AVR 8ﾋﾞｯﾄ ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ

# AVR053：tinyAVRとmegaAVR用内蔵RC発振器校正

応用記述

## 序説

この応用記述はISPまたはJTAGのｲﾝﾀｰﾌｪｰｽを持つAtmel® tinyAVR®とmegaAVR®のﾃﾞﾊﾞｲｽの内蔵RC発振器を校正する早くて正確な方法を記述します。それはAVRISP mkⅡ、JTAGICE mkⅡ、JTAGICE3、Atmel-ICE書き込み器を用いた校正を許すﾌｧｰﾑｳｪｱ ｿｰｽ ｺｰﾄﾞを提供します。それは製造用書き込み器に適合させることもできます。それらは「AT06015:Atmelﾏｲｸﾛ ｺﾝﾄﾛｰﾗの製品ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ」応用記述で網羅されます。

tinyAVRとmegaAVRの多くのﾃﾞﾊﾞｲｽは内蔵RC発振器からの走行の可能性を提供します。代表的に、この発振器はﾃﾞｰﾀｼｰﾄで指定される周波数の±1%以内に使用者校正することができます。この機能は外部発振器に比べて重要な費用節約を提供します。

工場校正は固定の動作電圧と温度で実行されます。特定の動作温度または温度に合わせるため、または発振器を異なる周波数に調節することにさえ、標準校正が提供するものより、より高い精度を達成するため、使用者によって、この応用記述での校正技術を実行することができます。

いくつかのｼｽﾃﾑでは外部ｸﾘｽﾀﾙを用いて発振器の走行時校正を実行することが必要かもしれません。これは「AVR055:内蔵RC発振器走行時校正に対する32kHzｸﾘｽﾀﾙの使い方」応用記述で網羅されます。

## 特徴

- 次の書き込みﾂｰﾙを用いた校正：AVR ISP mkⅡ,JTAGICE mkⅡ,JTAGICE3,Atmel-ICE
- 調整可能なRC発振器とISPまたはJTAGのｲﾝﾀｰﾌｪｰｽを持つtinyAVRとmegaAVRのﾃﾞﾊﾞｲｽを支援
- 代表的に±1%の精度で内蔵RC発振器を調節可能
- 仕様内でどの動作電圧と温度でもどの周波数へもRC発振器を調整
- 校正のために必要な外部部品が全くなし

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. 2555H-09/2016, 2555HJ2-02/2021

## 目次


序説 ・・・

# 1. 内蔵RC発振器

tinyAVRとmegaAVRの殆どのﾃﾞﾊﾞｲｽは工場校正された内蔵RC発振器を含みます。これは通常CPU用の既定ｸﾛｯｸ元で、どんな外部部品も必要ありません。

## 1.1. 内蔵RC発振器の使用者校正

製品に於ける内蔵RC発振器は5Vまたは3.3Vの特定の供給電圧で校正されます。校正中に使われた動作電圧についての情報に関しては個別ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。工場校正の精度はﾃﾞﾊﾞｲｽに依存して±3%または±10%以内です(ﾃﾞｰﾀｼｰﾄを参照してください)。設計が標準工場校正によって提供され得るものを超える精度を要求する場合、RC発振器の第2校正を実行することが可能です。これを行うことにより、代表的に±1%(工場校正で10%精度を持つﾃﾞﾊﾞｲｽついては±2%)以内の周波数精度を得ることが可能です。使用者校正手順の一部として発振器の周波数を変えることも可能です。

## 1.2. 基本周波数

発振器の基本周波数はどの前置分周にも先立って発振器周波数として定義されます。いくつかのAVR®ﾃﾞﾊﾞｲｽはｼｽﾃﾑ ｸﾛｯｸ前置分周器が特徴です。前置分周ﾚｼﾞｽﾀ(CLKPR)は予め定義された係数でｼｽﾃﾑ ｸﾛｯｸを分周するのに使うことができます。また、一般的にAVRﾃﾞﾊﾞｲｽのﾋｭｰｽﾞを通してこの前置分周器を事前設定することもできます。例えば、この任意選択を提供するﾃﾞﾊﾞｲｽ上のCKDIV8ﾋｭｰｽﾞのﾌﾟﾛｸﾞﾗﾑ(0)はｼｽﾃﾑ ｸﾛｯｸを8分周するように構成設定します。これはﾃﾞﾊﾞｲｽが最大周波数仕様以下で動作することを確実にするために行うことができます。CLKPRはｼｽﾃﾑ ｸﾛｯｸの周波数を内部的に変更するために走行時に変更することができます。

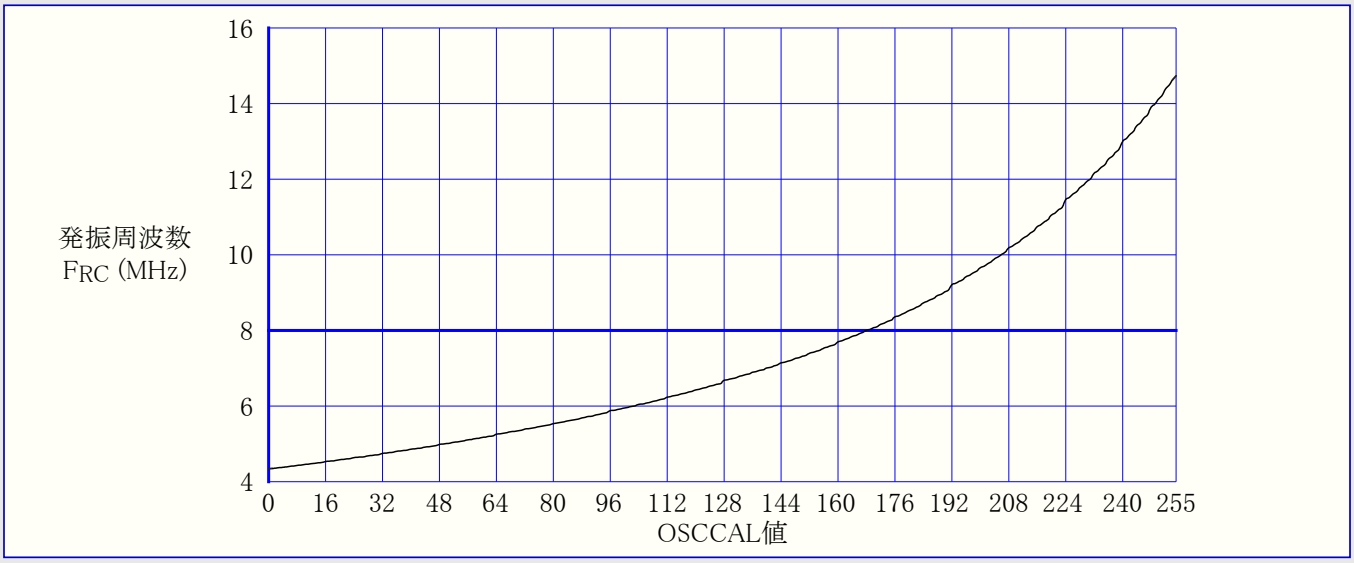
## 1.3. ｸﾛｯｸ選択

AVRﾃﾞﾊﾞｲｽのﾋｭｰｽﾞ設定は使われるｼｽﾃﾑ ｸﾛｯｸ元を制御します。内蔵RC発振器を校正するには、適切なﾋｭｰｽﾞを設定することによってCPUｸﾛｯｸ元として内蔵RC発振器を使うことが必要とされます。ﾋｭｰｽﾞの概要は特定ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄで利用可能です。

## 1.4. 調整可能なRC発振器

ｿﾌﾄｳｪｱによって調節することができる周波数のRC発振器は調整可能なRC発振器と呼ばれます。AVRﾃﾞﾊﾞｲｽでは、この目的に発振校正(OSCCAL)ﾚｼﾞｽﾀが使われます。それは下図で示されるように発振器周波数からの変動の処理を省くため校正付き内蔵RC発振器を調節するのに使うことができます。OSCCALﾚｼﾞｽﾀは1ﾊﾞｲﾄ幅です。

図1-1. 校正付き8MHz内蔵RC発振器周波数 対 発振校正(OSCCAL)値 (ATmega8)



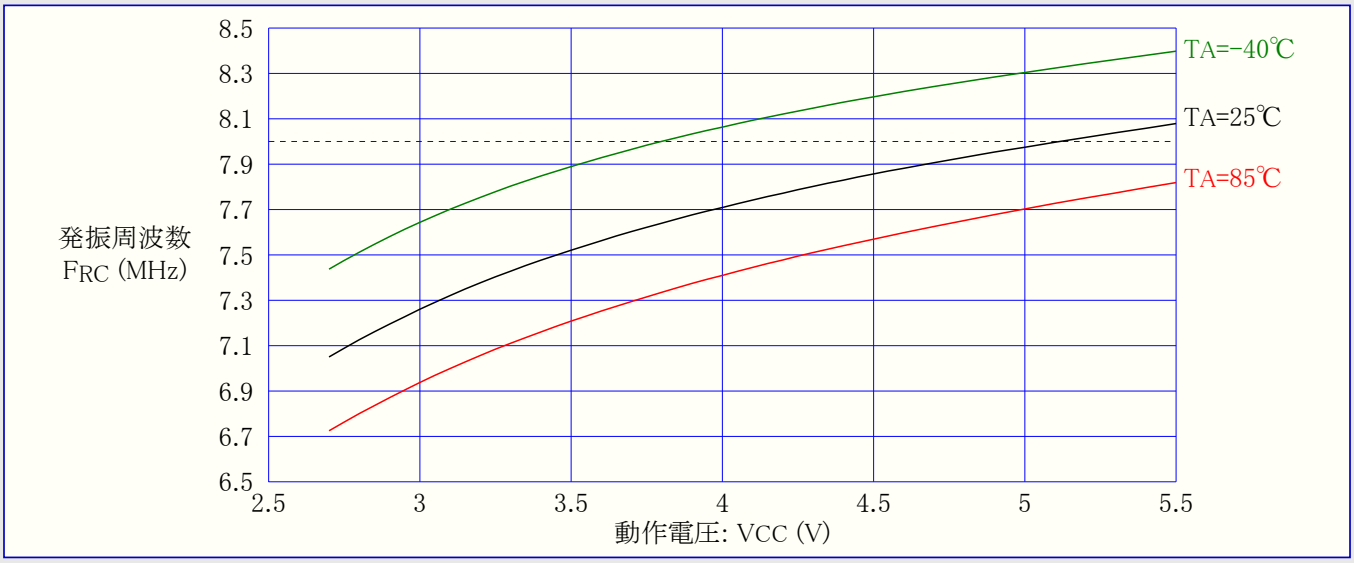

ﾃﾞﾊﾞｲｽ内のRC発振器が工場校正される時に、ﾃﾞﾊﾞｲｽの識票列に校正ﾊﾞｲﾄが格納されます。RC発振器周波数が工程依存のため、校正ﾊﾞｲﾄはﾃﾞﾊﾞｲｽ毎に変わり得ます。ﾃﾞﾊﾞｲｽが1つよりも多くの発振器を持つ場合、RC発振器の各々に対して校正ﾊﾞｲﾄが識票列に格納されます。

殆どのﾃﾞﾊﾞｲｽでは、始動で既定のRC発振器校正ﾊﾞｲﾄが自動的に識票列から取得されてOSCCALﾚｼﾞｽﾀ内に複写されます。例えば、ATmega8の既定ｸﾛｯｸ設定は1MHz内蔵RC発振器で、このﾃﾞﾊﾞｲｽに対して1MHz RC発振器に対応する校正ﾊﾞｲﾄが始動で自動的に設定されます。既定設定の代わりに4MHz発振器が使われるようにﾋｭｰｽﾞが変えられた場合、校正ﾊﾞｲﾄは手動でOSCCALﾚｼﾞｽﾀに設定されなければなりません。識票列から4MHz校正ﾊﾞｲﾄを読み、故にそれをﾌﾗｯｼｭ ﾒﾓﾘまたはEEPROMの位置へ格納するのに書き込みﾂｰﾙを使うことができ、そしてそれは走行時に主ﾌﾟﾛｸﾞﾗﾑによって読まれてOSCCAL内に複写されます。

## 1.5. 発振器特性

内蔵RC発振器の周波数は温度と動作電圧に依存します。この依存性の例は次図で、ATmega8の8MHz RC発振器の周波数を示します。周波数は電圧増加で高くなり、動作温度の増加で低くなります。これらの特性はﾃﾞﾊﾞｲｽ毎に変わります。特定ﾃﾞﾊﾞｲｽでの詳細についてはそれのﾃﾞｰﾀｼｰﾄを参照してください。

図1-2. 校正済み8MHz内蔵RC発振器周波数 対 動作電圧 (ATmega8)




## 1.6. RC発振器改訂履歴

その変遷を通してAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗで様々なRC発振器が利用されてきました。ﾃﾞﾊﾞｲｽ例と共にRC発振器の概要が下表で提供されます。この一覧は発振器形式によって分類され、そしてこれは発売時期による分類とも大体等価です。調整可能な発振器だけがこの表で一覧にされます。特定ﾃﾞﾊﾞｲｽでの発振器詳細についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

表1-1. 発振器の版

| 発振器版番号 | ﾃﾞﾊﾞｲｽ例 | RC発振器周波数 (MHz) | CKDIV8 | CLKPR |
|---|---|---|---|---|
| 1.1 | ATtiny12 | 1.2 | × | × |
| 1.2 | ATtiny15 | 1.6 | × | × |
| 2.0 | ATmega163 | 1.0 | × | × |
| 3.0 | ATmega16 | 1.0, 2.0, 4.0, 8.0 | × | × |
| 3.1 | ATmega128 | 1.0, 2.0, 4.0, 8.0 | × | XDIV (注1) |
| 4.0 | ATmega169 (注2) | 8.0 | ○ | ○ |
| 4.1 | ATtiny13 | 4.8, 9.6 | ○ | ○ |
| 4.2 | ATtiny2313 | 4.0, 8.0 | ○ | ○ |
| 5.0 | ATmega169P (注2) | 8.0 | ○ | ○ |

注1: 前置分周ﾚｼﾞｽﾀはこれらのﾃﾞﾊﾞｲｽでXDIVと名付けられています(訳補:分周係数もCLKPRと異なります)。

注2: ATmega169改訂A～Eは4.0版発振器を使い、一方ATmega169Pは5.0版発振器を使います。

### 1.6.1. 1.x版発振器

この版は校正することができるAVR用の最初の内蔵RC発振器です。これは1.2MHzから1.6MHzに渡る周波数で提供されます。校正ﾊﾞｲﾄは識票列に格納されますが、始動で自動的に設定されません。OSCCALﾚｼﾞｽﾀの設定は走行時にﾌｧｰﾑｳｪｱによって処理されなければなりません。この版での発振器周波数は動作電圧と温度に大きく依存します。

### 1.6.2. 2.x版発振器

この発振器版は1MHzの公称周波数を持ちます。動作電圧と温度での発振器周波数の依存性は1.x版と比べてかなり低減されます。

### 1.6.3. 3.x版発振器

この版の発振器ｼｽﾃﾑは多数の発振器周波数を提供します。このﾃﾞﾊﾞｲｽでは1,2,4,8MHzの公称周波数で4つの異なるRC発振器が存在します。この版は識票列から1MHz校正ﾊﾞｲﾄの自動設定が特徴です。4つの異なるRC発振器が存在するため、識票列に4つの異なる校正ﾊﾞｲﾄが格納されます。既定の1MHz以外の周波数が望まれる場合、OSCCALﾚｼﾞｽﾀは走行時に対応する校正ﾊﾞｲﾄで設定されるべきです。

### 1.6.4. 4.x版発振器

8MHzの単一周波数が4.0版で提供されます。後期の4.x版についてはATtiny2313に対して4MHzと8MHz、ATtiny13に対して4.8MHzと9.6MHzの2つの周波数が提供されます。OSCCALﾚｼﾞｽﾀは選択した発振器に対して周波数を調整するのに7ﾋﾞｯﾄだけが使われるように変更されます。MSBは使われません。既定校正値の自動設定とｼｽﾃﾑ ｸﾛｯｸ前置分周器が存在します。

### 1.6.5. 5.x版発振器

8MHzの単一周波数が5.0版で提供されます。既定校正値の自動設定とｼｽﾃﾑ ｸﾛｯｸ前置分周器が存在します。OSCCALﾚｼﾞｽﾀ内の全8ﾋﾞｯﾄが発振器周波数を調整するのに使われます。OSCCALﾚｼﾞｽﾀは2つの部分に分割されます。OSCCALのMSBは2つの重なっている周波数範囲の1つを選び、一方下位側7ﾋﾞｯﾄはこの範囲内で周波数を調整するのに使われます。ATmega406は4MHzの周波数を持ちますが、その他は同じです。

## 2. 校正

本章は校正規約と校正ﾌｧｰﾑｳｪｱに分けられます。この規約は校正を支援するどの検査またはﾌﾟﾛｸﾞﾗﾐﾝｸﾞのﾂｰﾙにも適合させることができます。AVRISP mkⅡ, JTAGICE mkⅡ,JTAGICE3,Atmel-ICEの書き込み器が実装された校正規約を支援します。ﾃﾞﾊﾞｲｽを校正するためのこれらのﾂｰﾙの使い方は後の項で記述されます。

### 2.1. 校正規約

校正用規約は製品環境で使えることを保証するために簡単且つ高速です。ﾃﾞﾊﾞｲｽをﾌﾟﾛｸﾞﾗﾐﾝｸﾞするのに使われるﾋﾟﾝ、それとISPｲﾝﾀｰﾌｪｰｽまたはJTAGｲﾝﾀｰﾌｪｰｽのどちらかは、それらが最終製品(または基板上)で利用可能である可能性が最も高いので校正のためにも使われます。

ISPｲﾝﾀｰﾌｪｰｽでのMOSIとMISO、またはJTAGｲﾝﾀｰﾌｪｰｽでのTDIとTDOの2つのﾋﾟﾝが校正に使われます。以降の説明を簡単にするため、MOSIとMISOだけが言及されますが、それらはJTAGｲﾝﾀｰﾌｪｰｽの場合にはTDIとTDOで置き換えることがでます。

全体概念は書き込み器が校正ｸﾛｯｸ(C-ｸﾛｯｸ)を生成し、ﾃﾞﾊﾞｲｽは内蔵RC発振器を校正するための参照としてこれを使うことです。ﾃﾞﾊﾞｲｽが校正を完了すると、MISO線で書き込み器への"OK"を合図します。

ﾃﾞﾊﾞｲｽにはMOSI線のﾌﾟﾙｱｯﾌﾟを許可する責任があり、書き込み器にはMISO線のﾌﾟﾙｱｯﾌﾟを許可する責任があります。不幸にも書き込み器は多くの場合でﾚﾍﾞﾙ変換器の後ろで、故にﾃﾞﾊﾞｲｽはMISO線もHighに設定します。これは雑音が校正を不正にしそうもないことを確実にするために行われます。

校正ﾙｰﾁﾝが1024C-周期(C-ｸﾛｯｸでの周期数)で完了されることを保証されているので、書き込み器は超過時間としてこのC-周期数を使うことができます。

### 2.2. 校正手続きを必要とする手順

1. 書き込み器はﾃﾞﾊﾞｲｽ内に校正ﾌｧｰﾑｳｪｱを書き、ことによるとMISOのﾌﾟﾙｱｯﾌﾟを許可してﾘｾｯﾄ線を開放します。JTAGｲﾝﾀｰﾌｪｰｽが使われる場合、ﾃﾞﾊﾞｲｽ内の校正ﾌｧｰﾑｳｪｱによってMCU制御ﾚｼﾞｽﾀ(MCUCR)またはMCU制御/状態ﾚｼﾞｽﾀ(MCUCSR)のJTAG禁止(JTD)ﾋﾞｯﾄが設定(1)されます。書き込み器によって(概ね32kHzの)校正ｸﾛｯｸがMOSI線に印加されます。
2. ﾃﾞﾊﾞｲｽはMOSI線の内蔵ﾌﾟﾙｱｯﾌﾟを許可し、MISO線をHighに設定して、MOSIでの校正ｸﾛｯｸ聴取を始めます。
3. ﾃﾞﾊﾞｲｽが校正ｸﾛｯｸを検出すると、1%精度の判定基準に合致するOSCCAL値を探すために2分検索が実行されます。この必要条件に合致する値が2分検索で判明しない場合、2分検索の結果に隣接する値が確認のために試験されます。校正失敗の場合、MISO線がLowに設定され、ﾌﾟﾛｸﾞﾗﾑの流れは**手順6.**へ行きます。
4. 校正値がEEPROM内に格納されます(校正失敗の場合は省かれます)。
5. ﾃﾞﾊﾞｲｽによってMISO線が8回/4循環交互に切り換えられます。MISO線の交互切り換えはMOSI線のｸﾛｯｸ(C-ｸﾛｯｸ)の下降端で実行されますが、5～10 CPU周期遅れます。校正失敗の場合はMISO線が交互に切り換えられません。
6. 必要なら、JTAGｲﾝﾀｰﾌｪｰｽが再許可され、ﾃﾞﾊﾞｲｽは無限繰り返しになります。
7. ﾃﾞﾊﾞｲｽがEESAVEﾋｭｰｽﾞを持たない場合、書き込み器は校正ﾌｧｰﾑｳｪｱがﾌﾗｯｼｭ ﾒﾓﾘから消去されてしまう時に後で再格納するためにEEPROMから校正ﾊﾞｲﾄを読み戻さなければなりません。ﾃﾞﾊﾞｲｽがEESAVEﾋｭｰｽﾞを持つなら、このﾋｭｰｽﾞはﾌﾗｯｼｭ ﾒﾓﾘの消去(ﾁｯﾌﾟ消去)がEEPROMも消去しないように設定(0)することができます。走行時にEEPROMまたはﾌﾗｯｼｭ ﾒﾓﾘからOSCCALﾚｼﾞｽﾀへ校正ﾊﾞｲﾄを複写することが必要です。従ってこれ用のﾙｰﾁﾝが最終ﾌｧｰﾑｳｪｱに実装されなければなりません。

### 2.3. 校正ﾌｧｰﾑｳｪｱ

校正ｺｰﾄﾞはAtmel Studio 7.0(またはそれ以降)用にAVRASM2ｱｾﾝﾌﾞﾘ言語で書かれています。校正ﾌｧｰﾑｳｪｱはISPまたはJTAGのｲﾝﾀｰﾌｪｰｽを持つ新しいtinyAVRまたはmegaAVRのﾃﾞﾊﾞｲｽを支援するのに容易に更新することができるような方法で構成されます。また、校正用のｲﾝﾀｰﾌｪｰｽを構成設定することもできます。

### 2.3.1. ファイル構成

表2-1. ファイル構成

| ファイル形式 | ファイルのパスと名前 | 説明 |
|---|---|---|
| Atmel Studio 解決策ファイル | rc_calib.atsln | プロジェクトと共に動き始めるためにAtmel Studioで開かれるべき解決策ファイル |
| 基礎コード ファイル | rc_calib¥RC_Calibration.asm | 他のファイル全てをインクルードする基礎ファイル |
| デバイス 特有ファイル | rc_calib¥Device specific¥all.asm | 発振器版のようなデバイス用の付加パラメータを決めるためにAtmel Studio内でのデバイス設定に使います。 |
| インターフェース 特有ファイル | rc_calib¥Interface specific¥ isp_atmelice_interface.inc、 jtag_jtagice3_interface.inc、など | ツールに特有な付加パラメータを指定します。RC_Calibration.asm内で1つだけがインクルードされ、その他は注釈にされるべきです。 |
| 共通ファイル | rc_calib¥Common¥macros.inc、 main.asmとmemoryMap.inc | |

基礎ファイルのRC_Calibration.asmは以下の(インクルード)ファイルを参照します。

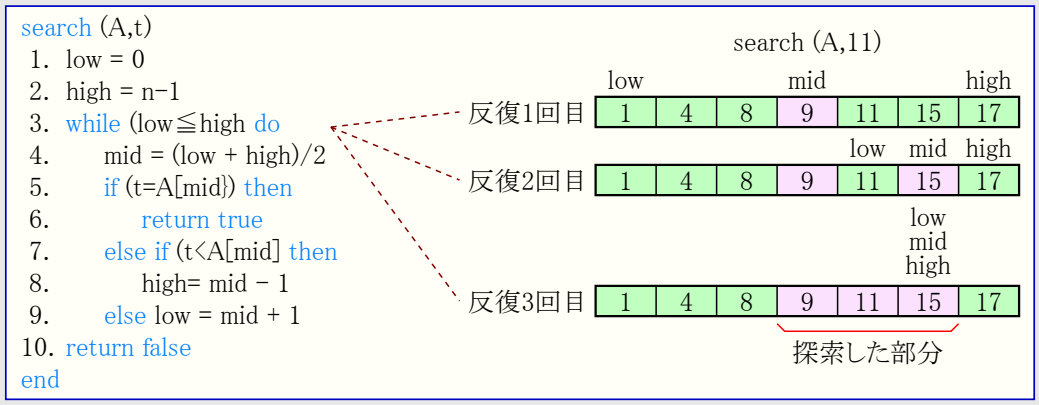
1. 付加パラメータを定義するデバイス特有ファイルはAtmel Studioでどのデバイスが選択されるかに依存します。このファイルは更に以下をインクルードします。
   1.1. コードが何処に配置されるかを定義するメモリ割り当てファイル
   1.2. 発振器版番号。このパラメータは或るOSCCALレジスタが7ビット幅でまた或るものは8ビット幅である事実を説明するため、2分検索で使われる初期段階量を決めます。
   1.3. デバイス特有ファイルにビットとレジスタの名前の再定義も存在するかもしれません。
2. 校正インターフェース特有ファイル。このファイルは主コードで使われる名前(ラベル)を持つISPまたはJTAGのポートとピンを割り当てます。校正クロック周波数はこのファイルで指定されます。
3. 使われるマクロを定義するファイル、macro.inc
4. 共通校正コード ファイル、main.asm

校正コードは望む目的対象デバイスと書き込み器に合わせるため、容易に変更を行えるように構成されます。更に、広範囲に渡るマクロの使用はコードが可能な最小容量になることを保証します。最後に、デバイスと校正インターフェースが設計される方法は新しいデバイスやインターフェースに対する支援が最小の労力で実装できることを保証します。

## 2.4. 2分検索

2分検索法は値によって整列された配列内で指定した入力値の位置を見つけます。整列された配列空間は望む値が中間(mid)部分とどう比較されたかに対応して繰り返して2つ割にします。下図は整列された配列Aからt=11の値を見つける例でこの算法を説明します。

図2-1. 2分検索



### 2.4.1. OSCCALの2分検索

検索は分割攻略法の2分検索法に基きます。

1. OSCCALレジスタはOSCCALの最大値の半分の初期値を設定されます。OSCCALの初期値は初期Step-Sizeで定義されます。
2. そしてシステム クロックの周波数が外部基準(校正クロック)と比較されます。
   - 2.1. 周波数が1%精度の限度内なら、**手順5.**へ行きます。
   - 2.2. システム クロックが速すぎるのを知ったならばOSCCAL値が減らされ、クロックが遅すぎるならばOSCCALが増やされ、**手順3.**へ行きます。
3. Step-Sizeは直前のStep-Sizeの半分の値を割り当てられます。
   - 3.1. Step-Sizeが0ならば2分検索は不成功でした。**手順4.**へ行きます。
   - 3.2. Step-Sizeが0でない場合、発振器周波数を増加または減少するためにStep-SizeがOSCCALレジスタの現在値(に加算)または(から減算)され、その後**手順2.**へ行きます。
4. OSCCALの最も近い4つの隣接値を調べます。これはOSCCALと発振器周波数間に厳密な単調関係がないことに対する補償のために行われます。
5. 調べたOSCCAL値が精度限度内なら、EEPROMに値の格納を続けて、成功を合図して校正を止めます。
6. 調べたOSCCAL値がどれも限度内でない(予想外)なら、MISOで線をLowに駆動することによって校正が失敗したことを合図し、校正停止へ行きます。

### 2.4.2. 発振器周波数の決定

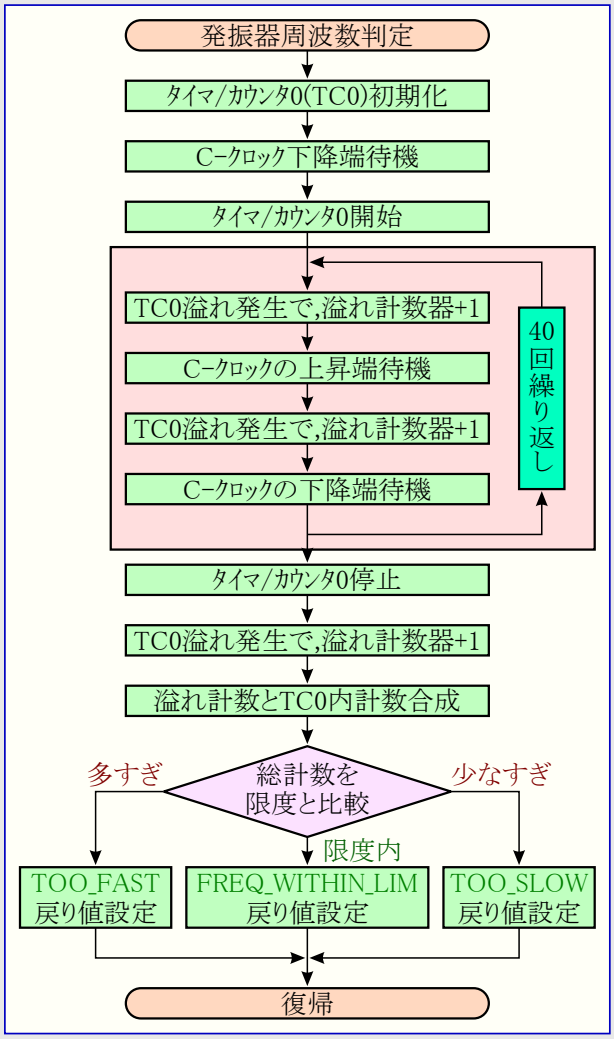
校正クロック(C-クロック)と内蔵RC発振器間の比較は8ビット タイマ/カウンタ0(TC0)を使って実行されます。8ビット タイマ/カウンタは調整可能なRC発振器を持つ全てのデバイスに存在するので使われます。考え方は40C-クロック周期の持続時間で予め定義された限度とタイマ/カウンタの計時数を比較することです。現実装のC-周波数はインターフェース特有インクルード ファイルで指定されます。発振器周波数を判定する方法は右の流れ図で記述されます。

1MHz～9.6MHzの発振器周波数の全範囲を網羅するため、事実上16ビットの計時器を提供する、8ビットによる(TC0)計時器範囲の拡張にTC0溢れフラグ(OVF)の検査が使われます。OVFフラグは(C-クロック)半周期毎に1度検査され、そしてこれは全てのTC0のOVF事象が検知されるのを保証するには充分な頻度です。実装された16ビット計時器の範囲関連で、最悪溢れ状態はOSCCALレジスタが$FFを格納された場合の9.6MHzです。この場合、発振器は指定周波数の100%以上になり得ます。この場合の計時器は未だ16ビット計時器の範囲内である23,541を計数するでしょう。

逆の方向へ行くことで、最低発振器周波数も考慮されなければなりません。最低周波数が得られるのはOSCCALに$00を書く時に起きます。その場合では指定よりも50%低くなるかもしれません。TC0のOVFフラグが半周期毎に検査されるため、OVFフラグを処理して(1MHzの指定周波数での)次のC-クロック端を検出するのに7 CPU周期よりも多いことは潜在的にありません。このタイミングの束縛はOVFフラグが設定(1)されていない時に遭遇し得ますが、フラグが設定(1)されるには8周期が必要です。これはタイミングの検出に於いて小さな誤差を引き起こしますが、全体結果に影響を及ぼさないでしょう。発振器は遅すぎとして正しく判定されるでしょう。

けれども、これらの極端な状態は2分検索法が使われるため、出会うことは全くありそうもありません。

**図2-2. C-クロックと内蔵発振器周波数間の関連判定法の流れ図**



### 2.4.3. ﾀｲﾐﾝｸﾞの不正確性の修正

全てのﾃﾞﾊﾞｲｽに於いてC-ｸﾛｯｸ端での割り込み駆動検出を使うことが不可能なので、ﾎﾟｰﾘﾝｸﾞ法が実装されます。この実装の結果はｴｯｼﾞ検出が最大2 CPUｸﾛｯｸ遅らされ得ることです。潜在的にこれは1%の要求精度に達する校正を失敗させ得ます。この潜在的なﾀｲﾐﾝｸﾞ誤差を補償するために限度が2計時器計数(2CPU周期)切り詰められます。限度と束縛の全ての計算はAVRASM2で64ﾋﾞｯﾄ精度を使うﾌﾟﾘﾌﾟﾛｾｯｻによって実行されます。表すことができない全ての値(浮動小数点値)は切り詰めた精度に丸められ、従って発振器に対する±1%精度の目的を危うくしないでしょう。

## 3. AVRﾂｰﾙを用いたﾃﾞﾊﾞｲｽ校正のための手順

以下の項はAtmel AVR書き込み器を使ってﾃﾞﾊﾞｲｽを校正する方法を説明します。

### 3.1. 校正ﾌｧｰﾑｳｪｱのｱｾﾝﾌﾞﾙ

校正ﾌｧｰﾑｳｪｱ用の基礎ｱｾﾝﾌﾞﾘ ﾌｧｲﾙはRC_Calibration.asmです。このﾌｧｲﾙはAtmel Studio解決策ﾌｧｲﾙのrc_calib.atslnによって指示されます。RC_Calibration.asmでは望む校正ｲﾝﾀｰﾌｪｰｽのAtmel-ICE、AVRISP mkⅡ、JTAGICE3、JTAGICE mkⅡを指定することが可能です。加えて、望む校正精度を指定することが可能です。一旦これらの選択が行われると、ﾌﾟﾛｼﾞｪｸﾄは既定副ﾌｫﾙﾀﾞにrc_calib.hex2進ﾌｧｲﾙを生成するように構築されるべきです。このﾌｧｲﾙは書き込み器を通してﾃﾞﾊﾞｲｽに書き込まれます。

**重要:** ﾃﾞﾊﾞｲｽを校正する前にﾋｭｰｽﾞが正しく構成設定されるのを確実にすることが重要です。ﾋｭｰｽﾞ設定によって1MHz RC発振器が選択された場合、ﾃﾞﾊﾞｲｽを8MHzに校正することはできません。

### 3.2. ｺﾏﾝﾄﾞ行ﾂｰﾙ

Atmel-ICE、AVRISP mkⅡ、JTAGICE3、JTAGICE mkⅡのﾂｰﾙでの校正支援はｺﾏﾝﾄﾞ行ﾂｰﾙのatprogramでだけ支援されます。Batch file副ﾌｫﾙﾀﾞでは目的対象ﾃﾞﾊﾞｲｽ内に校正ｺｰﾄﾞを書き込み、校正を実行してその後に最終ﾌｧｰﾑｳｪｱでﾃﾞﾊﾞｲｽを再書き込みするのにatprogramを使うﾊﾞｯﾁ ﾌｧｲﾙが提供されます。ﾊﾞｯﾁ ﾌｧｲﾙはAtmel-ICE、AVRISP mkⅡ、JTAGICE3、JTAGICE mkⅡを通してﾃﾞﾊﾞｲｽの校正を実行します。

### 3.3. ﾊﾞｯﾁ ﾌｧｲﾙ

校正に関する様々な動作を実行するために多数の命令が実行されるため、手動で個別命令を入力することは飽き飽きする工程です。故に手順を自動化するのにﾊﾞｯﾁ ﾌｧｲﾙが使われます。使われる書き込み器とｲﾝﾀｰﾌｪｰｽに基づき、ﾊﾞｯﾁ ﾌｧｲﾙの命名規則は次のとおりです。

atprogram_<書き込み器名>_<ｲﾝﾀｰﾌｪｰｽ名>_rc_calib.bat

例: atprogram_ATMELICE_ISP_rc_calib.bat

例えば、使われる書き込み器がJTAGICE3でｲﾝﾀｰﾌｪｰｽがJTAGの場合、そのﾌｧｲﾙ名は以下になるでしょう。

atprogram_JTAGICE3_JTAG_rc_calib.bat

### 3.4. 校正ﾌｧｰﾑｳｪｱの性能

校正全体がかなり素早く実行されるべきなので、ｺｰﾄﾞは効率に集中して書かれています。従って性能は校正ﾌｧｰﾑｳｪｱの大きさと校正を完了するためにかかる時間に依存します。校正ﾌｧｰﾑｳｪｱは代表的に数100ﾊﾞｲﾄで、目的対象ﾃﾞﾊﾞｲｽと校正に使われるｲﾝﾀｰﾌｪｰｽに依存します。故にﾌｧｰﾑｳｪｱを書き込むのに必要とされる時間は短時間です。校正ﾙｰﾁﾝは1024校正周期未満で完了されます。けれども、最短時間は2分検索法がどれくらい早く適合可能なOSCCAL値を見つけることができるかに依存します。

### 3.5. 校正ｸﾛｯｸ精度

校正の精度は外部校正ｸﾛｯｸの精度に大きく依存します。AVRﾂｰﾙによって生成される校正ｸﾛｯｸ周波数は変化するかもしれません。従って使われるﾂｰﾙの正確な周波数を測定して、ｲﾝﾀｰﾌｪｰｽ指定ｿｰｽ ﾌｧｲﾙ内にそれを入力することが重要です。振動子は動作電圧と温度の両方に依存するので、校正周波数はこれらのﾊﾟﾗﾒｰﾀが校正中に望む条件と等しい時に測定されるべきです。

### 3.6. 即時開始の手引き

支援されるﾃﾞﾊﾞｲｽで素早く校正の実行を開始には以下のそれらの手順に従ってください。

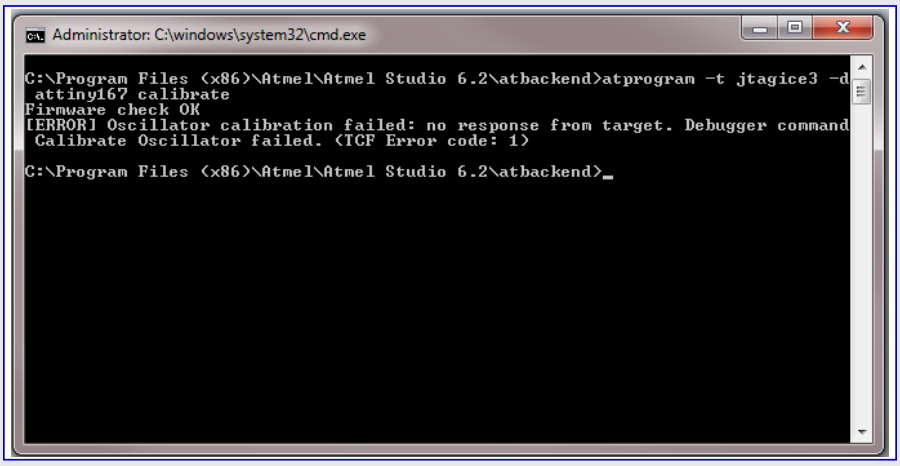
1. http://www.atmel.comからAtmel Studio 7.0(またはそれ以降)をﾀﾞｳﾝﾛｰﾄﾞしてｲﾝｽﾄｰﾙしてください。
2. この応用記述AVR053用のｺｰﾄﾞをﾀﾞｳﾝﾛｰﾄﾞして解凍してください(どの場所でも使え、ここでは¥AVR053¥と呼ばれます)。
3. Atmel Studioでrc_calib.atslnﾌﾟﾛｼﾞｪｸﾄ/解決策を開いてください。
4. Atmel Studioで目的対象ﾃﾞﾊﾞｲｽを選んでください(既定のﾃﾞﾊﾞｲｽはATmega168PBです)。全部ではありませんが多くのtinyAVRとmegaAVRのﾃﾞﾊﾞｲｽが支援されます。
5. ﾌｧｲﾙを開くためにRC_Calibration.asmｱｲｺﾝをｸﾘｯｸしてください。
6. 次図で示されるようにRC_Calibration.asm内のｲﾝｸﾙｰﾄﾞ行の1つからｾﾐｺﾛﾝ(";")を取り去ることによって望むｲﾝﾀｰﾌｪｰｽを選んでください。

図3-1. RC_Calibration.asmでのインターフェース選択

```
;*****************************************************************
;* 実行されるべき校正のインターフェース用ファイルのインクルード
;*****************************************************************
;.include "Interface specific¥isp_AVRISP_mkII_interface.inc"
;.include "Interface specific¥jtag_atmelice_interface.inc"
 .include "Interface specific¥isp_atmelice_interface.inc"
;.include "Interface specific¥isp_jtagice3_interface.inc"
;.include "Interface specific¥jtag_jtagice3_interface.inc"
```

7. 周波数計数器またはオシロスコープで校正クロックの周波数を測定してください。
   7.1. 校正クロックはISPツールのMOSIピンまたはJTAGツールのTDIピンで生成されます。
   7.2. ツールでこの信号を得るにはツールをPCに接続してコマンド シェル ウィンドウ(DOSプロンプト)を開き、C:¥Program Files (x86)¥Atmel¥Studio¥7.0¥atbackendフォルダへ誘導してください。atbackendフォルダへの正確なパスはAtmel StudioとWindows®の版とAtmel Studioがどうインストールされたかに依存してこれから僅かに変わるかもしれません。"atprogram -t <ツール名> -d <デバイス名> calibrate"を入力し、Enterを押してください。コマンド行の例は下図で示されます。これは校正クロックを数秒間生成させるでしょう。

図3-2. ツールで校正クロックを得るための命令




   7.3. "no response from target(目的対象からの応答なし)"異常が起こるかもしれませんが、命令の目的が正確に測定できるように校正クロックを生成することをツールに強制することなので、これはOKです。
8. 測定する周波数を反映させるようにインターフェース特有ファイル内の".EQU CALIB_CLICK_FREQ=xxxx"行をHzで変更してください。32092Hzの測定周波数用の例が下図で示されます。

図3-3. 校正クロック周波数の更新

```
;*****************************************************************
;* 校正クロック周波数指定
;*****************************************************************
.EQU  CALIB_CLOCK_FREQ          = 32092    ; Hzでの校正クロック周波数
```

9. RC_Calibration.asmファイルに於いて望む目的対象周波数(.EQU TARGET_FREQ = XXXX)と望む精度(.EQU ACCURACY = XX)を指定してください。

   注: 精度がきつ過ぎる場合はデバイスを校正することが不可能で校正は失敗するでしょう。達成できる精度についてはデバイスのデータシートを参照してください。
10. "File"その後に"Save All"をクリックすることによってAtmel Studioで変更した全てのファイルを保存してください。
11. プロジェクトをアセンブルしてデバイス内に書き込まれるべきhexファイルを生成するために"Build"その後に"Build rc_calib"をクリックしてください。
12. Atmel Studioを抜け出してください。
13. 何れかのテキスト エディタを用いて、選んだ書き込み器とインターフェースに対応するバッチ ファイルを開いてください。
14. "@SET CPU=xxxx"行を、例えば"@SET CPU=atmega168pb"のように変更することによって望むデバイスに合うようにファイルを編集してください。
15. JTAGインターフェースが校正に使われそうな場合、Atmel-ICE、JTAGICE3、JTAGICE mkⅡに関してリセット線が利用可能でなければならないことに注意してください。

16. CAL_FUSESに対して引数を変更することによって下図で示されるようにヒューズ設定をデバイスに対して望む設定に編集してください。望む校正周波数に対応した設定、デバイスを8MHzに校正するならば8MHz内蔵RCを選択することを確実にしてください。ウォッチドッグ タイマ常時ON (WDTON)ヒューズが設定されないことを確認してください。

図3-4. ヒューズ設定

```
@REM CPU型式と書き込み器への完全なパスで満たしてください。
@SET CPU=atmega2560
@SET SW_TOOL="C:¥Program Files (x86)¥Atmel Studio 6.2¥atbackend¥atprogram.exe"
@SET TOOL=atmelice
@SET CAL_FULSES=e299ff
```

ヒューズ：下位=$E2、上位=$99、拡張=$FF

17. Atmel Studioに対するインストール パスがバッチ ファイルで使われるものと違う場合、下図で示されるように適切なatprogram.exeファイルへのパスに変更してください。

図3-5. ツール パス

```
@REM CPU型式と書き込み器への完全なパスで満たしてください。
@SET CPU=atmega32
@SET SW_TOOL="C:¥Program Files (x86)¥Atmel Studio 6.2¥atbackend¥atprogram.exe"
@SET TOOL=atmelice
@SET CAL_FULSES=6499
```

18. 製品校正については校正成功での@PAUSE命令を取り去ることができます。
19. バッチ ファイルを保存してください。
20. ツールを目的基板に接続してください。目的対象基板とツールに給電してください。シリアルまたはUSBのケーブルがツールとPC間で接続されているのを確認してください。
21. コマンド シェル ウィンドウ(DOSプロンプト)を開き、"rc_calib¥Batch file¥"フォルダへ誘導して(atprogram_ATMELICE_ISP_rc_calib.bat, atprogram_JTAGICE3_ISP_rc_calib.bat,などの)適切なバッチ ファイルを実行してください。
22. 校正完了まで数秒間待ってください。バッチ ファイルは校正後にcalib_test.hexファームウェアよりもむしろ独自ファームウェアを(デバイスへ)書き込むために変更することもできます。ファームウェアによって走行時に新しい校正値がOSCCALレジスタに格納されるべきであることに注意してください。

## 3.7. 新デバイスに対する支援の追加

新デバイスに対する支援を追加するにはall.asmにデバイスに対する適切な定義を追加することが必要です。all.asmファイルで旧デバイス用の"#ifdef #endif"塊を複写/貼り付けをしてそれを新デバイスの特性に適合させることができます。ファイルを新デバイスに適合させる時に下の検査一覧を使うことができます。この検査一覧は例としてATmega8535を使います。

ピンと機能の互換デバイス用の"#ifdef #endif"塊を複写してください。

ATmega8535はATmega16とピン互換ですが、ATmega8535はJTAGインターフェースを持ちません。all.asmで"#ifdef _M16DEF_INC #endif"塊が複写され、"#ifdef _M8535DEF_INC #endif"と改名されます、

新デバイスの発振器版番号に合致するように発振器版番号を変更してください。

それが正しくアセンブルされたことを確認しれください。そうでなければ、これはポート、ピンまたは計時器の変更されたレジスタ名やビット名のための可能性が最も高いでしょう。ATtiny13はATtiny12の再割り当てとして実装され、再割り当てする名前に対する参照として使うことができます。

# 4. 参考文献


AVR054:内蔵RC発振器の走行時校正 – http://www.atmel.com/Images/doc2563.pdf

AVR055:内蔵RC発振器走行時校正に対する32kHzクリスタルの使い方 – http://www.atmel.com/Images/doc8002.pdf


# 5. 改訂履歴

| 資料改訂 | 日付 | 注釈 |
|---|---|---|
| 2555E | 2006年1月 | ハンドシェークの詳細更新と2.7.1.の重複情報削除 |
| 2555F | 2006年3月 | |
| 2555G | 2006年5月 | |
| 2555H | 2016年9月 | 資料の完全な刷新と一新 |

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

© 2016 Atmel Corporation. / 改訂:Atmel-2555H-Internal-RC-Oscillator-Calibration-for-tinyAVR-and-megaAVR-Devices_AVR053_Application Note-09/2016

Atmel®、Atmelﾛｺﾞとそれらの組み合わせ、Enabling Unlimited Possibilities®、AVR®、tinyAVR®、megaAVR®とその他は米国及び他の国に於けるAtmel Corporationの登録商標または商標です。他の用語と製品名は一般的に他の商標です。

**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のｳｪﾌﾞｻｲﾄに位置する販売の条件と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、または利益と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。

**安全重視、軍用、車載応用のお断り**: Atmel製品はAtmelが提供する特別に書かれた承諾を除き、そのような製品の機能不全が著しく人に危害を加えたり死に至らしめることがかなり予期されるどんな応用(”安全重視応用”)に対しても設計されず、またそれらとの接続にも使用されません。安全重視応用は限定なしで、生命維持装置とｼｽﾃﾑ、核施設と武器ｼｽﾃﾑの操作用の装置やｼｽﾃﾑを含みます。Atmelによって軍用等級として特に明確に示される以外、Atmel製品は軍用や航空宇宙の応用や環境のために設計も意図もされていません。Atmelによって車載等級として特に明確に示される以外、Atmel製品は車載応用での使用のために設計も意図もされていません。

© *HERO* 2021.

本応用記述はAtmelのAVR053応用記述(Rev.2555H-09/2016)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。