

AVR329 : USBファームウェア基本設計

要点

- 低速(1.5Mbps)と全速(12Mbps)のデータ速度
- 制御、大量(バルク)、等時(アイソクロノス)、割り込みの転送形式
- AVR USBソフトウェア ライブラリでの標準または独自のUSB装置クラス
- 6つまでのデータエンドポイント
- 単一または2重の緩衝

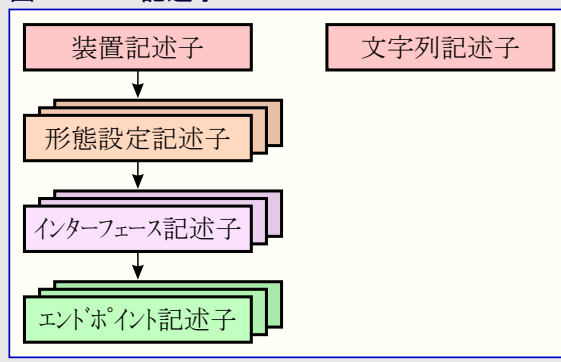
1. 序説

この資料の狙いはUSBファームウェアを記述して基本設計の概要を与えることです。主要なファイルはファームウェアの独自設定とあなた自身の応用を構築する最も簡単な方法を使用者に与えるために記述されています。本資料の内容を理解するために少なくともUSB仕様2.0(www.usb.org)の9章の知識が必要とされます。

2. USB記述子

列挙(接続認識)処理中、ホストはそれを識別して正しいドライブを設定するために多数の記述子を装置に問います。各USB装置はホストによって認証されるために少なくとも右図で示される記述子を持つでしょう。

図2-1. USB記述子



2.1. 装置記述子

USB装置は1つの装置記述子だけを持てます。この記述子は装置全体を表明します。それはUSB版、エンドポイント0の最大パケット容量、供給者ID、製品ID、装置が持ち得る可能な形態設定数などについての情報を与えます。

この下の表はこの記述子の形式を示します。

表2-1. 装置記述子

領域名	説明
bLength	記述子の大きさ
bDescriptorType	装置記述子
bcdUSB	USB版番号
bDeviceClass	クラス符号(0の場合はクラスが各インターフェースによって指定されます、\$FFの場合は供給者によって指定されます。)
bDeviceSubClass	(USB協会により割り当てられた)補助クラス符号
bDeviceProtocol	(USB協会により割り当てられた)規約符号
bMaxPacketSize	エンドポイント0のバイトでの最大パケット容量。それは8(低速)、16,32,64(全速)でなければなりません。
idVendor	(USB協会により割り当てられた)供給者識別
idProduct	(製造(供給)者により割り当てられた)製品識別
bcdDevice	(製造(供給)者により割り当てられた)装置版番号
iManufacturer	製造(供給)者記述子の文字列配列への指標
iProduct	製品記述子の文字列配列への指標
iSerialNumber	通番記述子の文字列配列への指標
bNumConfigurations	形態設定数



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. 7603A-02/06, 7603AJ1-03/14

2.2. 形態設定記述子

USB装置は1つよりも多くの形態設定記述子を持てますが、装置の大多数は単一の形態設定を使用します。この記述子は電力供給形態(自己給電またはバス給電)、装置によって消費され得る最大電力、装置に付属するインターフェース、全てのデータ記述子の総容量などを指定します。

例えば1つの装置は、それがバスによって給電される時の一方と、それが自己給電にされる時の他方の、2つの形態設定を持つことができます。異なる転送形態を使用する形態設定も考えられます。

右の表はこの記述子の形式を示します。

表2-2. 形態設定記述子

領域名	説明
bLength	記述子の大きさ
bDescriptorType	形態設定記述子
wTotalLength	記述子の総容量
bNumInterfaces	インターフェース数
bConfigurationValue	形態設定番号
bmAttributes	バス給電または自己給電、遠隔起動
bMaxPower	2mA単位の最大消費電流

2.3. インターフェース記述子

1つの装置は1つよりも多くのインターフェースを持つことができます。この記述子によって与えられる主な情報はこのインターフェースによって使用されるエンドポイント数とUSBクラス、補助クラスです。

この下の表はこの記述子の形式を示します。

表2-3. インターフェース記述子

領域名	説明
bLength	記述子の大きさ
bDescriptorType	インターフェース記述子
bInterfaceNumber	インターフェース番号
bAlternateSetting	置き換えるインターフェースを選ぶのに使用
bNumEndpoints	(エンドポイント0を除く)エンドポイント数
bInterfaceClass	(USB協会により割り当てられた)クラス符号
bInterfaceSubClass	(USB協会により割り当てられた)補助クラス符号 (0:補助クラスなし、1:パート インターフェース補助クラス)
iInterface	使用するインターフェースを記述するための文字列配列への指標

2.4. エンドポイント記述子

この記述子は、方向(INまたはOUT)、支援する転送形式(割り込み、大量、等時)、エンドポイントの容量、割り込み転送形態の場合に於けるデータ転送の間隔などのようなエンドポイントのパラメータを記述するために用いられます。

この下の表はこの記述子の形式を示します。

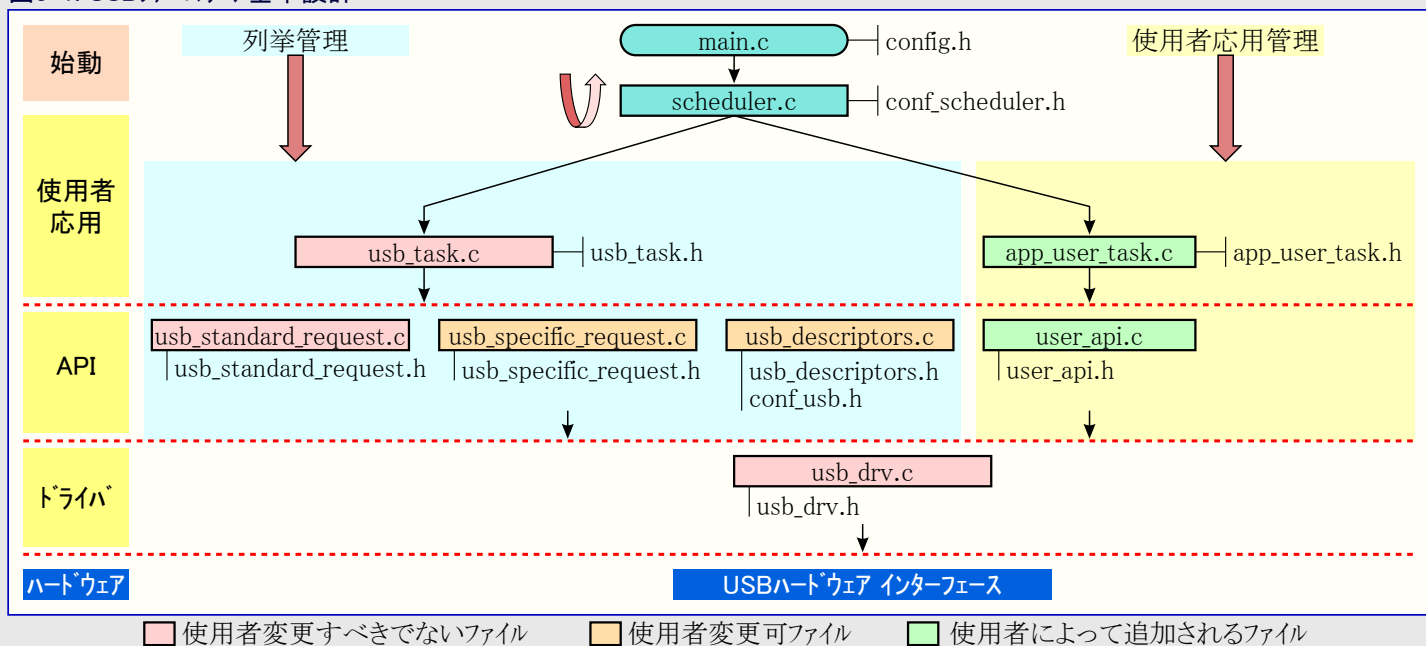
表2-4. エンドポイント記述子

領域名	説明
bLength	記述子の大きさ
bDescriptorType	エンドポイント記述子
bEndpointAddress	エンドポイントのアドレス (ビット7:方向,0=出力,1=入力、ビット6~4:予約(0に設定)、ビット3~0:エンドポイント番号)
bmAttributes	ビット7~2:等時転送を除いて予約 ビット1,0:転送形式 (11:割り込み,10:大量,01:等時,00:制御) 低速(Low-Speed)では制御と割り込みの動作形態だけが許されます。
wMaxPacketSize	このエンドポイントが支援する最大データ容量
bInterval	エンドポイントのデータ転送を要求するための時間間隔です。この値はフレーム数(ms)で与えられます。 大量と制御の転送では無視されます。 等時に対しては1~16(ms)間の値を設定してください。 全速での割り込み転送に対しては1~255(ms)の値を設定してください。 低速での割り込み転送に対しては10~255(ms)の値を設定してください。

3. ファームウェア基本設計

下図で示されるように、USBファームウェアの基本設計はお客様のどんなハードウェア インターフェースをも避けるように設計されています(ドライバ層は使用者によって変更されるべきではありません)。使用者は自身の応用を構築するために、API(`usb_standard_request.c`ファイルは変更されるべきではありません)と使用者応用層とのインターフェースを行わなければなりません。

図3-1. USBファームウェア基本設計



これより下は以下の主要ファイルの説明です。

`main.c`、`config.h`、`scheduler.c`、`conf_scheduler.h`、`usb_task.c`、`usb_task.h`、`usb_standard_request.c`、`usb_standard_request.h`、`usb_specific_request.c`、`usb_specific_request.h`、`usb_drv.c`、`usb_drv.h`、`app_user_task.c`、`user_api.c`

4. 始動

始動は`main.c`と`sheduler.c`の2つファイルを含みます。これらのファイルは作業の始めと進捗を管理します。

4.1. main.c

このファイルは応用に対する主な入口です。計画機構の開始も実行します。

4.2. config.h

このファイルはクロック定義のようなシステム形態設定を含みます。例えば16MHzのクロック周波数を定義するには以下のように宣言しなければなりません。

```
#define FOSC 16000 // 発振器周波数(kHz)
```

4.3. scheduler.c

このファイルは使用者によって変更されるべきではありません。これは作業を呼ぶためのルーチンを管理します。各作業は`conf_scheduler.h`で指定された順に従って呼び出されます。

4.4. conf_scheduler.h

計画機構によって呼び出される作業はこのファイルで定義されなければなりません。初期化作業は只一度呼び出され、その他は継続的に呼び出されます。自身の作業を追加するため、使用者はこのファイル内でそれらを宣言しなければなりません。

```
例: #define Scheduler_task_1_init usb_task_init
      #define Scheduler_task_1 usb_task
      #define Scheduler_task_2_init mouse_task_init
      #define Scheduler_task_2 mouse_task
```

上の例の宣言を用いることで、計画機構は`usb_task_init`、次に`mouse_task_init`を1度呼び出すことによって始まり、その後は`mouse_task`が後続する`usb_task`を継続的に呼び出します。

5. 使用者応用

使用者応用層は最終使用者インターフェースを管理します。使用者は多数の応用作業を追加するでしょう。USB作業(タスク)はATMELによって提供され、使用者はそれを変更すべきではありません。

5.1. usb_task.c

このファイルは全てのUSB要求(標準と特定)を実行します。これはUSB列挙(接続認識)処理と全ての非同期事象(休止、再開、リセット、起動など)を管理します。このファイルは使用者によって変更されるべきではありません。

5.2. app_user_task.c

使用者は1つよりも多くの作業を追加することができ、それはその応用に依存します。これらの作業は使用者応用(最終使用者とのインターフェース)のレベルより高く管理します。

例えば、マウス応用に関して、使用者はapp_user_task.cとして釦と感知器の管理を追加しなければなりません。

6. API

API層は使用者応用層とドライバ層間のインターフェースを許します。それは使用者に対してドライバ特権を隠します。

6.1. usb_standard_request.c

このファイルは全てのUSB標準要求を管理します。これらの要求はどのUSBクラス列挙(接続認識)とも同じです。使用者はこのファイルを変更すべきではありません。

6.2. usb_standard_request.h

このファイルは標準要求パラメータの値を定義します。

6.3. usb_specific_request.c

このファイルは特定要求を管理します。これらの要求は応用に依存し、使用者はそれらを管理するために関連する関数を追加しなければなりません。

6.4. usb_specific_request.h

このファイルは特定要求のパラメータに関する値を定義します。応用が特定要求を持つ場合、使用者はこのファイルにそれらの要求のパラメータ/値を追加しなければなりません。

6.5. usb_descriptor.c

このファイルに於いてusb_descriptor.hで定義される列挙記述子構造体の全てのパラメータ/値を満たします。このファイルは自身の応用を作る使用者に関して重要です。usb_descriptor.hで追加されるどの記述子の値もこのファイルで満たされなければなりません。

6.6. usb_descriptor.h

このファイルは記述子の構造体と全てのパラメータ/値を含みます。usb_descriptor.cにどんな記述子を追加するのにも先立って、使用者はこのファイルで関連する構造体型と値の定義を追加しなければなりません。

6.7. conf_usb.h

このファイルは応用によって使用されるエンドポイント番号の定義を定義します。

例えば、マウス応用エンドポイントとしてエンドポイント1の使用を欲したなら、conf_usb.hファイルで下のように定義すべきです。

```
#define EP_MOUSE_IN 1 //! マウス割り込みINエンドポイントの番号
```

6.8. user_api.c

このファイルは使用者によって追加されます。これはドライバレベルと使用者応用間のインターフェースを行います。

例えば、このファイルはPCと装置間のデータの転送を管理することができます。

7. ドライバ

ドライバ層は使用者に対して全てのハードウェアの複雑さを隠すことを許します

7.1. usb_drv.c

このファイルは使用者とUSBハードウェア間のインターフェースを提供し、それは全てのUSBルーチンを含みます。使用者はこのファイルを変更すべきではありません。

7.2. usb_drv.h

このファイルはUSB低位ドライバ定義を含みます。

8. よくある質問

8.1. VIDとPIDを変更する方法は?

VID(供給者ID)とPID(製品ID)はホストによる装置の識別を許します。各製造業者は自身のVIDを持つべきで、これは全ての製品に対して同じです(これはUSB協会によって割り当てられます)。各製品は自身のPIDを持つべきです(これは製造業者によって割り当てられます)。

VIDとPIDの値は`usb_descriptor.h`で定義されます。それらを変更するには、その値を以下のように変更しなければなりません。

```
// USB装置記述子 (USB Device descriptor)
#define VENDOR_ID      0x03EB      // ATMELの供給者ID = $03EB
#define PRODUCT_ID     0x201C
```

8.2. どうしたら文字列記述子の値を変更できますか?

文字列記述子の値は`usb_descriptor.h`で定義されます。例えば製品名の値を変更するには、以下のように値を変更しなければなりません。

文字列値の長さ:

```
#define USB_PN_LENGTH      18
```

文字列値:

```
#define USB_PRODUCT_NAME ¥
{ Usb_unicode(' A' ) ¥
, Usb_unicode(' V' ) ¥
, Usb_unicode(' R' ) ¥
, Usb_unicode(' ' ) ¥
, Usb_unicode(' U' ) ¥
, Usb_unicode(' S' ) ¥
, Usb_unicode(' B' ) ¥
, Usb_unicode(' ' ) ¥
, Usb_unicode(' M' ) ¥
, Usb_unicode(' O' ) ¥
, Usb_unicode(' U' ) ¥
, Usb_unicode(' S' ) ¥
, Usb_unicode(' E' ) ¥
, Usb_unicode(' ' ) ¥
, Usb_unicode(' D' ) ¥
, Usb_unicode(' E' ) ¥
, Usb_unicode(' M' ) ¥
, Usb_unicode(' O' ) ¥
}
```

8.3. どうすれば私の装置を自己給電またはバス給電形態で形態設定できますか?

装置を自己給電またはバス給電の形態に形態設定するためのパラメータは`usb_descriptor.h`ファイルで定義されます。これより下は各形態の定義です。

バス給電形態:

```
// USB形態設定記述子 (USB Configuration descriptor)
#define CONF_ATTRIBUTES    USB_CONFIG_BUSPOWERED
```

自己給電形態:

```
// USB形態設定記述子 (USB Configuration descriptor)
#define CONF_ATTRIBUTES    USB_CONFIG_SELFPOWERED
```

8.4. どうしたら新しい記述子を追加できますか？

あなたの応用に新しい記述子を追加するには、以下の段階に従わなければなりません。

1. `usb_descriptor.h`ファイルに於いて記述子パラメータの値とその構造体型を定義してください。

例えばHID記述子とその構造体は`usb_descriptor.h`ファイルで以下で示されるように定義されるべきです。

```

/* _____HID descriptor_____ */
#define HID                0x21
#define REPORT             0x22
#define SET_REPORT        0x02
#define HID_DESCRIPTOR    0x21
#define HID_BDC           0x1001
#define HID_COUNTRY_CODE  0x00
#define HID_CLASS_DESC_NB 0x01
#define HID_DESCRIPTOR_TYPE 0x22

/* _____U S B   H I D   D E S C R I P T O R _____ */
typedef struct {
    U8  bLength;           /* この記述子のバイトでの量 */
    U8  bDescriptorType;  /* HID記述子形式 */
    U16 bscHID;           /* 2進化10進仕様、公開 */
    U8  bCountryCode;     /* ハードウェア目的対象国 */
    U8  bNumDescriptors;  /* 伴うHIDクラス記述子数 */
    U8  bRDescriptorType; /* 報告記述子形式 */
    U16 wDescriptorLength; /* 報告記述費の合計長 */
} S_usb_hid_descriptor;

```

2. `usb_descriptor.h`ファイルに於いて`s_usb_user_configuration_descriptor`構造体のための記述子を追加してください。

```

typedef struct
{
    S_usb_configuration_descriptor  cfg_mouse;
    S_usb_interface_descriptor     ifc_mouse;
    S_usb_hid_descriptor           hid_mouse;
    S_usb_endpoint_descriptor      epl_mouse;
} S_usb_user_configuration_descriptor;

```

3. `usb_descriptor.C`ファイルに於いて形態設定記述子の`wTotalLength`パラメータに新しい記述子の量を追加し、記述子の値を追加してください(下の例をご覧ください)。

```

code S_usb_user_configuration_descriptor usb_conf_desc = {
    { sizeof(S_usb_configuration_descriptor)
    , CONFIGURATION_DESCRIPTOR
    , Usb_write_word_enum_struct(sizeof(S_usb_configuration_descriptor)¥
    +sizeof(S_usb_interface_descriptor)      ¥
    +sizeof(S_usb_hid_descriptor)           ¥
    +sizeof(S_usb_endpoint_descriptor)      ¥
    )
    , NB_INTERFACE
    , CONF_NB
    , CONF_INDEX
    , CONF_ATTRIBUTES
    , MAX_POWER
    }
    ,
    { sizeof(S_usb_interface_descriptor)
    , INTERFACE_DESCRIPTOR
    , INTERFACE_NB_MOUSE
    , ALTERNATE_MOUSE
    , NB_ENDPOINT_MOUSE
    , INTERFACE_CLASS_MOUSE
    , INTERFACE_SUB_CLASS_MOUSE
    , INTERFACE_PROTOCOL_MOUSE
    , INTERFACE_INDEX_MOUSE
    }
}

```

```

}
,
{ sizeof(S_usb_hid_descriptor)
, HID_DESCRIPTOR
, HID_BDC
, HID_COUNTRY_CODE
, HID_CLASS_DESC_NB
, HID_DESCRIPTOR_TYPE
, Usb_write_word_enum_struct(sizeof(S_usb_hid_report_descriptor_mouse))
}
,
{ sizeof(S_usb_endpoint_descriptor)
, ENDPOINT_DESCRIPTOR
, ENDPOINT_NB_1
, EP_ATTRIBUTES_1
, Usb_write_word_enum_struct(EP_SIZE_1)
, EP_INTERVAL_1
}
};

```

4. 新しい記述子の管理に関連する全ての関数の追加を忘れないでください。

8.5. どうしたら新しいエンドポイントを追加できますか?

新しいエンドポイントを追加するには、以下の段階に従わなければなりません。

1. [USB記述子章](#)で説明されたように、エンドポイントはインターフェースに属します。新しいエンドポイントを追加するために最初に行うことはNB_ENDPOINTパラメータ値を1つ増やすことです。この値はusb_descriptor.hファイルで定義され、インターフェース記述子パラメータに属します。

```

// USBインターフェース記述子 (Interface descriptor)
#define INTERFACE_NB          xx
#define ALTERNATE            xx
#define NB_ENDPOINT          xx           // このパラメータ=インターフェースのエンドポイント数
#define INTERFACE_CLASS      xx
#define INTERFACE_SUB_CLASS  xx
#define INTERFACE_PROTOCOL   xx
#define INTERFACE_INDEX      xx

```

2. 次の段階はエンドポイント記述子の値を定義することです。これらの値はusb_descriptor.hで定義されなければなりません。

```

// USBエンドポイント1記述子 (Endpoint 1 descriptor) FS
#define ENDPOINT_NB_1        (EP_MOUSE_IN | 0x80)
#define EP_ATTRIBUTES_1     0x03           // 大量(バルク)=0x02, 割り込み=0x03
#define EP_IN_LENGTH_1      8
#define EP_SIZE_1           EP_IN_LENGTH_1
#define EP_INTERVAL_1       0x02         // ホストからの割り込みポーリング間隔

```

応用によって使用されるエンドポイント番号を指定するために、conf_usb.hでEP_MOUSE_INが定義されます。

3. 形態設定記述子にエンドポイント記述子を追加してください(直前のよくある質問点で説明されるように進めてください)。

4. usb_specific_request.cにハードウェア初期化呼び出しを追加してください。

```

void usb_user_endpoint_init(U8 conf_nb)
{
    usb_configure_endpoint(EP_MOUSE_IN, ¥
        TYPE_INTERRUPT,
        DIRECTION_IN,
        SIZE_8,
        ONE_BANK,
        NYET_ENABLED);
}

```

9. コーディング様式

これより下で説明されるコーディング様式はファームウェアを理解するのに重要です。

- 定義された定数は大文字を使用します。

```
#define FOSC 8000
```

- マクロ関数は先頭文字を大文字として使用します。

```
#define Is_usb_sof() ((UDINT & MSK_SOFI) ? TRUE: FALSE)
```

- 使用者応用はusb_conf.hで以下のように定義された引っ掛け(フック)によって各USB事象で自身の特定命令を実行することができます。

```
#define Usb_sof_action()      sof_action();
```

注: 引っ掛け(フック)関数は短時間を必要とする操作だけを実行すべきです。

- Usb_unicode()マクロ関数はUSB規約でユニコード文字が交換される処(文字列記述子など)毎に使用されるべきです。



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 2006. 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AVR329応用記述(doc7603.pdf Rev.7603A-02/06)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。