AVR309 : USB-RS232規格変換器

要点

- ファームウェアにUSB規約を実装
- USB2.0適合の低速(Low Speed,1.5Mbps)USB支援
- 1K語(2Kﾊﾞｲﾄ)以上の極小AVRﾃﾞﾊﾞｲｽでの実装実行
- 必要とする少しの外部部品
  - 低速(Low Speed)USB検出用の1本の抵抗
  - 電圧の分割/安定化と、その濾波器部品
- 実装機能
  - 入出力ﾋﾟﾝ直接制御
  - USB-RS232変換器
  - EEPROM一時記録ﾚｼﾞｽﾀ
- 容易に実装可能な使用者定義機能
  - USBでのTWI(2線直列ｲﾝﾀｰﾌｪｰｽ)制御
  - USBでのA/D変換やD/A変換
- 開発者は(PC側から見れる)ﾃﾞﾊﾞｲｽ識別を任意設定可能
- PC側のｿｰｽ ｺｰﾄﾞと資料を完全に支援
  - MS Windows USBﾄﾞﾗｲﾊﾞ
  - DLLﾗｲﾌﾞﾗﾘ関数
  - Delphiでの実演応用ﾌﾟﾛｸﾞﾗﾑ
- 開発者用DLL経由での(Delphi, C++, Visual Basic)通信方法の例

1. 序説

USBは最終使用者に対して簡単な応用(再起動なしのPlug and Play)のため、非常に一般的となりました。然しながら開発者にとってﾃﾞﾊﾞｲｽへのUSB実装は、例えばRS232と比較したとき、大変複雑に込み入っています。加えてPC側で支援するｿﾌﾄｳｪｱとしてﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞが必要とされます。これはRS232基準の通信が多くの装置製造業者で未だ非常に一般的だからです。このｲﾝﾀｰﾌｪｰｽは充分に確立され、ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑでの支援も良いのですが、最近、USBﾎﾟｰﾄが優勢になり、物理的なRS232ﾎﾟｰﾄは標準的なPCｲﾝﾀｰﾌｪｰｽから削除されるようになってきています。

外部装置へのUSB実装は2つの方法で行えます。

1. USBｲﾝﾀｰﾌｪｰｽを実装するﾊｰﾄﾞｳｪｱとﾏｲｸﾛ ｺﾝﾄﾛｰﾗの使用。これにはUSB動作を知り、それに対応したﾌｧｰﾑｳｪｱをﾏｲｸﾛ ｺﾝﾄﾛｰﾗに書く必要があります。加えて(ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑが標準USBｸﾗｽを含まない限り)、PC側ﾄﾞﾗｲﾊﾞを作成することが必要です。この不利な条件は(小規模開発者とｱﾏﾁｭｱにとって、これが主に不利な条件ですが)、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗのこの種の利便性をなくし、単純なRS232のﾏｲｸﾛ ｺﾝﾄﾛｰﾗに比して高価になります。
2. 2つ目の選択はUSBと他のｲﾝﾀｰﾌｪｰｽ間の万能変換器をいくつか使用することです。このその他ｲﾝﾀｰﾌｪｰｽは通常、RS232, TWI, 8ﾋﾞｯﾄ ﾃﾞｰﾀ ﾊﾞｽでしょう。この場合、特別なﾌｧｰﾑｳｪｱの必要はなく、USB動作を知る必要もなく、全てを解決するための或るﾄﾞﾗｲﾊﾞを提供する、変換器開発者としてﾄﾞﾗｲﾊﾞを書く必要もありません。不利な条件は完成したｼｽﾃﾑが高価なことと、完成した製品が非常に大きくなることです。

本資料で表された解決方法は、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗのﾌｧｰﾑｳｪｱ内のUSB規約のｴﾐｭﾚｰﾄを通す低価格ﾏｲｸﾛ ｺﾝﾄﾛｰﾗへのUSB実装です。この設計に対する主な要求は充分な速度が得られました。USBﾊﾞｽは低速(Low Speed)が1.5Mbps、全速(Full Speed)が12Mbps、高速(High Speed)が480Mbpsの速さです。ATtiny2313, ATmega48/88/168は低速(Low Speed)USBのﾊｰﾄﾞ的速度の必要条件に合致する完全な能力があります。とは言え、この解決方法はより高速のUSB速度に対しては推奨されません。

8-bit 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. 2556B-02/06, 2556BJ2-02/14

2. 動作の理屈

物理的なUSB通信に関する広範囲な詳細はｳｪﾌﾞｻｲﾄ www.usb.org で得られます。この資料は大変複雑で初心者にとっては困難です(概ね650頁)。

大変良く、簡単な初心者用説明はCraig Peacookによって書かれた資料「USB in a Nutshell. Making Sense of the USB Standard」(概ね30頁)で得られます(参考文献②) (**訳補**:これは英語のため、代替として USBspcs.pdf (日本語)を同梱します)。

本応用記述の説明はﾃﾞﾊﾞｲｽのﾌｧｰﾑｳｪｱを理解する範囲に限定されます。USBの物理ｲﾝﾀｰﾌｪｰｽは外部装置の電力供給用の2本(VCCとGND)、2つの信号線(DATA+とDATA−)の4信号線から成ります。この電源線はおよそ5V, 最大500mAを与えます。VCCとGNDから装置に電力を供給できます。DATA+, DATA−と名付けられた信号線はﾎｽﾄ(ｺﾝﾋﾟｭｰﾀ)と装置間の通信を扱います。これらの線上の信号は双方向です。電圧ﾚﾍﾞﾙはDATA+がHighﾚﾍﾞﾙのとき、DATA−がLowﾚﾍﾞﾙ(またはその逆)の差動ですが、DATA+とDATA−が同じﾚﾍﾞﾙになるいくつかの場合(EOP:end of pakect, ｱｲﾄﾞﾙ(idle)状態)があります。

従ってUSBを駆動するﾌｧｰﾑｳｪｱの実装に於いて、これら両方の信号の監視と駆動ができなければなりません。

標準USBに従う信号線はUSBﾎｽﾄによって支援されたVCCが4.4～5.25V間、3.0～3.6V間でHigh駆動されなければなりません。故にﾏｲｸﾛ ｺﾝﾄﾛｰﾗがUSB信号線から直接的に電力供給される場合、ﾃﾞｰﾀ信号線は差動電圧ﾚﾍﾞﾙを補償するﾚﾍﾞﾙ変換器を通さなければなりません。他の解決方法はﾎｽﾄによって支援されたVCCを3.3Vに下げるように調整し、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗをその電圧ﾚﾍﾞﾙで動かすことです。

図1. Low Speedﾄﾞﾗｲﾊﾞ信号波形

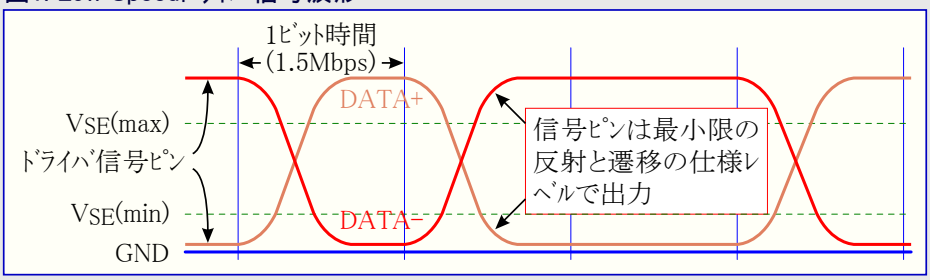
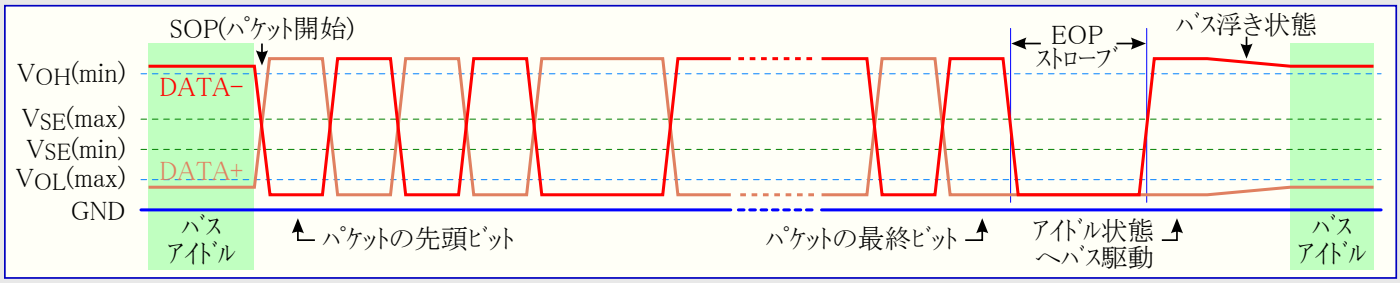


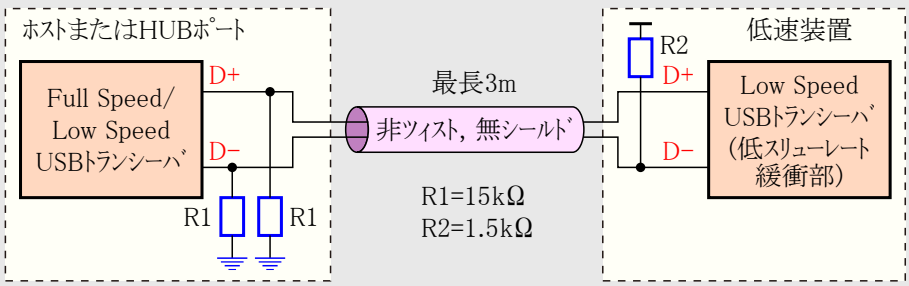
図2. ﾊﾟｹｯﾄ処理電圧ﾚﾍﾞﾙ



USB装置の接続と切断はUSB信号線で感知したｲﾝﾋﾟｰﾀﾞﾝｽを基準に検知されます。低速(Low Speed)USB装置についてはDATA−信号とVCC間に1.5kΩのﾌﾟﾙｱｯﾌﾟ抵抗が必要です(全速(Full Speed)の装置についてはこの抵抗がDATA+に接続されます)。

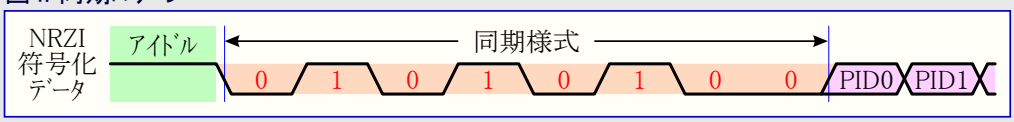
このﾌﾟﾙｱｯﾌﾟを元にﾎｽﾄ ｺﾝﾋﾟｭｰﾀはUSB信号線に接続される新規装置を検出します。

図3. Low Speed装置のｹｰﾌﾞﾙと抵抗接続



ﾎｽﾄが新規装置を検出した後、物理的なUSB規約に従った通信を開始できます。このUSB規約はUARTのようではなく、同期ﾃﾞｰﾀ転送が基本です。送受信での同期化が通信を実行するために必要です。このため送信部は実ﾃﾞｰﾀに先行して小さなﾍｯﾀﾞ(同期様式)を送信します。このﾍｯﾀﾞは2つの0に先行する方形波(101010)で、その後に実ﾃﾞｰﾀが送信されます(LSB先行)。

図4. 同期ﾊﾟﾀｰﾝ



同期を維持するため、USBは全速(Full Speed)装置の場合に秒毎にこの同期様式が送信されることを、また低速(Low Speed)装置の場合に両信号線が0に引かれることを要求します。ﾊｰﾄﾞｳｪｱで実装されたUSB受信部では、この同期化がﾃﾞｼﾞﾀﾙPLL(Phased lock loop)によって保証されます。AVRでの実装では同期様式でﾃﾞｰﾀ採取時間を同期しなければならず、その(方形波)後に2つの0を待ち、最終的にﾃﾞｰﾀの受信を始めます。

(**訳注**) 本節は一般形で説明されています。低速/全速の装置ではDATA−/DATA+のﾌﾟﾙｱｯﾌﾟ側が異なるために論理が異なり、低速/全速の装置間では逆論理になります。**図1.**と**図2.**でのDATA−/DATA+表記は低速(Low Speed)の場合です。図4.とその説明はDATA+を基準にすると、低速装置の場合は逆論理になります。従って低速装置の場合、**図4.**の信号自体はDATA−に現れます。

USBでのﾃﾞｰﾀ受信は全ての時間で送信部と受信部が同期であることを満たさなければならず、従ってﾃﾞｰﾀ列で連続する0または1の流れを送ることは許されません。USB規約はﾋﾞｯﾄ挿入により同期を保証します。これはﾃﾞｰﾀ列の6つの連続する0または1の後に1つの単一変更(1ﾋﾞｯﾄ)が挿入されることを意味します。

USB信号線上の信号はNRZIで符号化されます。NRZIでは各0が現在の信号ﾚﾍﾞﾙを変更することで表され、各1は現在ﾚﾍﾞﾙを保持することで表されます。ﾋﾞｯﾄ挿入に対してこれは論理ﾃﾞｰﾀ列内の6つの連続する論理1後に1つの0ﾋﾞｯﾄが挿入されることを意味します。

図5. ﾃﾞｰﾀのNRZI符号化

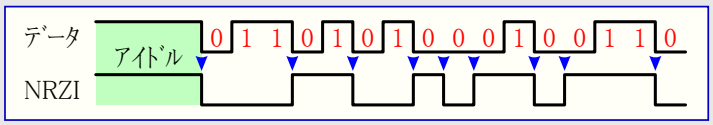
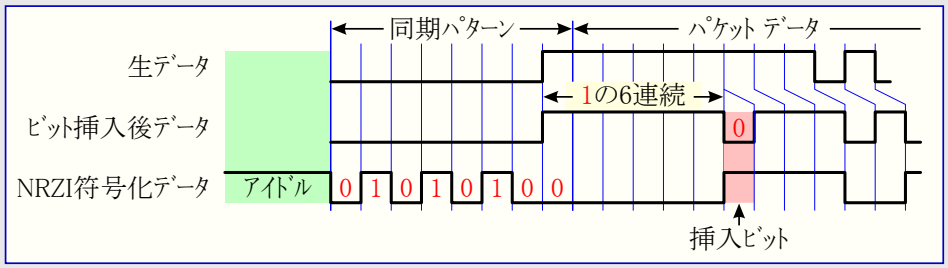
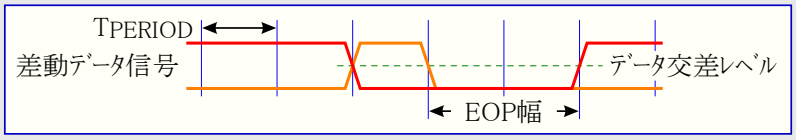


図6. ﾋﾞｯﾄ挿入とﾃﾞｰﾀ符号化手順



ﾃﾞｰﾀ転送の終了通知は両ﾃﾞｰﾀ信号線上の2つの0(物理的にDATA+とDATA-の両方がLowﾚﾍﾞﾙ電圧)から成るEOP(End of packet)によって成されます。EOPには短時間(ﾃﾞｰﾀ速度の最少2周期分)のｱｲﾄﾞﾙ状態が続きます。その後に次の操作が行えます。

図7. EOP幅ﾀｲﾐﾝｸﾞ



同期様式とEOP間のﾃﾞｰﾀはNRZI符号化したﾎｽﾄとUSB装置間の通信です。このﾃﾞｰﾀ列は同期領域(同期様式)、ﾊﾟｹｯﾄID(PID)、ｱﾄﾞﾚｽ領域(ADDR)、ｴﾝﾄﾞﾎﾟｲﾝﾄ領域(ENDP)、ﾃﾞｰﾀ、巡回冗長検査領域(CRC)の各領域から成るﾊﾟｹｯﾄで構成されます。ﾃﾞｰﾀ転送の各種形式でのこれらの領域の使用法は参考文献② (**訳補**:USBspcs.pdf)で適切に説明されます。USBは制御(Control)転送、割り込み(Interrupt)転送、等時(Isochronous)転送、大量(Bulk)転送の4つの転送形式で運用します。これらの転送の各々は各種装置の必要条件に対して専用で、それらの説明は参考文献② (**訳補**:USBspcs.pdf)で得られます。

AVRで作成する装置は制御(Control)転送を使用します。この転送種別は装置設定専用ですが、一般用途にも使用できます。制御(Control)転送は装置が接続される時に(装置からの情報取得、装置のｱﾄﾞﾚｽ設定などの)初期設定用に使用されるため、全てのUSB装置に存在しなければなりません。制御(Control)転送の記述とその内容は参考文献①と② (**訳補**:USBspcs.pdf)で得られます。各制御(Control)転送は設定(Setup)段階、ﾃﾞｰﾀ(Data)段階、状態(Status)段階のそれぞれの段階から成ります。

ﾃﾞｰﾀはUSBで転送されたﾊﾟｹｯﾄで、各々のﾊﾟｹｯﾄは多くのﾊﾞｲﾄを含みます。ﾊﾟｹｯﾄ容量は各装置によって決められますが、仕様によって制限されます。低速(Low Speed)の装置についてのﾊﾟｹｯﾄ容量は8ﾊﾞｲﾄに制限されます。開始領域と終了領域で挟まれた8ﾊﾞｲﾄ長は1つのUSB転送で装置の緩衝部内に受信されなければなりません。ﾊｰﾄﾞｳｪｱ基準のUSB受信部では転送の様々な部分が自動的に復号され、ﾒｯｾｰｼﾞ全体が特定装置に指示されてしまった時にその装置が通知されるだけです。ﾌｧｰﾑｳｪｱ実装でのUSBﾒｯｾｰｼﾞはﾒｯｾｰｼﾞ全体が緩衝部内に受信されてしまった後にﾌｧｰﾑｳｪｱによって復号されなければなりません。これはﾌｧｰﾑｳｪｱ実装に対して必要条件と制限を与えます。この装置はUSBﾒｯｾｰｼﾞ長全体の緩衝部とUSB送信(送信のための準備されたﾃﾞｰﾀ)用の他の緩衝部を持たなければならず、ﾒｯｾｰｼﾞの復号と検査での付随処理時間を管理しなければなりません。加えてﾌｧｰﾑｳｪｱは勿論高速実行と、正確に同期した速度での(物理ﾋﾟﾝから緩衝部への)受信と(緩衝部から物理ﾋﾟﾝへの)送信が必要とされます。これら全ての能力はﾏｲｸﾛ ｺﾝﾄﾛｰﾗの資源(速度とﾌﾟﾛｸﾞﾗﾑ/ﾃﾞｰﾀのﾒﾓﾘ容量)によって制限され、故にﾌｧｰﾑｳｪｱは入念に最適化されなければなりません。いくつかの条件でﾏｲｸﾛ ｺﾝﾄﾛｰﾗの処理能力は最低必要条件に大変近く、従って全てのﾌｧｰﾑｳｪｱはｱｾﾝﾌﾞﾘ言語で書かれなければなりません。

(**訳注**) 本頁も前頁の訳注が引き続き適用され、**図5.**と**図6.**の信号は低速装置の場合、DATA-に現れますが、**図7.**のEOPﾚﾍﾞﾙに注意してください。EOPは低速/全速に拘わらずDATA-とDATA+両方のLow電圧ﾚﾍﾞﾙで定義されます。

3. ハードウェアの実装

USBバスへのマイクロ コントローラ接続の回路図は図8.と図9.で示されます。これらの回路はUSB-RS232変換器の特定目的用に作成されています。これらはEEPROM読み書きと直接ピン制御としての特殊機能も実装されます。

図8. ATtiny2313でのUSBインターフェース
(32バイトFIFOのUSB-RS232変換器, 8ビットI/O制御, 128バイトEEPROM)

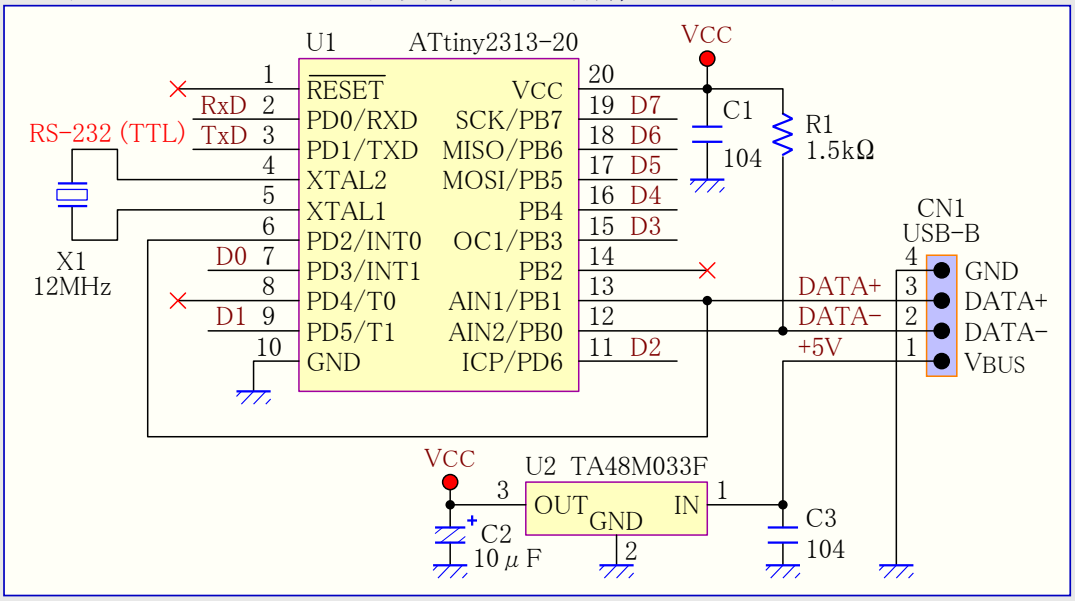
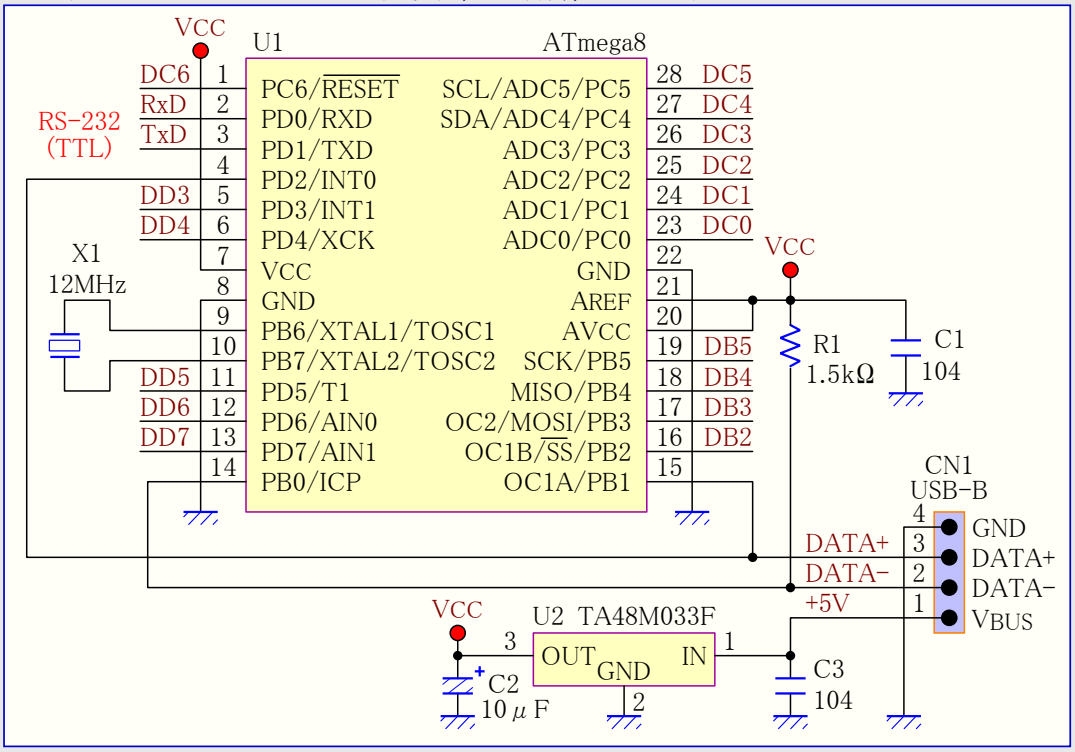


図9. ATmega8(48/88/168)でのUSBインターフェース
(800バイトFIFOのUSB-RS232変換器, I/O制御, EEPROM)



(訳注) 原書でのA版からB版への改訂でAT90S2313がATtiny2313に変更され、A版に存在したATmega8での回路図が削除されました。ATmega8の削除は後継品種でUSARTが拡張I/O領域に配置されたことによるタイミングの問題と思われます。本書では参考のために図9.としてそれを示します。

原書に於ける3.5Vの3端子電圧調整器は入手性から3.3Vに変更しています。AVRの駆動レベルが不足する場合はより高い電圧(例えば3.45V)の3端子レギュレータに変更するか、3端子レギュレータの2番ピンに電位を与えて出力電圧を調整してください。

RESET入力ピンの内部プルアップ抵抗は比較的高抵抗値のため、悪条件で使用する場合、VCCへの接続、10kΩ程度での外部プルアップ、またRSTDISBLヒューズありのデバイスでは外部リセット機能を禁止することが望まれます。電源OFF時の残留電圧が問題となる場合、低電圧検出(BOD)リセットで対処しますが、これをを持たないデバイスでは外部リセット回路が必要になるかもしれません。

低速(Low Speed)規格では先端がAプラグで容量200～450pFのケーブルが装置に付随されなければなりませんが、上図では実験の容易性から、これをBレセクタプルとして記しています。

USBﾃﾞｰﾀ信号線のDATA+とDATA−はAVRのPB1とPB0ﾋﾟﾝに接続されます。この接続はﾌｧｰﾑｳｪｱが高速信号受信に対して巧妙にAVRを使用するため、変更できません。ﾃﾞｰﾀ信号線で捕獲したﾋﾞｯﾄ信号はLSB(PB0:DATA−)からｷｬﾘｰ ﾌﾗｸﾞへ、そしてﾃﾞｰﾀ信号線からのﾋﾞｯﾄを収集する受信ﾚｼﾞｽﾀへと1ﾋﾞｯﾄ右移動されます。PB1(DATA+)は8ﾋﾟﾝのAT90S2323で、このﾋﾟﾝが外部割込みINT0として使用できるため(INT0への追加接続が必要なく、8ﾋﾟﾝ版AVRは利用可能な最少ﾋﾟﾝ数です)、入力信号として使用されます。異なるAVRﾏｲｸﾛ ｺﾝﾋﾟｭｰﾀ間でのﾌｧｰﾑｳｪｱ変更なしの保証を欲するなら、他のAVRではDATA+からINT0ﾋﾟﾝへの外部接続が必要です。

正しいUSB装置接続と信号について、低速(Low Speed)USB装置として動くAVRはDATA−に1.5kΩのﾌﾟﾙｱｯﾌﾟ抵抗を持たなければなりません。

USBﾎｽﾄから供給されたVBUSは4.4～5.25V間で変化するかもしれません。この供給電源は1.5kΩﾌﾟﾙｱｯﾌﾟ抵抗接続とAVRへの供給に先立って3.0～3.6Vに調整安定化されなければなりません。電圧調整器の規模は対象ｼｽﾃﾑの負荷電力に依存します。最適な方法はｼﾘｰｽﾞﾊﾟｽ型調整器を使用することです。非常に安い方法としてﾂｪﾅｰ ﾀﾞｲｵｰﾄﾞを使用することができます。代わりにﾃﾞｰﾀ信号線にﾚﾍﾞﾙ変換器の使用もあります。

その他の部品(ｸﾛｯｸ元としてのｸﾘｽﾀﾙ発振子、電源濾波器用のｺﾝﾃﾞﾝｻ)はﾏｲｸﾛ ｺﾝﾄﾛｰﾗの正しい動作用の機能を提供するだけです。

この少部品数はUSBｲﾝﾀｰﾌｪｰｽを通してｺﾝﾋﾟｭｰﾀと通信できるUSB装置の機能を得るに充分です。これは非常に簡単で安い方法です。装置機能を拡張するために、いくつかの付加部品が追加できます。赤外線(IR)受信を欲するなら、TSOP1738赤外感知器を追加できます。USB−RS232変換装置としての使用を欲するなら、MAX232(TTL−RS232ﾚﾍﾞﾙ変換器)を追加すべきです。LEDや表示器の制御を欲するなら、入出力ﾋﾟﾝに直接または抵抗を経由してそれらを接続してください。

4. ソフトウェアの実装

USB規約の全ての受信と復号はﾌｧｰﾑｳｪｱ段階で実行されます。このﾌｧｰﾑｳｪｱは最初に1つのUSBﾊﾟｹｯﾄ内のUSBﾋﾞｯﾄ列を内部緩衝部に受信します。受信の開始は同期様式に備えるINT0外部割り込みが基準とされます。受信の間中、ﾊﾟｹｯﾄの終了信号(EOP検出)だけが検査されます。これはUSBﾃﾞｰﾀ転送の(ﾏｲｸﾛ ｺﾝﾄﾛｰﾗにとって)非常な速度のためです。受信成功後、ﾌｧｰﾑｳｪｱはﾃﾞｰﾀ ﾊﾟｹｯﾄの符号化を復号し、それらを分析します。最初にそのﾊﾟｹｯﾄがこの装置を意図した対応するｱﾄﾞﾚｽかを検査します。このｱﾄﾞﾚｽは全てのUSB処理で転送され、従って装置は次に転送されたﾃﾞｰﾀが自身用であるかが判ります。USBｱﾄﾞﾚｽを与えた有効なUSB ﾊﾟｹｯﾄを認証する場合、装置はUSBﾎｽﾄにACKﾊﾝﾄﾞｼｪｰｸ ﾊﾟｹｯﾄで答えなければならないため、USBｱﾄﾞﾚｽの復号は非常に素早く行われなければなりません。従ってこれはUSB応答の重要部分です。

このﾋﾞｯﾄ列の受信後、入力緩衝部内にﾋﾞｯﾄ挿入されたNRZI符号化ﾋﾞｯﾄ群を得ます。復号処理では最初に挿入ﾋﾞｯﾄを取り除き、その後にNRZI符号の復号を行います。これら全ての変更は第2緩衝部(受信緩衝部の複写)内で行われます。最初のﾊﾟｹｯﾄが復号されている間、新規ﾊﾟｹｯﾄが受信できます。この時点で復号速度はそれほど重要ではありません。これは装置がそれ自身に対する応答を遅らせることができるからです。復号中にﾎｽﾄが応答について問い合わせたなら、装置は未だ準備できていないのをﾎｽﾄが理解するためのNAKで直ちに応答しなければなりません。このためﾌｧｰﾑｳｪｱは復号中にﾎｽﾄからのﾊﾟｹｯﾄを受信して復号し、自装置に対して意図された処理かどうかを調べ、そして何らかの復号実行中なら、NAKﾊﾟｹｯﾄを送信できなければなりません。その後ﾎｽﾄは再び問い合わせるでしょう。このﾌｧｰﾑｳｪｱは主USB処理の復号も、要求された動作(例えばRS232信号線へのﾃﾞｰﾀ送信と送信完了待機)の実行も、応答に対応する準備も行います。この処理中、装置はﾎｽﾄからのいくつかのﾊﾟｹｯﾄ(通常、装置からの応答を得るためのINﾊﾟｹｯﾄ)によって割り込まれるでしょう。これらのINﾊﾟｹｯﾄに対して装置はNAKﾊﾝﾄﾞｼｪｰｸ ﾊﾟｹｯﾄで応答しなければなりません。装置が必要とした動作を実行してしまい、応答の準備が整ったとき、その内容は最初にCRC対象領域の端から端までの計算とCRC付加、NRZI符号化とﾋﾞｯﾄ挿入をしなければなりません。そこでﾎｽﾄが応答を要求する時にUSB仕様に従って(同期様式からEOPまでの)このﾋﾞｯﾄ列をﾃﾞｰﾀ信号線に送信できます。

4.1. ファームウェア説明

以下でﾌｧｰﾑｳｪｱの主要部分を記述します。このﾌｧｰﾑｳｪｱは、割り込み処理ﾙｰﾁﾝ、復号化ﾙｰﾁﾝ(NRZI符号復号, ﾋﾞｯﾄ挿入の追加/削除, 等々)、USB受信、USB送信、要求動作の復号、要求された動作の実行、の部分に分けられます。

使用者はﾌｧｰﾑｳｪｱに自身の機能を追加できます。顧客仕様機能の作成法のいくつかの例はﾌｧｰﾑｳｪｱ ｺｰﾄﾞ内で得られ、使用者は既存組み込み機能に従った新規拡張装置を書くことができます。例えばTWI支援は直接ﾋﾟﾝ制御用の組み込み機能に従って追加することができます。

4.2. INT0外部割り込み処理ルーチン

外部割り込み0はファームウェア走行中の全時間で活性(有効)です。このルーチンがUSB直列データの受信(別名USB受信)を開始します。外部割り込みはINT0ピンの上昇端で起こります(この上昇端は図4.で見られるUSBパケットの同期様式の始まりを示します)。これがUSB受信ルーチンを起動します。

始めにデータ採取がビット幅の中央に同期化されなければなりません。これは(方形波信号である)同期様式に対応して行われます。ビット幅が8XTALクロック周期だけで、且つ割り込みの発生が(±4周期)遅延され得るため、同期パターンでのエッジ同期は注意深く行われなければなりません。同期様式の終了とデータ ビットの始まりは同期パケット内の最後の2つのLowビットに対応して検出されます(図4.をご覧ください)。

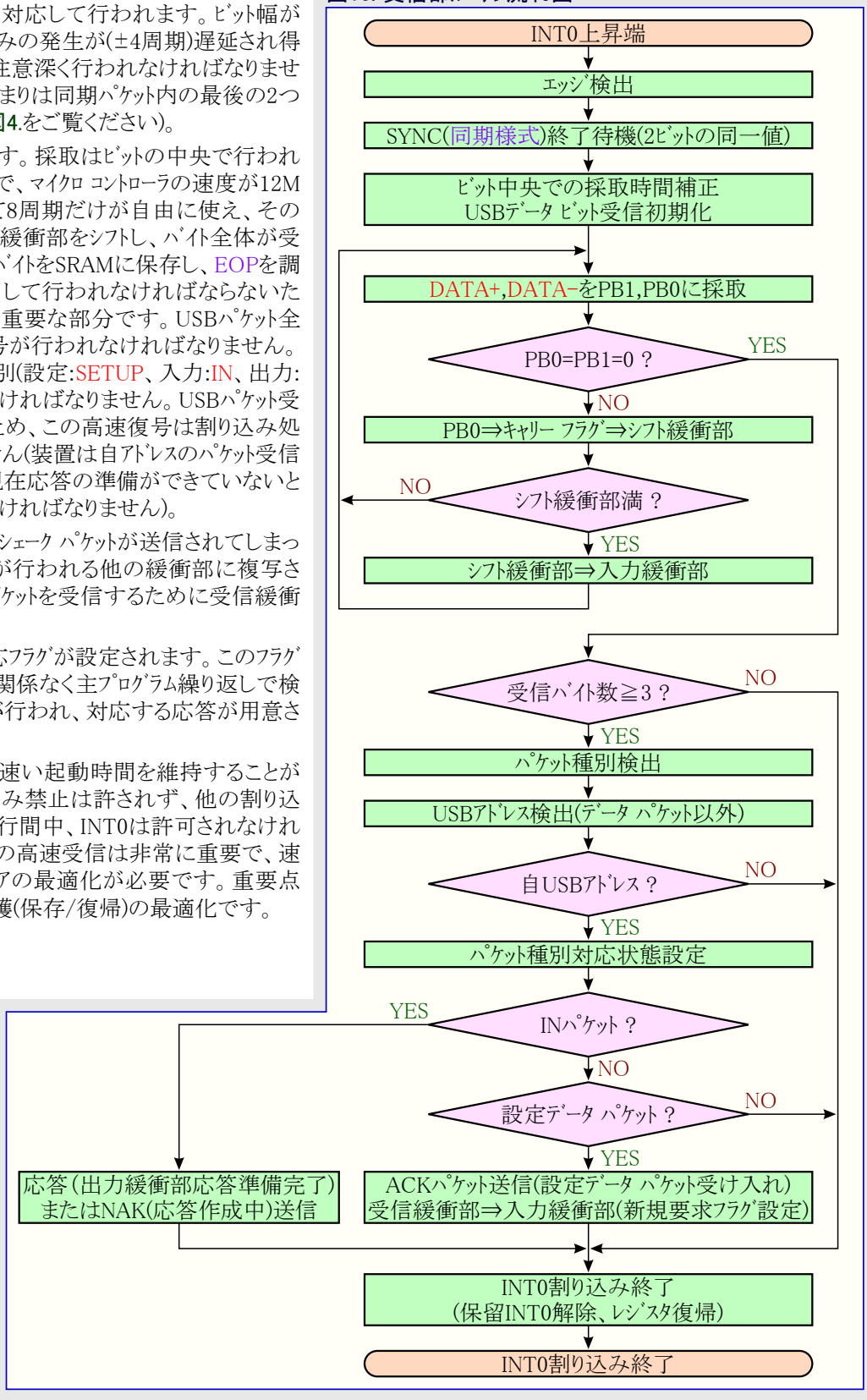
その後、実データの採取が開始されます。採取はビットの中央で行われます。データ速度が1.5Mbps(1.5MHz)で、マイクロ コントローラの速度が12MHzなので、データ ビットの採取に対して8周期だけが自由に使え、その結果をバイト緩衝部に保存して、バイト緩衝部をシフトし、バイト全体が受信されてしまったかを検査して、そのバイトをSRAMに保存し、EOPを調べます。全てが正確なタイミングで同期して行われなければならないため、多分この部分はファームウェアの最も重要な部分です。USBパケット全体が受信されてしまうと、パケットの復号が行われなければなりません。最初に受信したUSBアドレスとパケット種別(設定:SETUP、入力:IN、出力:OUT、データ:DATA)を素早く確定しなければなりません。USBパケット受信後、非常に早く応答が要求されるため、この高速復号は割り込み処理ルーチン側で行われなければなりません(装置は自アドレスのパケット受信時にACK、自宛パケットを受信したが現在応答の準備ができていないときにNAKハンドシェーク パケットで応答しなければなりません)。

受信ルーチンの最後で(ACK/NAKハンドシェーク パケットが送信されてしまった後)、採取したデータ緩衝部は復号が行われる他の緩衝部に複写されなければなりません。これは新規パケットを受信するために受信緩衝部を開放するためです。

受信中にパケット種別が復号され、対応フラグが設定されます。このフラグはマイクロ コントローラ速度の必要条件に関係なく主プログラム繰り返しで検査され、その値に応じた適切な動作が行われ、対応する応答が用意されます。

INT0は全ファームウェア ルーチンで非常に速い起動時間を維持することが許されなければならず、従って割り込み禁止は許されず、他の割り込み(例えば直列信号受信割り込み)実行間中、INT0は許可されなければなりません。INT0割り込みルーチンでの高速受信は非常に重要で、速度と正確なタイミングについてファームウェアの最適化が必要です。重要点の1つは割り込みルーチンでのレジスタ保護(保存/復帰)の最適化です。

図10. 受信部ルーチン流れ図



4.3. 主ﾌﾟﾛｸﾞﾗﾑ繰り返し

主ﾌﾟﾛｸﾞﾗﾑ繰り返しは非常に簡単です。それはいくつかのﾃﾞｰﾀ受信時に何を行うかが表された動作ﾌﾗｸﾞを検査することだけが必要とされます。加えてUSBｲﾝﾀｰﾌｪｰｽがﾘｾｯﾄされる(両信号線が長時間Lowﾚﾍﾞﾙ)かどうかを調べ、そうならば装置を再初期化します。何かが起きると(例えば動作ﾌﾗｸﾞが活性(有効))、対応する動作(ﾊﾟｹｯﾄのNRZI復号、ﾋﾞｯﾄ挿入の削除、送信緩衝部での要求された応答の準備(ﾋﾞｯﾄ挿入とNRZI符号化))が呼び出されます。そして送信について応答が準備されたことを示すために或るﾌﾗｸﾞが活性(有効)にされます。出力緩衝部のUSB信号線への物理的な送信はINﾊﾟｹｯﾄへの応答として受信ﾙｰﾁﾝで実行されます。

4.4. ﾌｧｰﾑｳｪｱの短い説明

以下でﾌｧｰﾑｳｪｱのｻﾌﾞﾙｰﾁﾝとそれらの目的が簡単に記述されます。

4.4.1.	Reset:	AVR マイクロ コントローラのスタック、シリアル信号線、USB緩衝部、割り込みなどの資源の初期化。
4.4.2.	Main:	主ﾌﾟﾛｸﾞﾗﾑ繰り返し。動作ﾌﾗｸﾞ値を調べ、ﾌﾗｸﾞが設定されていれば必要とされた動作を実行します。更にこのﾙｰﾁﾝはﾃﾞｰﾀ信号線上のUSBﾘｾｯﾄについて調べ、その状態ならばﾏｲｸﾛ ｺﾝﾄﾛｰﾗのUSBｲﾝﾀｰﾌｪｰｽを再初期化します。
4.4.3.	Int0Handler:	INT0外部割り込み処理ﾙｰﾁﾝ。USBﾃﾞｰﾀ信号線でのｴﾐｭﾚｰﾄ、緩衝部へのﾃﾞｰﾀ保存、自宛USBﾊﾟｹｯﾄ(USBｱﾄﾞﾚｽ)判定、ﾊﾟｹｯﾄ認証、USBﾎｽﾄへの応答送信を行う送受信の主部分です。根本的にUSBｴﾝｼﾞﾝの心臓部です。
4.4.4.	MyNewUSBAddress:	USBｱﾄﾞﾚｽの変更を示す要求があった場合にINT0受信ﾙｰﾁﾝから呼ばれます。ｱﾄﾞﾚｽは変更され、USBﾊﾟｹｯﾄ受信中の最速符号化解除用にｱﾄﾞﾚｽはNRZI符号化されます。
4.4.5.	FinishReceiving:	USB受信ﾊﾟｹｯﾄから符号化された生ﾃﾞｰﾀを(NRZIとﾋﾞｯﾄ挿入の)復号を行うﾊﾟｹｯﾄに複写します。
4.4.6.	USBreset:	USBを(電源投入後の状態として)既定値に初期化します。
4.4.7.	SendPreparedUSBAnswer:	用意された出力緩衝部内容をUSB信号線へ送信します。送信中にNRZI符号化とﾋﾞｯﾄ挿入が行われます。ﾊﾟｹｯﾄはEOPで終了されます。
4.4.8.	ToggleDATAPID:	ﾊﾟｹｯﾄ識別子(PID) DATAPIDをDATA0とDATA1間で切り替えます。この切り替えはUSB仕様により、送信間中に必要です。
4.4.9.	ComposeZeroDATA1PIDAnswer:	送信用ｾﾞﾛ応答作成。ｾﾞﾛ応答はﾃﾞｰﾀを含まず、装置で利用できる付加ﾃﾞｰﾀがない時の応答としていくつかの場合に使用されます。
4.4.10.	InitACKBuffer:	ACKﾃﾞｰﾀでのRAM内緩衝部初期化(ACKﾊﾝﾄﾞｼｪｰｸ ﾊﾟｹｯﾄ)。この緩衝部はﾒﾓﾘ内で常に準備可状態で維持されるため、応答として繰り返し送られます。
4.4.11.	SendACK:	ACKﾊﾟｹｯﾄのUSB信号線への送信。
4.4.12.	InitNAKBuffer:	NAKﾃﾞｰﾀでのRAM内緩衝部初期化(NAKﾊﾝﾄﾞｼｪｰｸ ﾊﾟｹｯﾄ)。この緩衝部はﾒﾓﾘ内で常に準備可状態で維持されるため、応答として繰り返し送られます。
4.4.13.	SendNAK:	NAKﾊﾟｹｯﾄのUSB信号線への送信。
4.4.14.	ComposeSTALL:	STALLﾃﾞｰﾀでのRAM内緩衝部初期化(STALLﾊﾝﾄﾞｼｪｰｸ ﾊﾟｹｯﾄ)。この緩衝部はﾒﾓﾘ内で常に準備可状態で維持されるため、応答として繰り返し送られます。
4.4.15.	DecodeNRZI:	NRZI復号実行。USB信号線からの緩衝部内ﾃﾞｰﾀはNRZIで符号化されています。本ﾙｰﾁﾝはこのﾃﾞｰﾀからNRZI符号化を解除します。
4.4.16.	BitStuff:	USB受信ﾃﾞｰﾀ内のﾋﾞｯﾄ挿入の追加/削除。ﾋﾞｯﾄ挿入はﾃﾞｰﾀ採取の同期を保証するためにUSB仕様に従ってﾎｽﾄ ﾊｰﾄﾞｳｪｱによって追加されます。本ﾙｰﾁﾝはﾋﾞｯﾄ挿入を除いた受信ﾃﾞｰﾀまたはﾋﾞｯﾄ挿入を施した送信ﾃﾞｰﾀを生成します。
4.4.17.	ShiftInsertBuffer:	ﾋﾞｯﾄ挿入追加実行時に使用する補助ﾙｰﾁﾝ。出力ﾃﾞｰﾀ緩衝部に1ﾋﾞｯﾄを追加し、従って緩衝部長を増します。緩衝部内の以降は後ろへ移動されます。
4.4.18.	ShiftDeleteBuffer:	ﾋﾞｯﾄ挿入削除実行時に使用する補助ﾙｰﾁﾝ。出力ﾃﾞｰﾀ緩衝部から1ﾋﾞｯﾄを削除し、従って緩衝部長を減らします。緩衝部内の以降は前へ詰められます。
4.4.19.	MirrorInBufferBytes:	ﾃﾞｰﾀがUSB信号線から緩衝部へ(LSB/MSB)逆順で受信されるため、ﾊﾞｲﾄ内ﾋﾞｯﾄ順変換。
4.4.20.	CheckCRCIn:	受信ﾃﾞｰﾀ ﾊﾟｹｯﾄのCRC(巡回冗長検査)実行。CRCは不正ﾃﾞｰﾀ検出のため、USBﾊﾟｹｯﾄに付加されます。
4.4.21.	AddCRCOut:	出力ﾃﾞｰﾀ ﾊﾟｹｯﾄへのCRC領域付加。CRCはUSB仕様に従って計算され、CRC領域に与えられます。
4.4.22.	CheckCRC:	CRC検査と追加で使用される補助ﾙｰﾁﾝ。

以下、次頁へ続く。

4.4.23. LoadDescriptorFromROM: ROMからUSB出力緩衝部へ(USB応答として)ﾃﾞｰﾀ取得/設定。

4.4.24. LoadDescriptorFromROMZeroInsert: ROMからUSB出力緩衝部へ(USB応答として)ﾃﾞｰﾀ取得/設定、けれども全偶数ﾊﾞｲﾄは$00として追加されます。これはUNICODE形式文字列種別要求時に使用されます(ROM保存)。

4.4.25. LoadDescriptorFromSRAM: RAMからUSB出力緩衝部へ(USB応答として)ﾃﾞｰﾀ取得/設定。

4.4.26. LoadDescriptorFromEEPROM: ﾃﾞｰﾀ用EEPROMからUSB出力緩衝部へ(USB応答として)ﾃﾞｰﾀ取得/設定。

4.4.27. Load～Descriptor: 応答元場所選択(～はROM, RAM, EEPROM)。

4.4.28. PrepareUSBOutAnswer: USBﾎｽﾄによる要求に従って出力緩衝部にUSB応答を準備、要求動作実行、応答へのﾋﾞｯﾄ挿入。

4.4.29. PrepareUSBAnswer: 要求動作実行と対応応答準備用の主ルーチン。このルーチンは最初に受信した入力ﾃﾞｰﾀ ﾊﾟｹｯﾄから機能番号を見つけることで実行する動作を判定し、そして要求機能を実行します。

これらのルーチンは2種類に分けられます。

- **標準要求**
- **供給者特殊要求**

標準要求は必須でUSB仕様内に記載されます(SET_ADDRESS,GET_DESCRIPTOR等)。

供給者特殊要求は(制御(Control)USB転送で)供給者特殊ﾃﾞｰﾀを得られる要求です。制御入力(Contol IN)USB転送はﾎｽﾄとの通信のために本AVR装置に対して使用されます。開発者は、ここに自分の機能を追加でき、この方法は装置を多方面に拡張します。ｿｰｽ ｺｰﾄﾞ内に記載された様々な組み込み機能は独自機能追加方法の雛形として使用できます。

4.5. 標準USB機能 (標準要求)

ComposeGET_STATUS
ComposeCLEAR_FEATURE
ComposeSET_FEATURE
ComposeSET_ADDRESS
ComposeGET_DESCRIPTOR
ComposeSET_DESCRIPTOR
ComposeGET_CONFIGURATION
ComposeSET_CONFIGURATION
ComposeGET_INTERFACE
ComposeSET_INTERFACE
ComposeSYNCH_FRAME

4.6. 供給者USB機能 (供給者要求)

DoSetInfraBufferEmpty
DoGetInfraCode
DoSetDataPortDirection
DoGetDataPortDirection
DoSetOutDataPort
DoGetOutDataPort
DoGetInDataPort
DoEEPROMRead
DoEEPROMWrite
DoRS232Send
DoRS232Read
DoSetRS232Baud
DoGetRS232Baud
DoGetRS232Buffer
DoSetRS232DataBits
DoGetRS232DataBits
DoSetRS232Parity
DoGetRS232Parity
DoSetRS232StopBits
DoGetRS232StopBits

4.7. ﾃﾞｰﾀ構造 (USB記述種別/文字列)

DeviceDescriptor
ConfigDescriptor
LangIDStringDescriptor
VendorStringDescriptor
DevNameStringDescriptor

4.8. USBホストからの入力メッセージ形式

前記の状態で本USB装置はUSB制御(Control)転送を使用します。この転送種別は参考文献②の13頁(Control Transfers)(**訳補**:USBspcs.pdfの「制御転送」)に記載されるUSB仕様で定義されたデータ形式を使用します。本資料はその詳細と制御(Control)転送がどう動くかを説明し、従って本USB装置がUSBホストとどう通信するかも記述します。AVRデバイスは制御入力エンドポイント(Control IN endpoint)を使用します。データ通信の良い例は参考文献②の15頁(**訳補**:USBspcs.pdfの「制御転送での大量データ」)で得られます。ホストとAVRデバイス間の通信はこの例に従って行われます。

実際の制御(Control)転送に加えて、転送でのDATA0/1領域の形式が検討されます。制御(Control)転送は8バイト長の標準要求での設定(Setup)段階で定義します。この形式は参考文献②の26頁(The Setup Packet)(**訳補**:USBspcs.pdfの「設定パケット」)に記載されています。これらは全バイトの意味を記述した表です。以下は本装置の目的に対して重要です。

標準設定(Setup)パケットは電源ON後の装置の検出と設定に対して使用されます。このパケットは要求種別(bmRequestType)領域で標準種別要求(ビット6,5=00)を使用します。全ての後続(bRequest, wValue, wIndex, wLength)領域の意味はUSB仕様内で得られます。それらの説明は参考文献②の27～30頁(Standard Requests)(**訳補**:USBspcs.pdfの「標準要求」)で得られます。

全ての設定(Setup)パケットは**表1.**に記載されたように使用される8バイトを持ちます。

表1. 標準設定(Setup)パケット領域 (制御(Contorol)転送)

オフセット	領域名	バイト数	値種別	意味		
0	bmRequestType	1	ビット割数値	要求の特性種別		
				ビット7　データ転送方向	0	ホスト⇒装置
					1	装置⇒ホスト
				ビット6,5　種別	0	標準(Standard)
					1	クラス(Class)
					2	供給者(Vendor)
					3	予約
				ビット4～0　受け取り部	0	装置(Device)
					1	インターフェース(Interface)
					2	エンドポイント(Endpoint)
					3	その他
					4～31	予約
1	bRequest	1	値	要求指定		
2	wValue	2	値	要求に従って変化する語領域		
4	wIndex	2	指標または相対位置	要求に従って変化する語領域 (通常、指標または相対位置の進行に使用)		
6	wLength	2	数値	データ状態の場合、転送バイト数		

表2. 標準装置(Standard device)要求

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 00 00000 0 00 00001 0 00 00010	CLEAR_FEATURE	機能選択子	0 インターフェース番号 エンドポイント番号	0	なし
1 00 00000	GET_CONFIGURATION	0	0	1	設定値
1 00 00000	GET_DESCRIPTOR	記述形式/指標	0または記述種別	記述長	記述内容
1 00 00001	GET_INTERFACE	0	インターフェース番号	1	交替インターフェース番号
1 00 00000 1 00 00001 1 00 00010	GET_STATUS	0	0 インターフェース番号 エンドポイント番号	2	装置, インターフェース またはエンドポイント の状態
0 00 00000	SET_ADDRESS	装置アドレス	0	0	なし
0 00 00000	SET_CONFIGURATION	設定値	0	0	なし
0 00 00000	SET_DESCRIPTOR	記述形式/指標	0または記述種別	記述長	記述内容
0 00 00000 0 00 00001 0 00 00010	SET_FEATURE	機能選択子	0 インターフェース番号 エンドポイント番号	0	なし
0 00 00001	SET_INTERFACE	交替設定値	インターフェース番号	0	なし
1 00 00010	SYNCH_FRAME	0	エンドポイント番号	2	フレーム番号

表3. 関数呼び出しとして使用される供給者装置(Vendor device)要求

bmRequestType	bRequest		wValue (第1ﾊﾟﾗﾒｰﾀ)	wIndex (第2ﾊﾟﾗﾒｰﾀ)	wLength	ﾃﾞｰﾀ
	関数名 (注)	番号				
1 10 xxxxx	SetInfraBufferEmpty	1	なし	なし	1	状態値
1 10 xxxxx	GetInfraCode	2	なし	なし	1	状態値
1 10 xxxxx	SetDataPortDirection	3	DDRB, DDRC	DDRD, 使用ﾎﾟｰﾄ	1	状態値
1 10 xxxxx	GetDataPortDirection	4	なし	なし	3	DDRB, DDRC, DDRD
1 10 xxxxx	SetOutDataPort	5	PORTB, PORTC	PORTD, 使用ﾎﾟｰﾄ	1	状態値
1 10 xxxxx	GetOutDataPort	6	なし	なし	3	PORTB, PORTC, PORTD
1 10 xxxxx	GetInDataPort	7	なし	なし	3	PINB, PINC, PIND
1 10 xxxxx	EEPROMRead	8	ｱﾄﾞﾚｽ	なし	ﾊﾞｲﾄ長	EEPROM読み出し値
1 10 xxxxx	EEPROMWrite	9	ｱﾄﾞﾚｽ	書き込み値	1	状態値
1 10 xxxxx	RS232Send	10	送信値	なし	1	状態値
1 10 xxxxx	RS232Read	11	なし	なし	2	状態値
1 10 xxxxx	SetRS232Baud	12	ﾎﾞｰﾚｰﾄ下位	ﾎﾞｰﾚｰﾄ上位	1	状態値
1 10 xxxxx	GetRS232Baud	13	なし	なし	2	ﾎﾞｰﾚｰﾄ値
1 10 xxxxx	GetRS232Buffer	14	なし	なし	ﾊﾞｲﾄ長	FIFO内の受信ﾊﾞｲﾄ
1 10 xxxxx	SetRS232DataBits	15	ﾃﾞｰﾀ ﾋﾞｯﾄ数	なし	1	状態値
1 10 xxxxx	GetRS232DataBits	16	なし	なし	1	ﾃﾞｰﾀ ﾋﾞｯﾄ数
1 10 xxxxx	SetRS232Parity	17	ﾊﾟﾘﾃｨ指定値	なし	1	状態値
1 10 xxxxx	GetRS232Parity	18	なし	なし	1	ﾊﾟﾘﾃｨ指定値
1 10 xxxxx	SetRS232StopBits	19	ｽﾄｯﾌﾟ ﾋﾞｯﾄ数	なし	1	状態値
1 10 xxxxx	GetRS232StopBits	20	なし	なし	1	停止ﾋﾞｯﾄ数

注: 実際の関数名は表内の関数名に先行してFNCNumberDoが必要です。

制御(Control)転送動作はﾌｧｰﾑｳｪｱ内に独自機能として実装した使用者通信用に使用されます。要求種別(bmRequestType)領域で供給者種別要求(ﾋﾞｯﾄ6,5=10)が使用されます。これに続く全ての領域(bRequest, wValue, wIndex)はﾌﾟﾛｸﾞﾗﾏの目的に従って変更できます。本実装でbRequest領域は機能番号、次の領域が機能ﾊﾟﾗﾒｰﾀに使用されます。最初のﾊﾟﾗﾒｰﾀがwValueで、第2ﾊﾟﾗﾒｰﾀがwIndexです。

実装例はEEPROM書き込みです。機能番号としてbRequest=9が選ばれます。wValue領域はEEPROMｱﾄﾞﾚｽに使用され、書き込み値(EEPROMﾃﾞｰﾀ)はwIndex領域です。これに従って次の関数、EEPROMWrite(Address,Value) を得ます。

多くの使用者関数が必要とされる場合、それはﾌｧｰﾑｳｪｱ内に必要とした機能の本体と機能番号を追加することで足ります。この技法はﾌｧｰﾑｳｪｱ内の組み込み機能から得ることができます(ｿｰｽ ｺｰﾄﾞをご覧ください)。

USBﾎｽﾄも制御入力(Control IN)転送で装置と通信します。ﾎｽﾄが上記で定義した(機能番号とﾊﾟﾗﾒｰﾀ)形式で8ﾊﾞｲﾄの入力ﾃﾞｰﾀ(IN data)ﾊﾟｹｯﾄを装置へ送り、その後に装置が要求されたﾃﾞｰﾀで応答します。応答ﾃﾞｰﾀ長はいくつかの場合でﾌｧｰﾑｳｪｱによって255ﾊﾞｲﾄまでに制限されますが、主な制限はﾎｽﾄ ｺﾝﾋﾟｭｰﾀ上のﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞ側にあります。現在のﾄﾞﾗｲﾊﾞは供給者種別要求で8ﾊﾞｲﾄ長応答を支援します。

4.9. ﾌｧｰﾑｳｪｱの独自化

使用者はﾌｧｰﾑｳｪｱ内に新機能を追加し、装置機能を拡張できます。ﾌｧｰﾑｳｪｱ内に使用者機能を追加する方法の例が3つあります(DoUserFunctionN(N=0,1,2))。同様に拡張した機能を解かるためにこれらの例を見てください。機能内容は装置の必要条件だけに依存します。

ｺﾝﾋﾟｭｰﾀ側で示される識別名と装置名はﾌｧｰﾑｳｪｱ内で変更できます。この名前は文字列としてﾌｧｰﾑｳｪｱ内に配置され、どの文字列も変更できます。けれどもこれらの名前は対象ｼｽﾃﾑでの正しい認証のため、USB製品識別(PID)と供給者識別(VID)と共に変更することが推奨されます。

VIDとPIDは共に装置種別を与えるために固有でなければなりません。従って装置機能が変更された場合、PIDとVID、またはどちらかも変更されるべきであることが推奨されます。供給者識別(VID)はUSB装置供給者(社)に対応し、USB協会から割り当てられなければなりません(参考①でより多くの情報をご覧ください)。全供給者(社)は自身の識別(ID)を持ち、この値はどの未割り当て値にも変更できません。しかし、製品識別(PID)は供給者(社)の選択だけに依存し、このPIDの目的は同一供給者(社)からの異なる装置を認知することです。

本応用記述はATMELのVID($03EB)とPID($21FF)で設定されます。利用者の対象ｼｽﾃﾑで、このVIDを使用してはなりません。

4.10. PCソフトウェア

本装置と通信するために、PC側でいくつかのソフトウェア支援が必要です。このソフトウェアは3つの段階に分けられます。

1. デバイス ドライバ：オペレーティング システム(Windows98/ME/NT/XP)内へのインストール用で、装置との下層通信に使用。
2. DLLライブラリ：デバイスドライバとの通信と装置機能のカプセル化に使用。本DLLは使用者の応用プログラムからの装置機能アクセスを簡易化します。それはいくつかの装置とオペレーティング システムに関連する機能(スレッド，緩衝部等)を含みます。
3. 使用者応用プログラム：使用者と装置間の友好的な通信のための使用者インターフェースの作成。DLLライブラリの関数呼び出しの使用だけです。

4.10.1. デバイス ドライバとインストール ファイル

コンピュータのUSBポートにUSB装置を初めて接続すると、オペレーティング システムはその装置を検知し、ドライバ ファイルを要求します。これはデバイスのインストールと呼ばれます。このインストール処理に対してデバイス ドライバを作成する必要はありませんが、インストール手順が記述されたインストール スクリプトが必要です。

本資料内に記述した装置用のデバイス ドライバはWindows2000 DDK(Driver Development Kit)で作成されています。このUSBドライバの開発はDDKに含まれる例の1つ、IsoUsbが元です。本ドライバは本資料の目的(AVR USB装置通信)用に修正されています。本装置がIOCTLを通してコンピュータと通信するため、元のソース コードでIOCTL通信についての部分が拡張/追加されています。ドライバのコード容量を減らすために未使用部分(読み書きルーチン)が削除されています。このドライバ名はAVR309.sysで、USB装置へ(制御入力(Control IN)転送)命令の送り主として動作します。このドライバはWindows95を除く全ての32ビット版Windowsで動作します。

INFファイル内に書かれたインストール スクリプトはデバイスのインストール中に使用されます。このINFファイルは様々なインストール段階が記述されます。AVR309.infファイルはテキスト エディタを使用して作成されました。このファイルはインストール中にオペレーティング システムにより要求されます。インストール処理中、ドライバ ファイルがシステム内に複写され、必要なシステム変更が行われます。INFファイルは様々な応用プログラムからの容易な到達のために、システム検索経路へのDLLライブラリ インストールを保証します。

装置(デバイス)のインストールに対し、INFファイル(AVR309.inf), ドライバ(AVR309.sys), DLLライブラリ(AVR309.dll)の3つのファイルが必要です。

4.10.2. DLLライブラリ

DLLライブラリはデバイス ドライバと通信し、装置の全機能がこのライブラリに実装されます。最終使用者用のこの応用プログラミング法は簡易化されます。本DLLライブラリはRS232データ受信用のシステム緩衝部を含み、装置への排他アクセス(装置アクセスの直列化)を保証し、装置のRS232データ緩衝部読み込み用の単一システム スレッドを作成します。

DLLでの直列化は与えられたどの時間に対しても1つの応用プログラム/スレッドだけが装置と通信することを保証します。これは同時に多数の応用プログラムからの問い合わせと応答が混ざることを可能とするために必要です。

RS232データ受信用システム緩衝部は装置のRS232信号線から受信したデータが全ての応用プログラムで共通の1つの緩衝部内に格納されることを保証します。他の或る応用プログラムがデータ内のいくつかを先に読むために、応用プログラムが不完全なデータを受信することの危険はありません。

全ての応用プログラムに対して1つのシステム スレッドだけが存在し、周期的に装置へRS232データについて要求します。そしてスレッドは受信したデータをシステム緩衝部内に格納します。1つのシステム緩衝部だけの方法は(全ての応用プログラムがそれら自身のスレッドを持つのに比較して)少ないCPU使用とシステム緩衝部内への容易なデータ保存を保証します。

装置の全機能はDLLライブラリ内で定義され、それらは機能番号とパラメータとしてではなく、パラメータとの整然とした関数名として使用者に好ましい形で表されます。RS232緩衝部データ読み込み関数のように、いくつかの関数は内部的に多数の複合体です。この方法で最終使用者応用プログラムの開発者はDLLインターフェースだけを使用して応用プログラムを素早く書けます。DLLライブラリがハードウェア レベルから応用プログラマを分離するように、それらは下層装置機能を学ぶ必要がありません。

宣言はBorland Delphi, (BorlandまたはMicrosoft) C++, Visual Basicの主に使用される3つのプログラミング言語について書かれています。これらの関数の詳細記述はAVR309_DLL_help.htmヘルプ ファイル内で得られます。

このDLLはDelphiで書かれ、全ソース コードは(**訳注**:著者サイトの)本応用記述内に含まれます。

4.10.3. 最終使用者応用プログラム

最終使用者応用プログラムは装置との通信にDLLライブラリからの機能だけを使用するでしょう。その主目的は使用者に友好的なGUI (Graphical User Interface)を作ることです。

応用プログラマはそれら自身の応用を書くのにDLLライブラリを使用します。例は全ソース コードが利用可能な公表したプロジェクト内で得られます。多くの応用プログラムは開始点としてこの例を使用して、それぞれのプログラミング言語(Delphi, C++, Visual Basic)で書くことができます。

それらにはAVR309demo.exeと呼ばれる最終使用者応用プログラムの例が含まれます。このソフトウェアはDLLライブラリからの機能の使用法の例として示されるだけです。含まれたソース コードはDelphiで書かれ、他の応用プログラム用の雛形として使用できます。

4.10.4. UART速度誤差の検討

このマイクロ コントローラはUSB採取のために12MHzクロックを使用します。しかし、このクロック値の使用はボーレート生成が標準ボーレートに対して小さな誤差を含むことで小さな劣性を持ちます。けれども高クロック値はこの誤差を小さくします。1バイトの最大ビット数は1(開始ビット)+8(データ ビット)+1(パリティ ビット)+2(停止ビット)=12ビットで、最大誤差がビット幅の半分(1/2)となるため、ボーレート生成で受け入れ可能な絶対最大誤差は概ね4%です。この時の誤差は1/2÷12×100(%)=4.1%です。

DLL内の関数はこの誤差を自動的に検査し、誤差が4%以下の場合にだけマイクロ コントローラのボーレートを設定します(未支援ボーレートの場合は誤りメッセージを返します)。けれども2%よりも大きな誤差で決して使用しないことが推奨されます。

表4.は12MHzクロック使用時の標準ボーレート誤差を一覧で示します。

表4. AVR UARTボーレート誤差 (12MHzクロック)

標準ボーレート	AVRでのボーレート	誤差 (%)
600	602	+0.33
1200	1204	+0.33
2400	2408	+0.33
4800	4808	+0.17
9600	9616	+0.17
19200	19230	+0.16
28800	28846	+0.16
38400	38462	+0.16
57600	57692	+0.16
115200	115384	+0.16

5. 参考文献と資料

5.1. USB関連資料:

① http://www.usb.org ・・・ USB仕様とその他のUSB関連資料
② http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf ・・・ USB動作の簡単で大変良い文書。
③ http://www.beyondlogic.org ・・・ USB関連資料
④ enumeration.pdf ・・・ USB動作一覧の図解
⑤ http://mes.loyola.edu/faculty/phs/usb1.html
⑥ http://www.mcu.cz ・・・ チェコ/スロバキア語でのUSB項目
⑦ crcdes.pdf ・・・ USBでのCRC実装
⑧ USBspec1-1.pdf ・・・ USB1.1仕様書
⑨ usb_20.pdf ・・・ USB2.0仕様書

5.2. AVR関連資料:

⑩ http://www.atmel.com/AVR ・・・ AVR 8ビット マイクロ コンピュータ系統
⑪ doc2543.pdf ・・・ ATtiny2313データシート
http://www.atmel.com/dyn/products/product_card.asp?part_id=3229
⑫ doc2545.pdf ・・・ ATmega48/88/168データシート
http://www.atmel.com/dyn/products/product_card.asp?part_id=3301
⑬ avr910.pdf ・・・ AVR ISPプログラミング
⑭ http://www.avrfreaks.com ・・・ 多くのAVR資料と情報
⑮ AVR Studio 4 ・・・ AVR系統用デバッグ ツール (http:/www.atmel.comから)
⑯ http://www.hw.cz/products/lpt_isp_prog/index.html ・・・ 簡単なLPT ISP書き込み器

5.3. ドライバ関連資料:

⑰ http://www.beyondlogic.org ・・・ USBドライバについてのUSB関連資料
⑱ http://www.cypress.com ・・・ 充分に記述されたUSBドライバ (USB温度計用)
⑲ http://www.jungo.com ・・・ WinDriverとKernelDriver (簡単なUSBドライバ使用)
⑳ http://micrsoft.com Microsoft Windows DDK(Driver Development Kit) ・・・ ドライバ書き用ツール

6. 追補

追補は(**訳注**:著者サイトの)本応用記述に個別の付随ファイルで含まれます。

6.1. ソース コード付き AVRファームウェア

ATmega8 AVR マイクロ コントローラ用のファームウェア ソース コードはAVR Studio 4で書かれました。ソース コードはUSBtoRS232.asmテキスト ファイルまたはUSBtoRS232asm.pdf書式強調形式ファイルで得られます。

6.2. ソース コード付き DLLライブラリ

AVR309.dllライブラリはDelphi3で書かれ、故にソース コードはPascalオブジェクト言語が基準とされます。AVR309.dllへの(Delphi, C/C++, Visual Basic)インターフェース(外部関数)は、AVR309_DLL_help.htmファイル内で記述されます。Delphi3で書かれたAVR309.dllライブラリの完全なソース コードはAVR309.dpr(Delphi3 プロジェクト)で得られます。

6.3. ソース コード付き 最終使用者応用例

最終使用者応用プログラムとしてのDLLライブラリ使用例はAVR309USBdemo.exeです。このソース コードはAVR 309demo.dpr Delphi3応用プログラムです。

7. 著者について

Ing. Igor Cesko Slovakia www.cesko.host.sk cesko@internet.sk

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
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
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

本応用記述はATMELのAVR309応用記述(doc2556.pdf Rev.2556B-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訳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。