

## AVR309補足

---



Rev. 1.0C-05/21

## 目 次

---

はじめに .....	3
本書の注意点 .....	3
USB仕様の基礎知識 .....	4
ビット速度の考察 .....	5
プログラム構造の考察 .....	6

---

ハードウェアの注意点 .....	6
電源電圧 .....	6
クロック周波数誤差 .....	6

---

ファームウェア概説 .....	7
USB送受信 .....	7
送受信に於けるCRC .....	8
UART/USART受信割り込み処理 .....	8

---

付録: ラベル対応表 .....	9
------------------	---

## はじめに

本書はAVR309応用記述に基づいて作成または改造する場合の参考技術情報です。基本的にAVR309応用記述とUSB2.0仕様書間の関係を側面から補足します。

応用記述内にはファームウェアで必要とする最低限のUSB情報と本応用に対する供給者機能(関数)情報が記述されています。この応用記述はATmega8版を基に記載されていますので、AT90S3213版では利用できない機能も含まれています。AT90S3213版はフラッシュメモリ容量の制限のため、いくつかの供給者機能が削除されています。

応用記述内にも記載されていますが、特にAT90S2313は保証範囲外での動作(12MHz/3.3V)になります。然しながら、電気的特性で規定される動作周波数範囲は、同項目で規定される他の条件(例えば温度範囲)内全域にわたる保証範囲であるため、使用者が通常に動作させる条件下では、かなり緩和されます。例えば、通常使用時に於いて、使用者が居続けられる温度範囲であれば、上限動作周波数は電気的特性での上限を越えます。従って、一般的に、本応用(12MHz/3.3V)でのAT90S2313は問題なく動作するでしょう。より厳しい条件下で動作させる場合は、応用記述内にも記載されるように、AT90S2313に代わりATtiny2313、ATmega8に代わりATmega48/88/168の使用が望まれます。

ATmega48/88/168を使う場合、USARTレジスタが拡張I/O領域に配置されているため、本ソース内で使っている標準I/O領域用命令が利用できません。これらはデータシートの「コード例について」記述のように**LDS,STS**命令を基にした命令で書き換える必要があります。この書き換えにより、特にビット操作命令は実行クロック周期が2から5に増加します。更にこの操作用の一時レジスタを必要としますので、場所によってはそのための退避/復帰が必要になるかもしれません。この影響で**同期パターン**検出機会が減少し、再送が増え、状態によっては通信異常となるかもしれません。**「UART/USART受信割り込み処理」**を参照してください。

この場合の対処として、クロック周波数の増加が考えられます。本応用のクロック周波数は12MHzですが、例えばこれを15MHzにします。USB通信の1ビット時間は12MHzで8クロック周期ですが、15MHzでは10クロック周期になり、この問題が緩和されます。但し、クロック周波数を15MHzにする場合、ATmega48/88/168の安全動作領域とするには動作電圧を3.6V以上にしなければなりません。けれども、USB信号規約から3.6Vを越えることはできませんので、必然的に3.6Vで動作させることになります。**「ビット速度の考察」**と**「ハードウェアの注意点」**の**「動作電圧」**を参照してください。

## 本書の注意点

本書内のラベル名(“〜”で表記)は日本語版ソースでの名称です。本書の記述はソースの説明に対して必要最低限のものです。USB仕様関係の詳細については添付の**USBspecs.PDF**をご覧ください。

本書内の青文字で示されるリンクは同梱の**AVR309.PDF**と**USBspecs.PDF**も参照しますので、これらのPDFファイルを本書と同一のフォルダに置いてください。

## USB仕様の基礎知識

本応用では「**ビット速度の考察**」で記述されるように、USB2.0規約の**Low-Speed**(1.5Mbps)動作だけが支援可能です。従って本応用を利用するには基本的にUSB2.0規約の**Low-Speed**動作に関する部分を理解する必要があります。とはいえ、本応用では最小限のファームウェアでの実現を行なうために、最低実装部分である**制御(Control)転送**だけを使うので、**Low-Speed**動作に関する部分全てを理解する必要はありません。この状態での通信方法は以下のようになります。

- USB規約では必ずホストからの要求により装置が応答する形式で、装置が一方的に送信を行うことはありません。Low-Speed装置での最小送受信単位はパケットです。パケットには次の3種類があり、連続するこの3種類が1つの**処理単位(Transaction)**を構成します。

- ① 指示票(Token) …… 次のデータ(Data)種別を規定
- ② データ(Data) …… 指示票の指定による要求内容やデータ内容
- ③ ハンドシェイク(Handshake) …… 上記要求内容やデータ内容に対する(可否)応答

各パケットの構成は以下で示されます。以下の各要素末尾の単位記述のない( )内は元データでのビット数です。**ビット挿入(0挿入)**が行なわれるため、バス上の信号では、このビット数かそれ以上のビット数になります。

- ① 指示票(Token)パケットは次の構成です。

SYNC (8)	PID (8)	アドレス (7)	エンドポイント (4)	CRC5 (5)	EOP (3以上)
----------	---------	----------	-------------	----------	-----------

パケット種別(PID)は**SETUP\_PID**、**IN\_PID**または**OUT\_PID**の何れかです。本応用では既定エンドポイント0だけを扱うので、エンドポイント領域は無視されます。また受信時のCRC5検査も省略されています。

- ② データ(Data)パケットは次の構成です。

SYNC (8)	PID (8)	データ (最大8バイト)	CRC16 (2バイト)	EOP (3以上)
----------	---------	--------------	--------------	-----------

PIDは**DATA0\_PID**または**DATA1\_PID**です。CRC16は最大8バイトのデータ部のみが対象です。本応用では受信時のCRC16検査が省略されています。

- ③ ハンドシェイク(Handshake)パケットは次の構成です。

SYNC (8)	PID (8)	EOP (3以上)
----------	---------	-----------

PIDは**ACK\_PID**、**NAK\_PID**または**STALL\_PID**の何れかです。

- 制御(Control)転送は次の3段階で実行されます。

- ① 設定(Setup)段階 …… SETUP\_PIDの指示票パケットで始まる処理単位(Transaction)
- ② データ(Data)段階 …… 設定(Setup)段階の内容によりIN/OUT\_PIDの指示票パケットで始まる処理単位(Transaction)
- ③ 状態(Status)段階 …… 設定(Setup)段階の内容によりOUT/IN\_PIDの指示票パケットで始まる処理単位(Transaction)

これら3段階で1つの転送になります。②の**データ(Data)段階**は転送データ容量に応じて複数の処理単位(Transaction)に成り得、①の**設定(Setup)段階**内のデータ(Data)パケット内容(bmRequestTypeのビット7)に従ってIN\_PIDまたはOUT\_PIDになります。③の**状態(Status)段階**でのIN\_PIDまたはOUT\_PIDは、この②の方向と逆になります。1転送の例を次に示します。

### ■ 実データをホストから装置へ転送する場合

	ホスト	装置
設定(Setup)段階	→ SETUP_PIDパケット	→
	→ DATA0_PIDパケット	→
	← ACK/NAK_PIDパケット	←
データ(Data)段階	→ OUT_PIDパケット	→
	→ DATA0/1_PIDパケット	→
	← ACK/NAK_PIDパケット	←
状態(Status)段階	→ IN_PIDパケット	→
	← DATA1_PIDパケット	←
	→ ACK/NAK_PIDパケット	→

### ■ 実データを装置からホストへ転送する場合

	ホスト	装置
設定(Setup)段階	→ SETUP_PIDパケット	→
	→ DATA0_PIDパケット	→
	← ACK/NAK_PIDパケット	←
データ(Data)段階	→ IN_PIDパケット	→
	← DATA0/1_PIDパケット	←
	→ ACK/NAK_PIDパケット	→
状態(Status)段階	→ OUT_PIDパケット	→
	→ DATA1_PIDパケット	→
	← ACK/NAK_PIDパケット	←

データ(Data)段階が複数処理単位(Transaction)で構成される場合、基本的にその転送に於けるデータの終了は実データが存在しない処理単位(Transaction)で示されます。これはその処理単位(Transaction)内のデータ(Data)段階(DATA0/1\_PIDパケット)内の実データ部が存在しないことを意味します(即ちSYNC+DATA0/1\_PID+CRC16+EOP形式)。

本応用では制御(Control)転送だけを使うので、この場合のDATA0/1\_PIDは設定(Setup)段階でDATA0\_PID、状態(Status)段階でDATA1\_PIDが常に使われます。データ(Data)段階ではDATA1\_PIDが使われますが、複数処理単位(Transaction)の場合にはその処理単位(Transaction)毎に以降、DATA0\_PIDとDATA1\_PIDを交互に使用します。

## ビット速度の考察

基本的にマイクロコントローラでハードウェアの支援なしにソフトウェアのみでUSB動作の実装を考慮すると、少なくともUSB通信の1ビット時間内でCPUは数クロック以上の命令実行が出来なければなりません。このクロック数がUSBバス上に送られてくる信号に対する同期化時の初期誤差を決定します。例えばUSB通信の1ビット時間を8倍の速度で同期化採取する場合を考えます。

右の図は1ビット時間と採取間隔との両端での採取状況を示します。例①の場合は緑位置の採取が丁度1ビット時間開始直後だった場合、例②はその位置が開始直前で次の採取位置からの認識となった場合を表します。この2例が1ビット時間開始認識に対する絶対的な初期誤差になります。この場合の初期誤差は1/8(12.5%)です。

理想的なビット値採取は1ビット時間の中心であるべきです。そのためには1ビット時間に対する採取数が奇数倍でなければなりません。本例のように偶数倍の場合、中心採取位置に対する誤差は初期誤差に加え、採取間隔の半分の時間に対する時間誤差も考慮しなければなりません。

本応用ではソフトウェアのみでこの同期を行わなければなりませんので、これには次例のように最低、該当ポートビットの検査命令と待機繰り返し用分岐命令が必要になります。

```
SYN1:    SBIS      PINB,0    ; DATA-=1(DATA+=0)でスキップ
          RJMP      SYN1      ; DATA-=1(DATA+=0)まで待機
```

このループは3クロック周期で1周し、この1周が採取間隔になります。本応用はシステムクロックが12MHzですので1.5Mbpsの1ビット時間は8システムクロックになります。従って初期誤差は3/8になり、これが最大初期誤差です。逆に考えると、USBの1ビット時間がCPUの3システムクロック以下では同期化が不可能であると言えます。けれども、この誤差は必ず実際の変化点に対して内部的な遅れになります。このため、変化点検出からビット中央までのクロック数を-1することで最大初期誤差を2/8(-2/8~+1/8)にできます。

上記は送受信側での基準周波数が完全に一致している場合です。実際には若干の誤差が存在します。この誤差は信号上での各エッジ毎に再同期処理を行わなければ、ビット数の増加に伴い、誤差が累積して行きます。従って或る時点の概略誤差は次式で表されます。

誤差 = 初期誤差 + 同期後のビット数 × 送信側/受信側での基準周波数誤差

一般的なUSB制御ICなどのようにハードウェアで処理する形態の場合は、信号上での各エッジ毎に再同期処理が行われます。規約での「ビット挿入」処理により、最長エッジ間隔は7ビット時間になりますので、上記の累積計算でのビット数は最大でも7になり、誤差の無限的な累積を防止しています。しかし、ソフトウェアのみで処理を行う場合、再同期処理用の端検出とビット値採取を同時に行わなければならないこととなりますので、本応用では実現不可能です。従ってソフトウェアのみで処理を行う場合はEOPまでの全ビットに対して上式が適用され、誤差が累積しますので、長ビット列通信に問題があることが判ります。

送信時は単にUSBの1ビット時間間隔で送信するだけです。これらは受信に対する考慮です。

これらを念頭に置いた上での考察を試みます。

USB(USB2.0)規約には以下の転送速度種別があります。

■ High-Speed 480Mbps    ■ Full-Speed 12Mbps    ■ Low-Speed 1.5Mbps

ソフトウェアのみでの処理を考えた場合、High-Speedは勿論無理で、Full-Speedの場合も何らかのハードウェア機能の支援なしには不可能なことが判ります。Low-Speedの場合は可能性があります。本応用ではCPUクロックを12MHzとして、これを実現しています。USBの1ビットは12MHz ÷ 1.5MHzでCPUの8実行クロックに相当します。従って最大初期誤差は上記から2/8ビット時間になります。

送信側/受信側での基準周波数誤差の累積問題については、短ビット列通信しか行わないことで対処します。元々Low-Speed装置にあっては長ビット列通信となる転送種別は基本的に存在しません。本応用では必須の標準制御要求転送と供給者定義の供給者制御要求転送のみを用います。これらの転送での連続する最長ビット列は(ビット挿入後)109ビットになります。従って送信側/受信側での基準周波数誤差は109ビット目で2/8ビット時間以内と言う条件になります。これはCPUクロックを元にとすると、 $2 \div (109 \times 8) = \text{約} 0.23\%$ になり、これは一般的な水晶発振子の精度であれば満足するでしょう。

しかし、ここで注意しなければならないのはUSB仕様でのホスト側の許容誤差です。現実には存在しないと思いますが、仮にLow-speed適合のみのホストが存在した場合、その許容誤差は1.5%ですので、既に通信の可能性は著しく低いと言えます。同様にUSB1.1までのホストではFull-speed適合での許容誤差が0.25%のため、通信できたり、できなかつたりになる可能性があります。USB2.0のホストはHigh-speed適合での許容誤差が0.05%なので、例えば実際にはLow-speedでの通信であったとしても、これに準じた許容誤差が期待できますので、おそらくは問題なく通信可能と思われます。



## プログラム構造の考察

これまでの記述から連続するビット列(パケット)はビット採取の正確さが要求されるため、他の割り込みなしで、それ専用に動作しなければならない事が判ります。加えて基本的に1処理単位(Transaction)内は速やかに処理されなければなりませんので、指示票(Token)に対する何らかの応答までは連続的に処理する必要があります。本応用では同期領域(SYNC)の遷移点でINT0を起動することで受信を開始し、その割り込み処理内でこれらを一括処理しています。

RS232受信割り込みを使っているため、この割り込み処理中にUSBのSYNC送信が開始されるかもしれません。その場合の最悪状態はUSBホストでの再送になります。この場合、USB転送に対してRS232受信割り込み間隔が十分に長いため、殆ど実害は起こりませんが、基本的にINT0以外の割り込みは、短周期での使用ができないと考えるべきです。

このINT0割り込み処理が本応用での中核になります。けれども連続する時間以外でも処理可能なものは平タスクで処理することによって、直接的なUSB処理に対するCPU占有を緩和し、他への応用可能範囲の拡大を図っています。例えば制御要求の解析や対応するデータの準備などが平タスクで処理されます。これらの結果、最終的にINT0割り込み処理で実行される最長時間がUSB処理以外の応用に対する制限条件になります。

最大INT0割り込み処理時間については各ソースファイル内の「最大処理クロック数と処理時間」を参考に算出します。各ルーチン題目に(最大処理クロック数:最大処理時間)形式で、そのルーチンでの最大処理クロック数と最大処理時間を記載してあります。またルーチンの呼び出し/分岐の要所にはコメント行右端に(最大処理クロック数)形式で、その呼び出し/分岐命令以降の最大処理クロック数を記載してあります。これらの値が物理的に可能な最大値であることに注意してください。即ち本来、正常動作ならば起こらない場合も含む最大値であり、あくまでも最悪条件計算用であり、一般的にはこれらの値より少ない値となる部分も含まれていることをご承知ください。

本応用はRS232変換器としての応用です。この場合、上記の最大INT0割り込み処理時間がRS232転送速度の上限値を制限します。INT0割り込み発生直後にUART/USARTで1文字受信が完了すると、UARTの場合は次の1文字受信完了時に、USARTの場合は次からの2文字目受信完了時に、UART/USARTの緩衝部溢れが発生し、受信値を失ってしまいます。UARTとUSART間の違いはUARTが単一緩衝、USARTが二重緩衝であることの差です。従って本応用でのRS232最大転送速度は概ねUARTの場合が57600bps、USARTの場合が115200bps程度になります。例えば57600bps、8データビット、パリティなし、1停止ビットの場合、1文字は10ビットで構成されます。従って1文字に対するCPUクロック数は $12\text{MHz} \div (57600\text{bps} \div 10) = \text{約}2083\text{クロック}$ になります。最大INT0割り込み処理時間は概ね2000クロック程度ですので、UARTの場合は57600bpsまで可能と言えます。USARTの場合は二重緩衝のため、この2倍、即ち115200bpsまで可能と言えます。但し、USBホストからのRS232受信データ取得要求間隔が本事例でのRS232受信緩衝部容量分の時間を上回らないとの条件付きです。またRS232送信間隔はUSBホストからのRS232データ送信要求間隔に依存しますので、必ずしも隙間無く連続的な送信が可能とは限りません。

本応用以外の応用を実装する場合の要点を次に示します。

- 基本的にUSBのINT0割り込みを阻害する割り込みは使えません。
- 応用は平タスク内でUSB関連処理以外の空きCPU時間で処理可能でなければなりません。
- 応用は最長INT0割り込み時間を許容できなければなりません。

## ハードウェアの注意点

以下は正常且つ安定な動作を行なうための留意点です。

### 動作電圧

CPUの動作電圧はCPUがUSB信号線を直接駆動するため、USB信号規約によって制限されます。基本的に3.3V中心で最大3.6Vです。これはバス上での規定ですので、直接CPU電源電圧を示すものではありません。負荷によって一般的にポートの出力電圧は若干中心電圧側の電圧になるため、電源電圧幅一杯に振れることはありません。従ってCPU電源電圧はこの規定より若干高めにてできます。

CPUの安定動作のためには可能な限り高い電圧が望まれます。然しながらUSBバスからの最低供給電圧が4.40Vなので、使う3端子電圧調整器はこれを(低降下型などで)満足しなければなりません。

### クロック周波数誤差

CPUはクリスタル発振でなければなりません。セラミック振動子では偶然を除いて正常に動作しないでしょう。これは本応用が同期パターン(SYNC)でのみ同期動作を行い、以降再同期動作を行なえないので、絶対誤差が非常に重要となるためです。また、同様の理由でUSB2.0ホストとの使用が望まれます。USB1.1までのホストでの動作は微妙です。

加えて温度特性にも注意してください。本応用は電源投入直後にUSB通信が行なわれるため、温度特性上、非常に不利です。これは電源投入直後がデバイスにとって一番温度変化が大きいからです。従って絶対誤差が微妙な場合、初回電源投入時のみ認識、または暫く通電後に接続すると認識などの症状になる可能性があります。

これらの詳細に関しては「[ビット速度の考察](#)」をご覧ください。

## ファームウェア概説

本応用は基本的にUSB送受信部、要求処理部とUART/USART受信部に大別されます。要求処理部はUSB受信での要求が起点になりますので、その延長と考えても差し支えありません。UART/USART受信部は割り込み処理で、これがUSB同期のINT0割り込みに関係するため、本応用ではかなり重要な部分です。

## USB送受信

USB受信パケットは“**USBF:**”内に下位バイト側から格納されますが、以下の点に注意してください。

- **SYNC**は格納されず、**PID**から格納されます。
- バイト内のビット順が逆順(MSBが本来のLSB)で格納されます。
- **PID**判定が先行し、その後に端数ビット格納/後処理(“**MVRB:**”)が行われます。

端数ビット格納後、生受信緩衝部(“**USBF:**”)から平タスクで使う受信緩衝部(“**URBF:**”)へ受信データを移動します。これは応用記述内でも記載されるように、処理速度の関係で高速応答が必要な部分までを割り込み処理内で行い、それ以降を平タスクで行うため、これによって“**URBF:**”内の受信パケット処理中に“**USBF:**”内で新規パケットの受信が行えます。

“**USBF:**”から“**URBF:**”への移動(“**MVRB:**”)では以下の点に注意してください。

- **SOP**は“**USBF:**”内に含まれませんので、“**URBF:**”先頭に定数として書かれます。従って“**USBF:**”先頭からは“**URBF:**”+1位置からの移動になります。
- 受信バイト数(“**URBC:**”)はSOPを含みますので、転送は“**USBF:**”内の実データバイト数+1になります。
- 移動後もビット逆順に変更はありません。

INT0割り込み処理内の**PID**(**PID**と**アドレス**)判定に注意してください。これらは**NRZI**復号、**ビット挿入**削除前に判定されます。従って、単に上記のビット逆順だけでなく、**NRZI**化、**ビット挿入**後の**PID**値や**アドレス**値で比較する必要があります。更に**PID**は**DATA**-側で受信しているため、**SYNC**の(バス上の)最終ビット値が**0**、従って**PID**に先行するビット値が**0**になります。このため、比較する**PID**値は規約で表記される値に対して、前記の処置に加えて(**NRZI**化に対応するために)論理反転も必要になります。**PID**値に関する対処は予め **EQU**定義で処置後の値を定義しています。**アドレス**値については処置後の値(つまり受信したまま)を比較対象とすることで対応します。このためホストからの**USB****アドレス**指定時、**アドレス**値は直前の**PID**値に応じて **SETUP/IN**(**PID**の**MSB**=**0**)用と **OUT**(**PID**の**MSB**側=**111**)用の2種類の比較値が以降の処理用に作成されます(“**SADR:**”)。

“**URBF:**”内の受信パケットは平タスク上で以下の順で元データに復元されます。

- ① **NRZI**復号(“**DNRZI:**”)
- ② バイト内ビット順反転(正順復帰, “**RVBITB:**”)
- ③ **(0)**ビット挿入削除(“**BSX:**”)

上記処理完了後、“**URBF:**”内は**SYNC**から **CRC16**までの元データになります。

ホストからの要求は**設定**(**Setup**)段階で**指示票**(**Token**)パケットが **SETUP\_PID**の時の**データ**(**Data**)パケットで決まります。この**データ**(**Data**)パケットは8バイトの制御情報を示し、必須の**標準要求**と任意の**供給者要求**を使用します。これらは **AVR309.PDF**内の**表1,2,3**で示されます。

本応用での受信処理が**処理単位**(**Transaction**)ではなくパケット単位なため、各段階を識別するために状態番号(“**UJNO**”)を設定することで対処しています。例えば、この処置を行わない場合、 **DATA0\_PID**のパケットを受信した時に、それが**設定**(**Setup**)段階の(制御用)データ(**Data**)パケットか、データ(**Data**)段階の( **OUT\_PID**の次の)データ(**Data**)パケットか、状態(**Status**)段階の( **IN\_PID**の次の)データ(**Data**)パケットかが判別できなくなります。

INT0割り込み処理内で標準または供給者要求のデータ(**Data**)パケットを受信すると、平タスクでそれに対する実処理を開始させるために平タスク用の状態番号(“**MJNO**”)を更新します。平タスクではその要求を解析し、対応する処理(準備)を行います。これは2段階で行われます。これは次の理由に依ります。

本装置からの送信データはホストからの **IN\_PID**に対応して(**Low-Speed**規定により)8バイト単位毎に送信されます。これらのデータ元はその要求に対応して、フラッシュメモリ、**SRAM**、**EEPROM**、**I/O**が在り得ます。このため、8バイトを越えるデータの場合、データ(**Data**)段階に於いて複数の処理単位(**Transaction**)で送信されることになります。従って送出データを準備する段階と各処理単位(**Transaction**)での **IN\_PID**に対応して実際に送信する段階との2段階で構成されなければなりません。実際の送信は、この **IN\_PID**を含むパケット受信直後にINT0割り込み処理内で行われます。よって平タスクの処理としては、要求解析/初回送出データ準備と次回送出データ準備の2状態が必要になります。最初に要求解析/送出データ準備状態になり、以降は次回送出データ準備が続き、最後にアイドル状態になります。

送出データ準備は8バイト単位データを送信緩衝部(“**UTBF:**”)に用意するところまでを実行します。ここで“**UTBF:**”内データは**(0)**ビット挿入処理が行われます。INT0割り込み処理内の実送信では、この用意されている“**UTBF:**”内データを**NRZI**符号化しながら**USB**バスへ送信します。

送受信共、1.5Mbpsを守るために8クロック周期(@12MHz)間隔の入出力を維持しなければなりません。詳細については「**ビット速度の考察**」をご覧ください。

## 送受信に於けるCRC

本応用はホスト機能を持ちませんのでCRC5の送信はありません。CRC5受信時の処理は省略されています。またデータパケットに於けるCRC16受信時の処理も省略されています。従って行われているのはデータパケット送信に於けるCRC16付加処理だけです。

本応用の実装方法は添付USBspcs.pdfの付録-A: CRC内の「ハードウェアCRC16計算回路」をエミュレートする形で行われています。  
**"GCRC1:"**で剰余レジスタ(TMP1:TMP0)を初期化(=全1)し、以降で実データ部に対する剰余値を求めています。**"GCRC3:"**からの3行がビット入力と $x^{16}$ のEOR値(BITCのLSB)取得、次の2行が剰余レジスタのビット移動(ビットクロック↑相当)、その次の**"CBR TMP0, 0b00000001"**は注意しなければなりません。これは剰余レジスタ最下位ビット( $x^0$ )値の仮設定です。本来、直前のビット移動でTMP0のLSBにはビット入力と $x^{16}$ のEOR値(BITCのLSB)が入らなければならないのですが、以降の処理の都合でビット移動直後は不定となっています。ここでの仮設定により、以降の**"BRCC GCRC4"**でCF=0(即ちBITCのLSB=0)の場合、この仮設定が本設定となり、CF=1(即ちBITCのLSB=1)の場合はCRC16P値とEORすることによって1に再設定され、結果的にTMP0のLSBにはビット入力と $x^{16}$ のEOR値(BITCのLSB)が設定されることになります。

結果となる剰余レジスタ値の論理反転とビット位置反転処理は**"ACRC16:"**内で行われます。

## UART/USART受信割り込み処理

本割り込み処理はINT0割り込みによるUSBパケットの初期同期処理への影響を極力少なくするようにしなければなりません。INT0割り込みが直ぐに実行されず、保留状態になる可能性は割り込みが禁止されている場合です。

本応用に於いて、この状態はUART/USART割り込み処理ルーチンだけです。割り込みが実行されると、全割り込み許可(I)ビットが自動的にクリア(0)され、全割り込みが禁止となるため、UART/USART受信割り込み処理では、可能な限り早く全割り込み許可(I)ビットをセット(1)して、INT0割り込みを許可しなければなりません。けれども単純にこれを行なう(例えばSEI命令)と、UART/USARTが2バイト以上の受信データを保持している場合、この割り込み処理ルーチン内で、更にUART/USART受信割り込みが実行されてしまいます。これを可能とするには、本処理ルーチンを再入可能な形式で記述しなければならず、処理が複雑になるため、本応用には不適切です。

この問題を避けるため、全割り込みを許可するのに先立ち、UART/USARTの受信割り込みを禁止(RXCIE=0)します。同様の理由により、本処理ルーチンの最後で受信割り込みを許可(RXCIE=1)する場合、その前に全割り込みを禁止します。

結果的に、これらの処置によるINT0割り込み禁止時間とINT0割り込み処理のビット同期検出までの時間の合計が、概ねSYNCの6ビット目の前半までの時間以下でなければなりません。現状の、この合計時間は概ね3ビット時間です。



以下はATmega8版原書を基準にした日本語版との使用シンボル名対応一覧です。記述順は原書での出現順です。AT90S2313版は一部を除き、基本的に同一シンボル名を使っているのので、本一覧を参考にしてください。

UCR	固有名使用により消滅(注)	backupSREG	SSVI
UBRR	固有名使用により消滅(注)	ACC	TMP4
EEAR	固有名使用により消滅(注)	lastBitstuffNumber	LJBP
USR	固有名使用により消滅(注)	OutBitStuffNumber	TRBC
E2END	EEEE	BitStuffInOut	BSFF
RAMEND128	定義法変更により消滅	TotalBytesToSend	TBYC
inputport	固有名使用により消滅	TransmitPart	TPKC
outputport	固有名使用により消滅	InputBufferLength	URBC
USBdirection	固有名使用により消滅	OutputBufferLength	UTBC
DATApplus	直接数値記述により消滅	MyOutAddress	OADR
DATAMinus	直接数値記述により消滅	MyInAddress	IADR
USBpinmask	直接数値記述により消滅	ActionFlag	MJNO
USBpinmaskDplus	元来未使用のため削除	temp3	TMP3
USBpinmaskDminus	元来未使用のため削除	temp2	TMP2
TSOPPort	固有名使用により消滅	temp1	TMP1
TSOPpullupPort	固有名使用により消滅	temp0	TMP0
TSOPPin	直接数値記述により消滅	bitcount	BITC
SOPbyte	SYNCbyte	ByteCount	BYTC
nSOPbyte	nSYNCbyte	inputbuf	INPD
nNRZISOPbyte	nNRZISYNCbyte	shiftbuf	SHFT
BaseState	直接数値記述により消滅	State	UJNO
SetupState	直接数値記述により消滅	RS232BufptrX	XL
InState	直接数値記述により消滅	RS232BufptrXH	XH
OutState	直接数値記述により消滅	USBBufptrY	YL
SOFState	直接数値記述により消滅	ROMBufptrZ	ZL
DataState	直接数値記述により消滅	;	
AddressChangeState	直接数値記述により消滅	USER_FNC_NUMBER	直接数値記述により消滅
DoNone	直接数値記述により消滅		
DoReceiveOutData	直接数値記述により消滅	■ プログラム内ラベル	
DoReceiveSetupData	直接数値記述により消滅	URXCaddr	URXINT
DoPrepareOutContinuousBuffer	直接数値記述により消滅	FIFOBufferNoOverflow	URXI1
DoReadySendAnswer	直接数値記述により消滅	UARTBufferOverflow	URXI2
MAXUSBBYTES	UBSZ	NoUARTBufferOverflow	URXI3
NumberOfFirstBits	元来未使用のため削除	NoIncRS232BufferFull	URXI4
NoFirstBitsTimerOffset	元来未使用のため削除	reset	START
InitBaudRate	RSIBR	Main	MAIN
InputBufferBegin	URBF	CheckUSBReset	MAIN1
InputShiftBufferBegin	USBF	WaitForUSBReset	MAIN2
MyInAddressSRAM	IASV	ProcPrepareOutContinuousBuffer	COTJ
MyOutAddressSRAM	OASV	ProcReceiveSetupData	SUPJ
OutputBufferBegin	定義法変更により消滅	INT0Handler	INT0I
AckBufferBegin	AKBF	CheckchangeMinus	SYN1
NakBufferBegin	NKBF	CheckchangePlus	SYN2
ConfigByte	CFLG	DetectSOPEnd	SYN3
AnswerArray	ADWK	Increment1	PC参照として無効なため削除
StackBegin	STKP-1	Increment0	SYN4
MAXRS232LENGTH	RBSZ	EndInt0Handler	INT0R
RS232BufferBegin	定義法変更により消滅	EndInt0HandlerPOP	INT0R1
RS232BufferEnd	基本的にRSBF+RBSZで記述	EndInt0HandlerPOP2	INT0R2
RS232ReadPosPtr	RSRP	USBBeginPacket	PKRX
RS232WritePosPtr	RSWP	USBloopBegin	PC参照として無効なため削除
RS232LengthPosPtr	RSBC	USBloop1_6	PKRX1
RS232Reserved	元来未使用のため	USBloop7	PC参照として無効なため削除
	.BYTE領域として確保	USBloop0	PC参照として無効なため削除
RS232FIFOBegin	RSBF	USBloopEnd	ANL
;		TestIOPacket	PC参照として無効なため削除
RS232BufferFull	元来未使用のため削除	TestSetupPacket	PC参照として無効なため削除
backupbitcount	BCSV	TestOutPacket	ANL1
RAMread	MEMF	TestInPacket	ANL2
backupSREGTimer	SSVU		



TestDataPacket	ANL3	CRC16LoopByte	GCRC3
Data0Packet	ANL4	CRC16NoXOR	GCRC4
NoMyPacked	ANL5	CRC16End	GCRCR
AnswerToInRequest	PKIN	LoadDescriptorFromROM	SFMD
ReceiveSetupData	PKSU	LoadDescriptorFromROMZeroInsert	SFMZ
ReceiveOutData	PKOT	InsertingZero	SFMZ1
NoReadySend	PKNA	InsertingZeroEnd	SFMZ2
SetMyNewUSBAddress	XADR	LoadDescriptorFromSRAM	SSMD
FinishReceiving	MVRB	LoadDescriptorFromEEPROM	SEED
ShiftRemainingBits	MVRB1	LoadXXXDescriptor	SXXD
NoRemainingBits	MVRB2	FromROM	PC参照として無効なため削除
MoveDataBuffer	MVRB3	FromRAMorEEPROM	SXXD1
USBReset	USBRST	EndFromRAMROM	M2BE
SendPreparedUSBAnswer	PKTXPC	PrepareUSBOutAnswer	PSUA
SendUSBAnswer	PKTXP	MakeOutBitStuff	TPIBS
SendUSBBuffer	PKTX	PrepareUSBAnswer	PRQA
SendUSBAnswerLoop	PKTX1	VendorRequest	VREQ
SendUSBAnswerByteLoop	PKTX2	NoDoSetInfraBufferEmpty	VREQ2
NoXORSend	PKTX3	NoDoGetInfraCode	VREQ3
NoXORSendLSB	PC参照として無効なため削除	NoDoSetDataPortDirection	VREQ4
SendUSBAnswerBitstuffLoop	PKTX4	NoDoGetDataPortDirection	VREQ5
NoXORBitstuffSend	PKTX5	NoDoSetOutDataPort	VREQ6
ZeroBitStuf	PKTX6	NoDoGetOutDataPort	VREQ7
SendUSBWaitEOP	PKTX7	NoDoGetInDataPort	VREQ8
ToggleDATA1PID	XDATAN	NoDoEEPROMRead	VREQ9
SendData0PID	XDATN1	NoDoEEPROMWrite	VREQ10
ComposeZeroDATA1PIDAnswer	MKZD1	NoDoRS232Send	VREQ11
ComposeZeroAnswer	MKZD	NoDoRS232Read	VREQ12
InitACKBuffer	MKACK	NoDoSetRS232Baud	VREQ13
SendACK	SDACK	NoDoGetRS232Baud	VREQ14
InitNAKBuffer	MKNAK	NoDoGetRS232Buffer	VREQ15
SendNAK	SDNAK	NoDoSetRS232DataBits	VREQ16
ComposeSTALL	MKSTL	NoDoGetRS232DataBits	VREQ17
DecodeNRZI	DNRZI	NoDoSetRS232Parity	VREQ18
NRZIloop	DNRZI1	NoDoGetRS232Parity	VREQ19
BitStuff	BSX	NoDoSetRS232StopBits	VREQ20
BitStuffRepeat	BSX1	NoDoGetRS232StopBits	VREQU0
SumAllBits	BSX2	NoDoUserFunction0	VREQU1
BitStuffLoop	BSX3	NoDoUserFunction1	VREQU2
BitStuffByteLoop	BSX4	NoDoUserFunction2	VREQX
IncrementBitstuff	BSX5	DoUserFunctionX	UFNJ
NoBitcountCorrect	BSXA1	DoUserFunction0	UF0J
CorrectOutBuffer	BSXAI	DoUserFunction1	UF1J
CorrectBufferEnd	BSXA2	DoUserFunction2	UF2J
DontShiftBuffer	BSX6	DoSetInfraBufferEmpty	SIRBEJ
EndBitStuff	BSXB	DoGetInfraCode	GIRCJ
DecrementLength	BSXBD	DoSetDataPortDirection	SDPDJ
IncrementLength	BSXBI	DoGetDataPortDirection	GDPDJ
NoChangeByteCount	BSXR	DoSetOutDataPort	SODPJ
ShiftInsertBuffer	INSB	DoGetOutDataPort	GODPJ
HalfInsertPosuvMask	INSB1	DoGetInDataPort	GIDPJ
ShiftInsertBufferLoop	INSB2	DoGetIn	GXX1S
NoEndShiftInsertBuffer	INSB3	DoEEPROMRead	EERJ
ShiftDeleteBuffer	DELB	DoEEPROMWrite	EEWJ
ShiftDeleteBufferLoop	DELB1	WaitForEEPROMReady	EEWJ1
HalfDeletePosuvMask	DELB2	DoRS232Send	RSSJ
DoneMask	DELB3	WaitForRS232Send	RSSJ1
MirrorInBufferBytes	RVBITB	DoRS232Read	RSRJ
MirrorBufferBytes	RVBIT	DoSetRS232Baud	SRSBRJ
MirrorBufferloop	RVBIT1	DoGetRS232Baud	GRSBFJ
MirrorBufferByteLoop	RVBIT2	DoGetRS232Buffer	GRSBDJ
CheckCRCIn	CRCK	SomeRS232Send	GRSBJ1
AddCRCOut	ACRC16	AsRequiredGetRS232Buffer	GRSBJ2
CheckCRC	GCRC16	NoShortGetRS232Buffer	GRSBJ3
ComputeDATACRC	GCRC1	ReadOverflow	GRSBJ4
CRC16Loop	GCRC2	ReadNoOverflow	GRSBJ5

DoSetRS232DataBits	SRSDBJ	InterfaceAnswer	IFMSG
Databits8or9Set	SRSDJ1	StatusAnswer	Z2MSG
RS232DataBitsLocal	SUCSZ	ConfigDescriptorEnd	定義法変更により消滅
GetUCSRCtotemp1	GUCSRC	LangIDStringDescriptor	LGMSG
Settemp1toUCSRC	SUCSRC	LangIDStringDescriptorEnd	定義法変更により消滅
DoGetRS232DataBits	GRSDBJ	VendorStringDescriptor	VNMSG
DoSetRS232Parity	SRSPBJ	CopyRight	元来未使用のため削除
SetParityOut	SRSPJ1	CopyRightEnd	元来未使用のため削除
StableParity	SRSPJ2	VendorStringDescriptorEnd	定義法変更により消滅
ParityErrorAnswer	SRSPJ3	DevNameStringDescriptor	DNMSG
DoGetRS232Parity	GRSPBJ	DevNameStringDescriptorEnd	定義法変更により消滅
ParityIsStable	GRSPJ1		
RetGetParity	GRSPJ2		
DoSetRS232StopBits	SRSSBJ		
DoGetRS232StopBits	GRSSBJ		
OneZeroAnswer	SZ1P		
StandardRequest	SREQ		
ComposeSET_ADDRESS	S_AD		
ComposeSET_CONFIGURATION	S_CF		
ComposeCLEAR_FEATURE	C_FT		
ComposeSET_FEATURE	S_FT		
ComposeSET_INTERFACE	S_IF		
ZeroStringAnswer	SZS1		
ComposeGET_STATUS	G_ST		
TwoZeroAnswer	SZ2P		
ComposeGET_STATUS2	SZNP		
ComposeGET_CONFIGURATION	G_CF		
ComposeGET_INTERFACE	G_IF		
ComposeSYNCH_FRAME	S_FM		
ComposeSET_DESCRIPTOR	S_DP		
ComposeGET_DESCRIPTOR	G_DP		
ComposeDeviceDescriptor	G_DDP		
ComposeConfigDescriptor	G_CDP		
ComposeEndXXXDescriptor	G_XDP		
HostConfigLength	G_XDP1		
Length8Multiply	G_XDP2		
ComposeStringDescriptor	G_SDP		
ComposeVendorString	G_VSDP		
ComposeDevNameString	G_DSDP		
ComposeLangIDString	G_LSDP		
ZeroDATA1Answer	SZD1		
SetMyNewUSBAddresses	SADR		
NewAddressNo6ones	SADR1		
NewAddressNo3ones	SADR2		
NRZIforAddress	NRZIAD		
SetMyNewUSBAddressesLoop	NRZIA1		
NoXORBits	NRZIA2		
PrepareOutContinuousBuffer	MDPK		
PrepareContinuousBuffer	MDPC		
NextAnswerInBuffer	MDPC1		
ComposeNextAnswerPart	MDPS		
Nad8Bytov	MDPS1		

## ■ 定数部

USBversion	USBver
VendorUSBID	VendorID
DeviceUSBID	DeviceID
DeviceVersion	DevVer
MaxUSBCurrent	MaxCurr
;	
DeviceDescriptor	DVMSG
DeviceDescriptorEnd	定義法変更により消滅
ConfigDescriptor	CFMSG
ConfigDescriptorLength	直接記述により消滅
ConfigAnswerMinus1	CAMSG



---

© *HERO* 2021.

本書はAVR309応用記述の補足情報です。

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。