT25F4096	
	BP2	BP1	BP0	施錠アドレス	施錠セクタ
0 (なし)	0	0	0	なし	なし
1 (1/8)	0	0	1	\$070000~\$07FFFF	セクタ8
2 (1/4)	0	1	0	\$060000~\$07FFFF	セクタ7,8
3 (1/2)	0	0	1	\$040000~\$07FFFF	セクタ5~8
4 (全体)	1	x	x	\$000000~\$07FFFF	全セクタ(1~8)

注: x=無関係

表2-3. AT25128A/AT25256Aメモリ保護任意選択

段階	状態レジスタ		保護アドレス	
	BP1	BP0	AT25128A	AT25256A
0 (なし)	0	0	なし	なし
1 (1/8)	0	1	\$3000~\$3FFF	\$6000~\$6FFF
2 (1/4)	1	0	\$2000~\$3FFF	\$4000~\$6FFF
3 (1/2)	0	1	\$0000~\$3FFF	\$0000~\$6FFF

提唱ドライブは書き込み保護領域の定義と活性化(有効)化を使用者に許すSetWriteProtectedArea関数を含みます。

状態レジスタに書いて読み戻すことによって直列メモリがハードウェア書き込み保護されている(\overline{WP} 信号がLow)かをSetWriteProtectedArea関数が検知することに留意してください。ハードウェア書き込み保護の場合、書き込み保護形態の設定も活性化のどちらも実行することができません。

データ書き込みアクセス(WRITE/PROGRAM命令)により、直列メモリの状態レジスタが読まれます。BPxビットの値と書かれるべき最初のバイトのアドレス位置に従って、書き込み関数はその目的アドレスが書き込み保護領域に置かれているかを検知します。

そうならば、返されるアクセス状態符号はDATA_WR_PROTECTED値に設定され、そのアクセスは中止されます。

AT25Fデバイスに関してはSECTOR ERASEまたはCHIP ERASEの操作を実行する時にこの書き込み保護検出が活性化(有効)です。

2.5. 多忙検出

直列メモリは状態レジスタ内の $\overline{\text{RDY}}$ ビットで現在の状態を示します。

このビットが'1'に設定されている場合、 RDSR アクセスを除く全ての操作が禁止されます。

従って各関数は $\overline{\text{RDY}}$ ビットが設定(1)されているかを確認します。そうならば、返される状態符号は BUSY 値に設定され、そのアクセスは中止されます。

注: メモリ書き込み循環操作の間は状態レジスタの全ビットが'1'に設定されるため、状態レジスタの情報ビットを読んで評価するのに先立ってメモリは準備可でなければなりません。

3. コード実装

以降の項はSPIインターフェース、ポーリング動作関数、割り込み動作関数の初期化を記述します。

3.1. SPIインターフェースの初期化

- SPIインターフェースは主装置として形態設定されなければなりません。SPIが従装置選択線を通して従装置に切り替えられないことを保証するために、 $\overline{\text{SS}}$ (PB2)ポートはHighレベルの出力として形態設定されなければなりません。
 - PB2ポートを形態設定するためにSPIインターフェースが最初に禁止されます。
 - PB2ポートは出力として形態設定されます。
 - PB2ポートにHigh値が設定されます。
 - MSTR ビットが設定(1)されます。
 - SPIが許可されます。
- 特にチップ選択の上昇端に於いてクロック信号がLowであることを保証するために、SPI動作種別0が選択されなければなりません。最上位ビットが最初に送出されるビットでなければなりません。データ速度の選択は使用者定義です。
- 他のポートの方向は次のように形態設定されなければなりません。
 - 出力：PB5/SCK, PB3/MOSI
 - 入力：PB4/MISO
- SPSR と SPDR のレジスタを読むことによって SPIF フラグが解除(0)されなければなりません。
- SPI割り込みが許可されます。

3.2. ポーリング動作関数

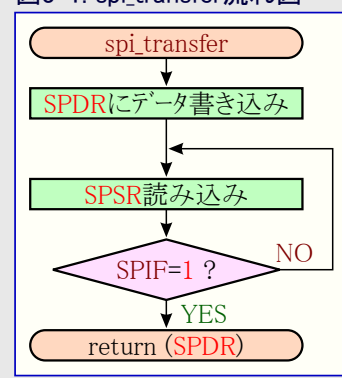
WREN , WRDI , WRSR , RDSR , SECTOR ERASE , CHIP ERASE , RDID , READ の全ては spi_transfer 基本関数を呼び出します。

3.2.1. SPI転送基本関数

spi_transfer 関数はポーリング動作でSPIインターフェースを扱う基本の関数です。これはバイトの送出と受信されたバイトを格納します。この関数がチップ選択線のレベルを制御しないことに注意してください。

原型: `char spi_transfer(char data)`

図3-1. spi_transfer流れ図



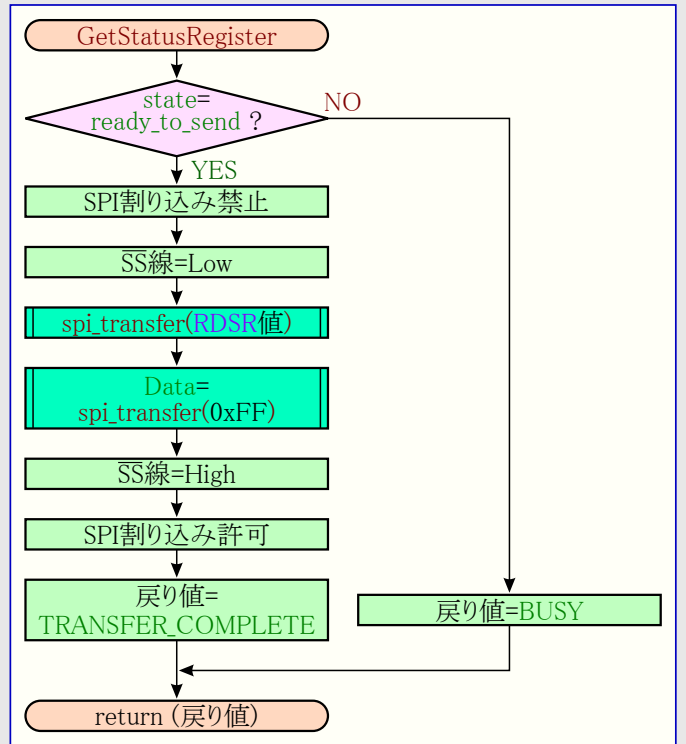
3.2.2. 状態レジスタ取得関数

この関数はRDSRアクセスを実行します。これはアクセス状態符号を返します。状態レジスタの値は引数ポインタの変数内に置かれます。

最初に書き込み処理によってSPIが閉ざされていないことを調べます(データ書き込みアクセス項目をご覧ください)。

原型: `char GetStatusRegister(char *data)`

図3-2. GetStatusRegister流れ図

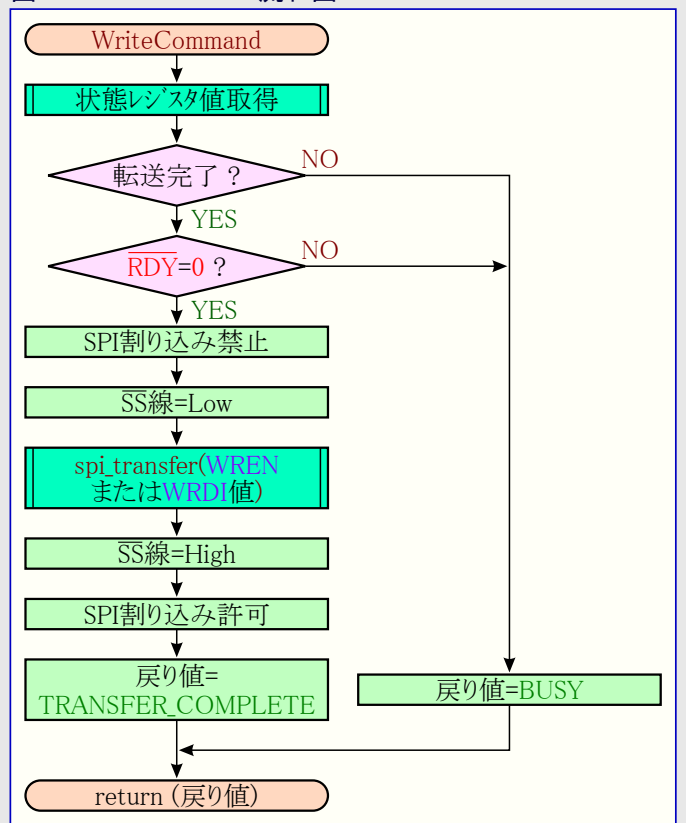


3.2.3. 単一バイト書き込み命令

WRENとWRDIのアクセスはバイト長で、WriteCommand関数によって実行することができます。このアクセスを実行する前に直列メモリは準備可でなければなりません。これはアクセス状態符号を返します。

原型: `char WriteCommand(char op_code)`

図3-3. WriteCommand流れ図

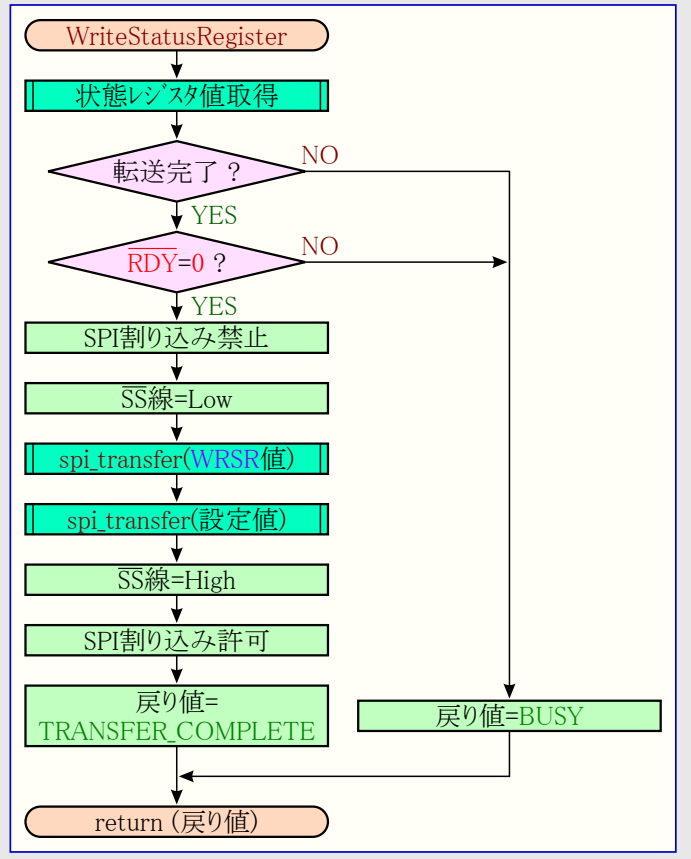


3.2.4. 状態レジスタ書き込み関数

この関数はRDSRアクセスを実行します。これは直列メモリが書き込み準備可であることを調べます。WriteCommand関数と同じアクセス状態符号を返します。

原型: char WriteStatusReg(char sr)

図3-4. WriteStatusRegister流れ図



3.2.5. データ列取得関数

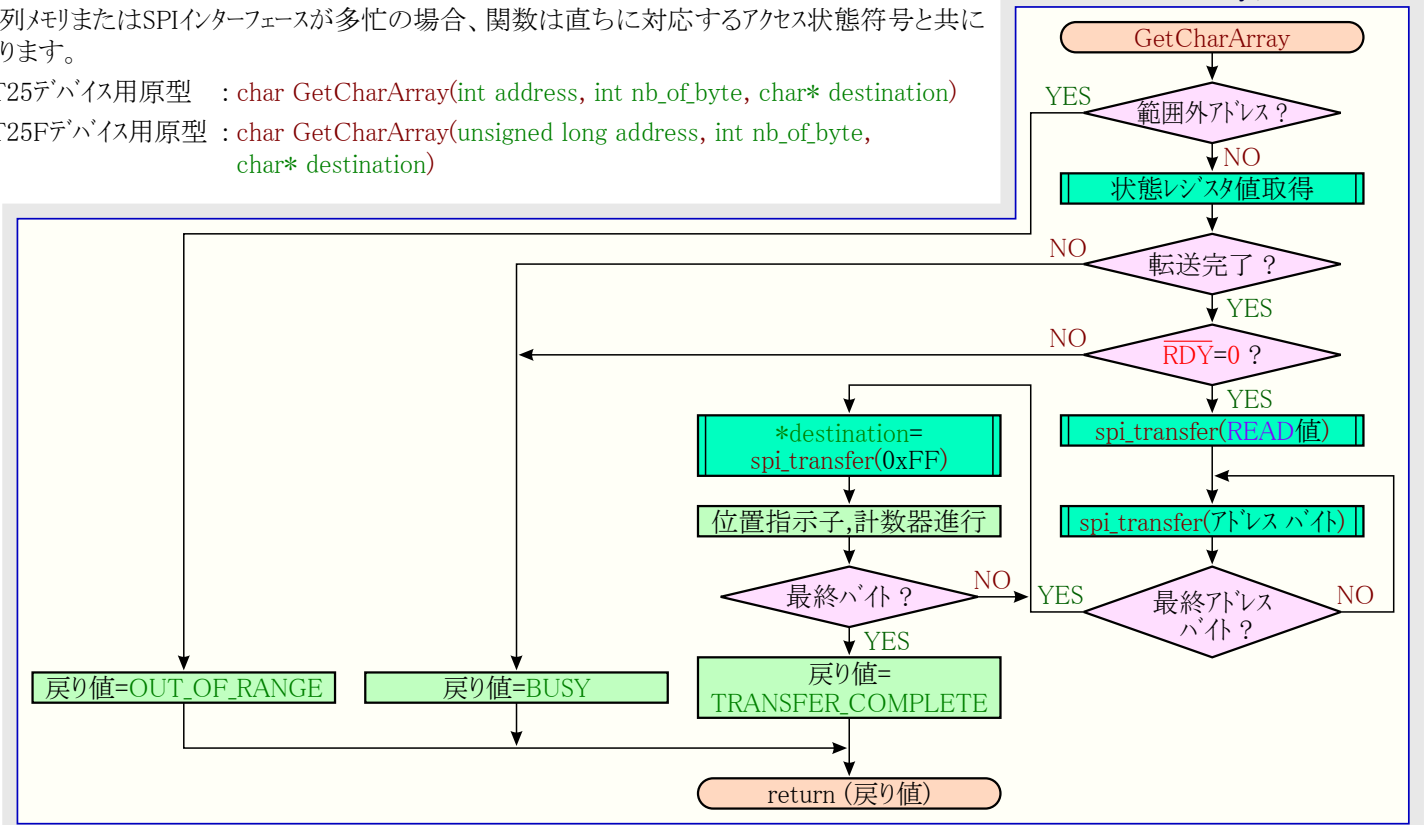
この関数は直列メモリから1または複数のバイトを読みます。

直列メモリまたはSPIインターフェースが多忙の場合、関数は直ちに対応するアクセス状態符号と共に返ります。

AT25デバイス用原型 : char GetCharArray(int address, int nb_of_byte, char* destination)

AT25Fデバイス用原型 : char GetCharArray(unsigned long address, int nb_of_byte, char* destination)

図3-5. GetCharArray流れ図



3.2.6. データ取得関数

コード量最小化のため、GetChar関数は引数として1バイトのみでGetCharArray関数を呼びます。

AT25デバイス用原型 : `char GetChar(int address, char *data)`

AT25Fデバイス用原型 : `char GetChar(unsigned long address, char *data)`

3.3. 割り込み動作関数

3.3.1. 原理

直列メモリへの書き込みアクセスは読み込みアクセスのようにSPI転送手順で構成されます。しかし読み込みアクセスとは異なり、SPI転送の終了はSPI割り込みによって決められます。書き込みアクセスを主ルーチン実行に対して全てを背面で実行させるため、SPI割り込み処理はチップ選択線を制御するだけでなく、次のSPI転送の準備もできなければなりません。

このような機構を可能とするため、割り込み処理は現在の状況を知らなければなりません。この状況は全域変数に保存されます。

状態全域変数は現在のSPI転送形式(命令(INSTRUCTION)、アドレス(ADDRESS)、データ(DATA))についての情報を含みます。

SPI転送の終了を意味するSPI割り込みが起されると、割り込み処理はどのSPI転送の形式かを判断し、何かあるならそれを開始しなければなりません。割り込み処理はSPI割り込み事象で動かす有限状態機構として働きます。

3.3.2. 全域変数の説明

状態全域変数は書き込みアクセスの段階を記載します(より多くの詳細については[有限状態機構構成図](#)をご覧ください)。

INSTRUCTION : 命令バイトが転送されつつあります。

ADDRESS : 現在のアドレスバイトが転送されつつあります。

DATA : 現在のデータバイトが転送されつつあります。

nb_byte 全域変数は書き込むべきデータバイト総数-1を含みます。

byte_cnt 全域変数は既に書かれたアドレスまたはデータのバイト数を含みます。

address 全域変数は先頭バイトが書かれなければならない直列メモリのアドレスを含みます。

data_ptr 全域変数は先頭バイトが配置されたSRAMのアドレスを含みます。

3.3.3. データ列書き込み関数

この関数はバイト配列を直列メモリ内へ書きます。配列の容量は直列メモリのページ容量に上げることができます。引数の源はこの配列へのポインタです。

関数はアドレスが範囲外か、または書き込み保護領域に合致するかを調べます。加えて関数は直列メモリの多忙状態の検出を提供するアクセス状態を返します。

注: 書き込みの流れがページ境界を横切る場合、巻き丸めが起きます。

更なる詳細についてはDoxygen資料を参照してください。

AT25デバイス用原型 : `char PutCharArray(int address, char nb_of_byte, char* source)`

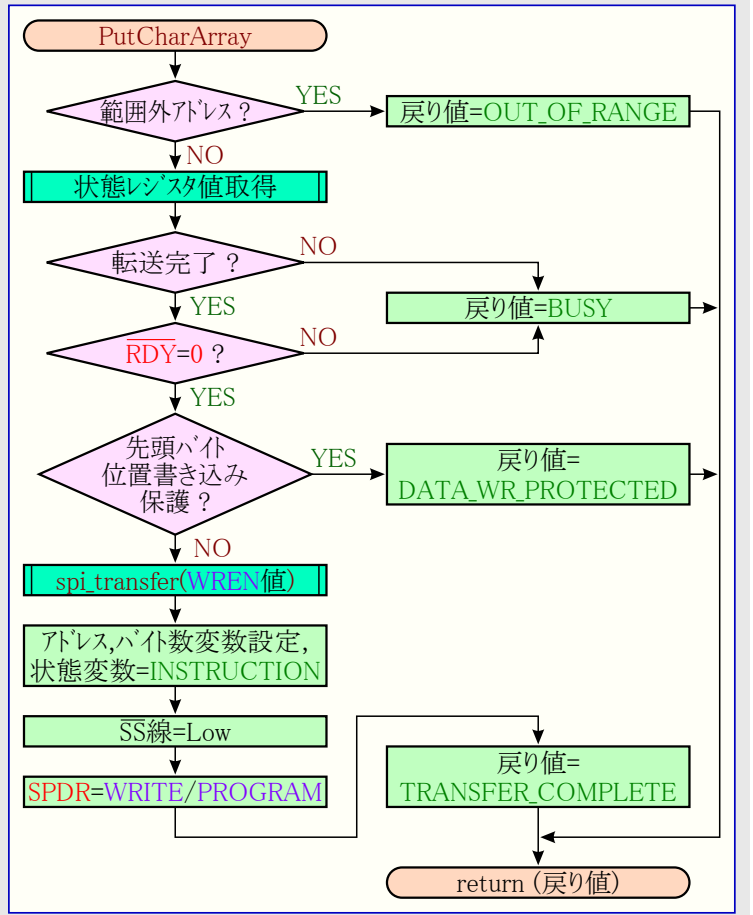
AT25Fデバイス用原型 : `char PutCharArray(unsigned long address, char nb_of_byte, char* source)`

注: 引数nb_of_byteはバイト数-1で、故に例えページが256バイト値でも、byte_cnt全域変数は1バイトだけを専有します。

- AT25Fデバイスに関して、関係するセクタはPutCharArray操作に先立って消去されていなければなりません。

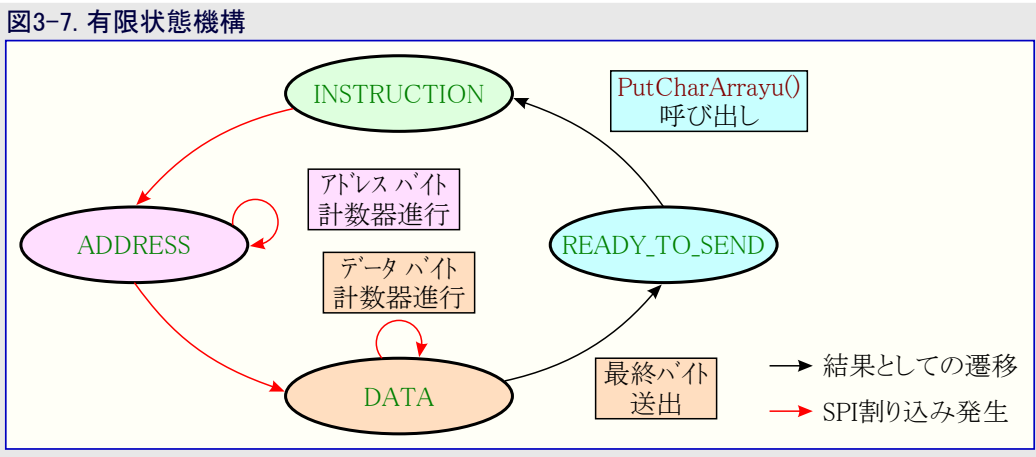
- チップとセクタ(AT25Fxxxxデバイスに関してのみ)の消去関数はデータ段階が存在しないことを除き、GetCharArrayと同じ構造とアクセス状態符号を持ちます。関数は消去されるべき位置が書き込み保護されていないことを照査します。

図3-6. PutCharArray流れ図



3.4. 有限状態機構

以下の構成図は書き込み循環中の状態全域変数遷移を記述します。SPI割り込み処理に移行する度に状態が評価され、次のSPI転送が実行されます。最終バイトが転送された場合、状態はREADY_TO_SENDと呼ぶアイドル状態になります。



3.4.1. 命令(INSTRUCTION)状態

INSTRUCTIONに設定された状態変数での割り込み処理移行は、WRITE(またはAT25Fxxxxフラッシュメモリ型に関してはPROGRAM)命令符号がSPIインターフェースを伝って送出されたことを意味します。

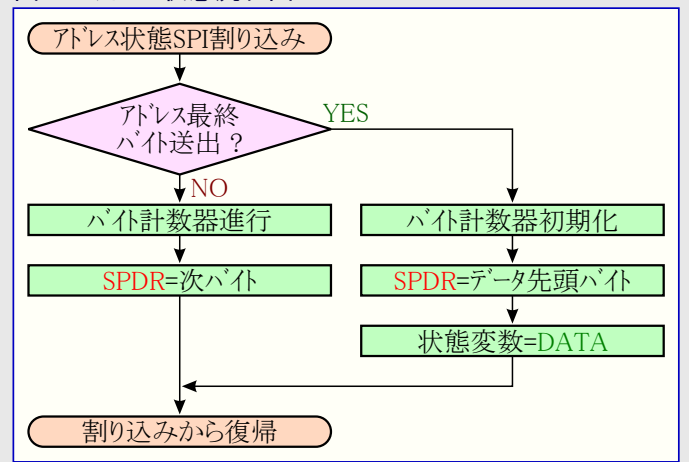
次の状態のADDRESSになり、バイト計数器が1に初期化されます。

アドレスの先頭バイトがSPDRレジスタに書かれます。

3.4.2. アドレス(ADDRESS)状態

ADDRESSに設定された状態変数での割り込み処理移行は、アドレスの1バイトがSPIインターフェースを伝って送出されたことを意味します。

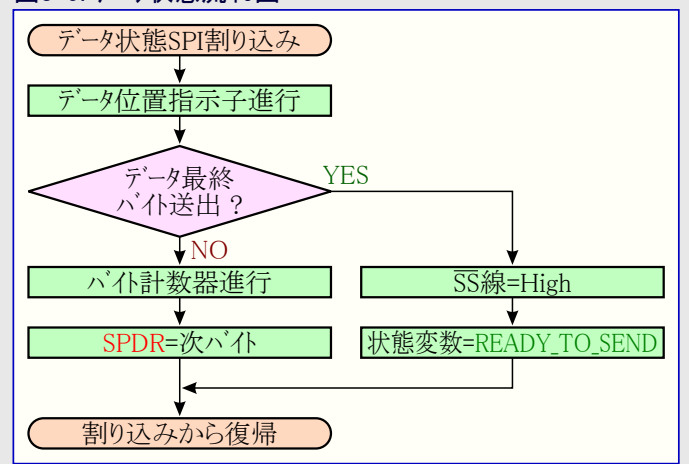
図3-8. アドレス状態流れ図



3.4.3. データ(DATA)状態

DATAに設定された状態変数での割り込み処理移行は、データの1バイトがSPIインターフェースを伝って送出されたことを意味します。

図3-9. データ状態流れ図



3.4.4. 送出準備可(READY_TO_SEND)状態

この状態はSPI割り込み処理に移行している時に活性(有効)になり得ません。この状態は現在SPIインターフェースを使用するデータ書き込みアクセスが全くないことを示します。

従ってSPI書き込み転送の終了を判断するために状態全域変数を検査することができます。

注: 直列メモリが未だ多忙状態中にこの状態値がREADY_TO_SENDになり得ます。

3.5. PutChar関数 (AT25128A/256Aデバイス専用)

直列型EEPROMデバイスはセクタで構成されておらず、書き込みアクセスに先行する\$FF様式(消去)を含む必要がありません。従って読み-変更-書き操作を実装する必要がなく、PutChar関数は引数として1バイトを渡してPutCharArray関数を呼ぶことで構築できます。

注: AT25F直列型フラッシュメモリに関しては、PutChar関数の通常の機能を保証するために、セクタ全体に渡って読み-変更-書き操作が必要とされます。このような操作が時間とメモリ空間を大きく消費するため、直列型フラッシュメモリ用のこの関数は提供しません。

4. AT25128A/256Aデバイス: ドライバ使用例

この例は進行を妨げないPutCharArray関数からの恩恵を示します。

```
main()
{
    int start_address = 0x100;           // 直列メモリ書き込み開始アドレス
    char* src = malloc(PAGE_SIZE* sizeof(char)); // 書き込みデータ列開始アドレス
    char* dest = malloc(PAGE_SIZE * sizeof(char)); // 読み込みデータ列開始アドレス
    char AccessStatus = BUSY;           // 直列メモリ状態符号(初期値=多忙)

    ////////////////////////////////////////////////////////////////////
    // SPI周辺機能初期化
    ////////////////////////////////////////////////////////////////////
    init_spi_master();

    // ソフトウェア書き込み保護禁止
    do {AccessStatus = SetWriteProtectedArea(NONE);} // ソフトウェア書き込み保護なし設定
    while (AccessStatus == BUSY); // 直列メモリ命令完了待機
    if (AccessStatus == HW_PROTECTED) // ハードウェア書き込み保護(/WP=L)ならば、
    {
        // ここでハードウェア書き込み保護検出を処理してください。
    }

    // ページ書き込み
    do {AccessStatus = PutCharArray(start_address, (PAGE_SIZE-1), src);}
    while (AccessStatus == BUSY); // 直列メモリ命令完了待機
    // 任意: 異常処理
    if (AccessStatus == OUT_OF_RANGE) // アドレス範囲外ならば、
    {}
    else if (AccessStatus == DATA_WR_PROTECTED) // データ保護(BPx>0)ならば、
    {}

    ////////////////////////////////////////////////////////////////////
    // 使用者コードはここです。(SPI割り込み処理で書き込みアクセスが実行されつつあります。)
    ////////////////////////////////////////////////////////////////////

    // 任意: SPIメモリでの書き込み禁止(メモリ準備可待機)
    while (WriteCommand(WRDI) != TRANSFER_COMPLETED) {}

    ////////////////////////////////////////////////////////////////////
    // 追加の使用者コードはここです。
    ////////////////////////////////////////////////////////////////////

    // ページ読み込み
    while (GetCharArray(start_address, (PAGE_SIZE-1), dest) != TRANSFER_COMPLETED) {}
    // 取得したページはdestで示されたSRAMで利用可能です。
}

```

5. AT25F1024/2048/4096デバイス: ドライバ使用例

この例は進行を妨げないPutCharArray関数からの恩恵を示します。

```

main()
{
    int start_address = 0x100;           // 直列メモリ書き込み開始アドレス
    char* src = malloc(PAGE_SIZE* sizeof(char)); // 書き込みデータ列開始アドレス
    char* dest = malloc(PAGE_SIZE * sizeof(char)); // 読み込みデータ列開始アドレス
    char AccessStatus = BUSY;           // 直列メモリ状態符号(初期値=多忙)

    ////////////////////////////////////////////////////////////////////
    // SPI周辺機能初期化
    ////////////////////////////////////////////////////////////////////
    init_spi_master();

    // ソフトウェア書き込み保護禁止
    do {AccessStatus = SetWriteProtectedArea(NONE);} // ソフトウェア書き込み保護なし設定
    while (AccessStatus == BUSY); // 直列メモリ命令完了待機
    if (AccessStatus == HW_PROTECTED) // ハードウェア書き込み保護(/WP=L)ならば、
    {
        // ここでハードウェア書き込み保護検出を処理してください。
    }
    // 関連するセクタの消去
    do {AccessStatus = EraseSector(start_address);} // 書き込みセクタ消去
    while (AccessStatus == BUSY); // 直列メモリ命令完了待機
    // 任意: 異常処理
    if (AccessStatus == OUT_OF_RANGE) // アドレス範囲外ならば、
    {}
    else if (AccessStatus == DATA_WR_PROTECTED) // データ保護(BPx>0)ならば、
    {}
    else
    {
        // ページ書き込み
        while (PutCharArray(start_address, (PAGE_SIZE-1), src) == BUSY) {}
    }
    ////////////////////////////////////////////////////////////////////
    // 使用者コードはここです。(SPI割り込み処理で書き込みアクセスが実行されつつあります。)
    ////////////////////////////////////////////////////////////////////

    // 任意: SPIメモリでの書き込み禁止 (進行防止操作(準備可待機))
    while (WriteCommand(WRDI) != TRANSFER_COMPLETED) {}

    ////////////////////////////////////////////////////////////////////
    // 追加の使用者コードはここです。
    ////////////////////////////////////////////////////////////////////

    // ページ読み込み
    while (GetCharArray(start_address, (PAGE_SIZE-1), dest) != TRANSFER_COMPLETED) {}
    // 取得したページはdestで示されたSRAMで利用可能です。
}

```

6. 開発環境

6.1. ハードウェア

- ・ポートBでの8ピンPDIPソケットを持つSTK500基板
- ・AVR ATmega168
- ・PDIP外圍器のAT25256A
- ・PDIP外圍器のAT25F4096
- ・デバッグWIREを持つJTAGICEmk II インサーキット エミュレータ

6.2. ソフトウェア

- ・IAR Embedded Workbench – EWAVR 3.20C
- ・AVR Studio 4.10版

7. 開発環境

下表はmap一覧ファイルに基づく計算です。コード最適化選択はHighに設定されました。
主ルーチンのコード量は含まれません。

表7-1. コード量

目的対象デバイス	コード量 (バイト)
AT25256A	752
AT25F4096	1086

注: コード量は参考例としての現状値です。
これらの数値はソースでの最適化や改良によって更新されるべき値です。

8. 参照

データシート:

- ・SPI直列型EEPROM AT25128AとAT25256A
- ・SPI直列型フラッシュ メモリ AT25F4096
- ・AVR ATmega168V



本社

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia

Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe

Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines
Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan

104-0033 東京都中央区
新川1-24-8
東熱新川ビル 9F
アトメル ジャパン株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製造拠点

Memory

2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3
France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR
Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn
Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-76-58-47-50
FAX (33) 4-76-58-47-60

文献請求

www.atmel.com/literature

お断り: 本資料内の情報はATMEL製品と関連して提供されています。本資料またはATMEL製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。ATMELのウェブサイトに表示する販売の条件とATMELの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、ATMELはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえATMELがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益の損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してATMELに責任がないでしょう。ATMELは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。ATMELはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、ATMEL製品は車載応用に対して適当ではなく、使用されるべきではありません。ATMEL製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

© Atmel Corporation 2005. 全権利予約済 ATMEL®、ロゴとそれらの組み合わせ、AVR®とその他はATMEL Corporationの登録商標または商標またはその付属物です。他の用語と製品名は一般的に他の商標です。

© HERO 2014.

本応用記述はATMELのAVR107応用記述(doc2595.pdf Rev.2595A-03/05)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。