AVR140 : LIN応用のためのATmega48/88/168系列 内蔵RC発振器の走行時校正

要点

- UART経由での内蔵RC発振器の校正
- 目標周波数の±2%以内へのLIN 2.0適合の同期/校正
- ATmega48/88/168 AVR®系列を支援
- 変化する動作条件に於いて安価なｸﾛｯｸ元で強いLIN UART通信が可能

序説

この応用記述はUART経由で内蔵RC発振器を校正する方法を記述します。使う方法は局所相互連結網(LIN:Local Interconnect Network)規約で使われる校正方法に基き、毎回のﾒｯｾｰｼﾞ ﾌﾚｰﾑの始めで従節点(ﾉｰﾄﾞ)を主節点に同期化します。これは例え内蔵RC発振器のような安価なｸﾛｯｸ元で走行している時でも、指定された制限内のﾎﾞｰﾚｰﾄで他の節点との通信を従節点に許します。

ATmega48/88/168 AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗ系列は、内蔵RC発振器で走行する可能性を提供します。内蔵RC発振器の周波数はﾃﾞﾊﾞｲｽに対するﾃﾞｰﾀｼｰﾄで指定される周波数の±1%以内に校正することができます。この特徴は同期化の目的に理想的で、外部発振器の使用に比べて重要な費用節約を提供します。

この実装が内蔵RC発振器の周波数を変えるのに同期信号を使い、更にUART部のﾎﾞｰﾚｰﾄを変えることに注意してください。この場合に於ける用語の「同期」と「校正」は本質的に同じことを意味し、同義に使われます。語句の選択は単に目的に対する関連にすぎません。

動作の理屈 – 内蔵RC発振器

製品に於ける内蔵RC発振器は3.3Vと周囲温度で校正されます。この工場校正の精度はATmega48/88/168ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ系列を含み、全ての車載AVRに関して±1%以内です。

ｸﾛｯｸ選択

AVR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。ﾃﾞﾊﾞｲｽは8.0MHzでの内蔵RC発振器とﾌﾟﾛｸﾞﾗﾑ(0)されたCKDIV8ﾋｭｰｽﾞで結果として1.0MHzのｼｽﾃﾑ ｸﾛｯｸで出荷されます。始動時間は最大と時間経過区間許可に設定されます(CKSEL=’0010’,SUT=’10’,CKDIV8=’0’)。この既定設定は全ての使用者が利用可能な、どのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｲﾝﾀｰﾌｪｰｽを使っても、それらを望むｸﾛｯｸ元設定にすることができることを保証します。正しいLIN応用のために推奨される値である8MHzで内蔵発振器を使うには、対応するﾋｭｰｽﾞ設定がCKSEL3～0=’0010’とCKDIV8=’1’に設定されなければなりません。

基準周波数

後続項はATmega48/88/168 AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗ系列で利用可能な内蔵RC発振器の概要を提供します。

ATmega48/88/168は1つの8MHz内蔵RC発振器を持ちます。充分な精度の内蔵RC発振器にするためにAVRのI/O領域内に発振校正(OSCCAL)ﾚｼﾞｽﾀが存在します。OSCCALﾚｼﾞｽﾀは1ﾊﾞｲﾄ幅です。このﾚｼﾞｽﾀの目的は発振器周波数の調整を可能にすることです。この調整はRC発振器を校正する時に利用可能です。

Atmelがﾃﾞﾊﾞｲｽを校正するとき、校正ﾊﾞｲﾄがﾃﾞﾊﾞｲｽの識票列に格納されます。RC発振器周波数が工程に依存するため、この校正ﾊﾞｲﾄはﾃﾞﾊﾞｲｽ毎に変わり得ます。ATmega48/88/168は内蔵RC発振器用にﾊﾞｲﾄ校正値を持ちます。このﾊﾞｲﾄは識票ｱﾄﾞﾚｽ空間のｱﾄﾞﾚｽ$0000の上位ﾊﾞｲﾄに属しています。ﾘｾｯﾄ中、校正付きRC発振器の正しい周波数を保証するために、このﾊﾞｲﾄがOSCCALﾚｼﾞｽﾀ内へ自動的に書かれます。識票列から8MHz校正ﾊﾞｲﾄを読んでそれをﾌﾗｯｼｭ 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ます。主ﾌﾟﾛｸﾞﾗﾑは走行時にこの位置を読んでOSCCAL内に複写します。

8ﾋﾞｯﾄ 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. 7653A-09/06, 7653AJ2-02/21

RC発振器概要

ATmega48/88/168 AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗ系列の概要とそれらの発振器は表1.で得られます。この8MHz発振器は周波数を調整するためのOSCCALﾚｼﾞｽﾀの8ﾋﾞｯﾄによって制御されます。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。

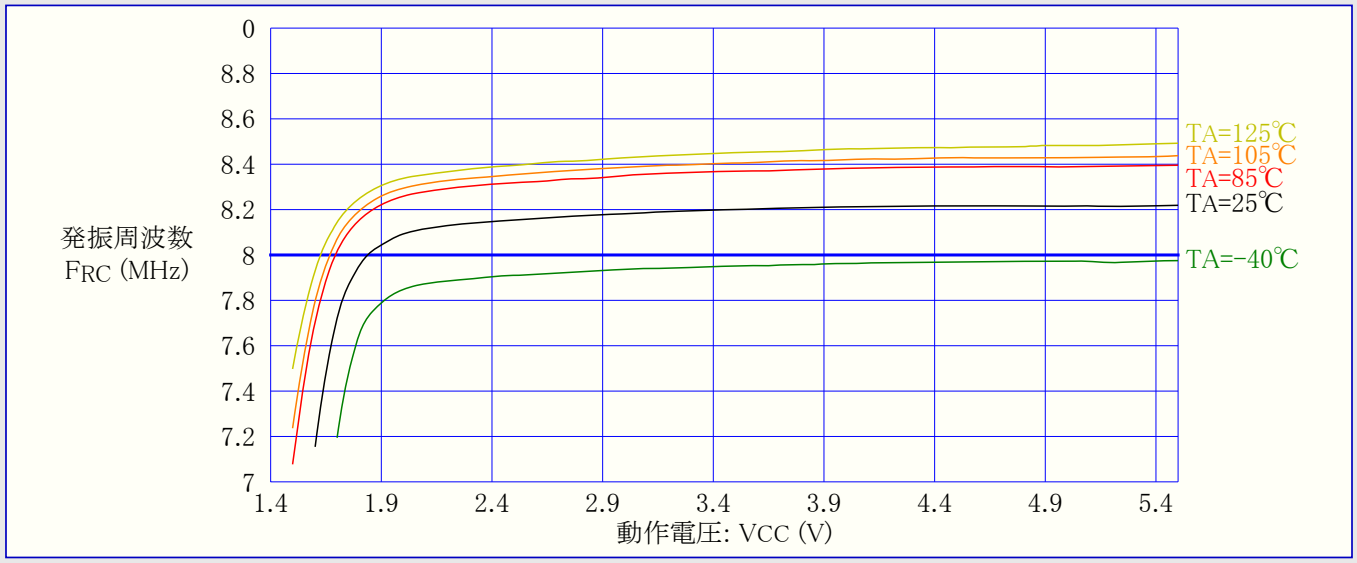
表1. ATmega48/88/168ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ系列に関するRC発振器の主な特徴

発振器版番号	ﾃﾞﾊﾞｲｽ	RC発振器周波数 (MHz)	CKDIV8	CLKPR
5.0	ATmega48/88/168	8.0	○	○

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。この依存性の例が図1.で図解され、そしてこれはATmega48/88/168の8MHz RC発振器の周波数を示します。この図から見られるように、周波数は温度上昇で増加し、動作電圧上昇でも僅かに増加します。

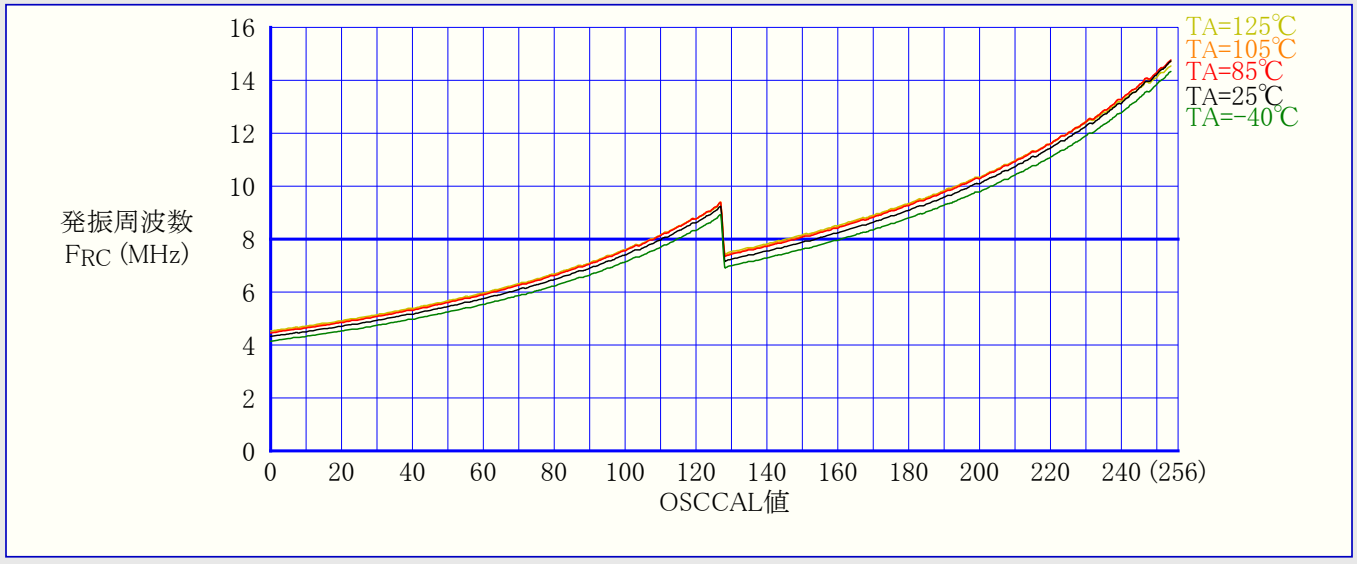
図1. 未校正発振器周波数と、温度と動作電圧による影響、ATmega88 8MHz発振器周波数 対 動作電圧



調整可能な発振器付きの全ﾃﾞﾊﾞｲｽは発振器周波数を調整するためのOSCCALﾚｼﾞｽﾀを持ちます。OSCCALの値増加は周波数に於いて"擬似単調"増加に帰着します。これを擬似単調と呼ぶ理由はOSCCAL値の或る単一増加に関して、周波数が増加しない、または僅かに減少するからです。けれども、次の単一増加は常に再び周波数を増加します。換言すると、OSCCALﾚｼﾞｽﾀを1増加することは周波数を増加しないかもしれませんが、OSCCAL値を2増加することは常に周波数を増加します。この情報は与えられた周波数に合致する最良の校正値を見つける時に大いに関係します。

重要: 校正原点からの最大偏差は温度の電圧の全範囲で±10%未満です。

図2. 校正付き8MHz内蔵RC発振器周波数 対 発振校正(OSCCAL)値



ATmega48/88/168のOSCCAL値と発振器周波数間に関連する擬似単調の例は図2.で見ることができます。この図はOSCCAL=0～255間の周波数変化を示します。OSCCALﾚｼﾞｽﾀは2つの部分に分けられます。OSCCALのMSB(OSCCAL7)は2つの重複する周波数範囲の1つを選択し、一方下位7ﾋﾞｯﾄはその範囲内で周波数を調整するのに使われます。

重要: 調整可能な全ての発振器に関して、ﾃﾞｰﾀｼｰﾄで指定された基準周波数から10%よりも多く外れる周波数調整が推奨されないことに注意するのが重要です。この理由はﾃﾞﾊﾞｲｽ内の内部ﾀｲﾐﾝｸﾞがRC発振器周波数に依存しているからです。

周波数安定時間

新しいOSCCAL値が設定されてしまうと、新しい周波数で安定させるために内蔵RC発振器に対して幾らかの時間がかかり得ます。この安定時間は5µsよりも長いことは決してありません。校正に関してどんな周波数測定をするのにも先立って、発振器を新しい周波数で安定させて置いてください。

LINの同期方法

局所相互連結網(LIN)規格は内蔵RC発振器のような安価なｸﾛｯｸ元を使う時でも信頼できる通信を可能にするように設計されています。本来の不正確さとｸﾛｯｸ元のような環境に依存する特性のため、同期調整は規約に含まれています。LIN同期の原理はこの応用記述で記述される同期方法を基礎として用います。

LIN網は1つの主節点(ﾉｰﾄﾞ)と多数の従節点から成ります。主節点にはﾊﾞｽ上の全ての通信を管理する責任があります。LIN用語ではﾊﾞｽ上にﾒｯｾｰｼﾞ ﾌﾚｰﾑを送出することによって通信が発生します。全てのﾒｯｾｰｼﾞ ﾌﾚｰﾑはﾌﾚｰﾑ先頭部で始まり、主節点によって開始されます(**図3.**をご覧ください)。この先頭部はBREAK(同期中断)とSYNCH(同期領域)の様式で始まり、ﾊﾞｽ上で何かの通信が開始される前に主節点に同期することを従節点に許します。BREAK/SYNCH様式は以下から成ります。

- BREAK信号：最低13ﾋﾞｯﾄ時間の優性(Low)値。
- SYNCH分離子：最低1ﾋﾞｯﾄ時間の劣性(High)値。
- SYNCHﾊﾞｲﾄ：$55が送信されます。開始ﾋﾞｯﾄと停止ﾋﾞｯﾄを含め、結果として'0101010101'のﾋﾞｯﾄ様式が送信されます(ﾋﾞｯﾄ送信順はLSB先行です)。**図4.**をご覧ください。

図3. LINﾌﾚｰﾑ形式

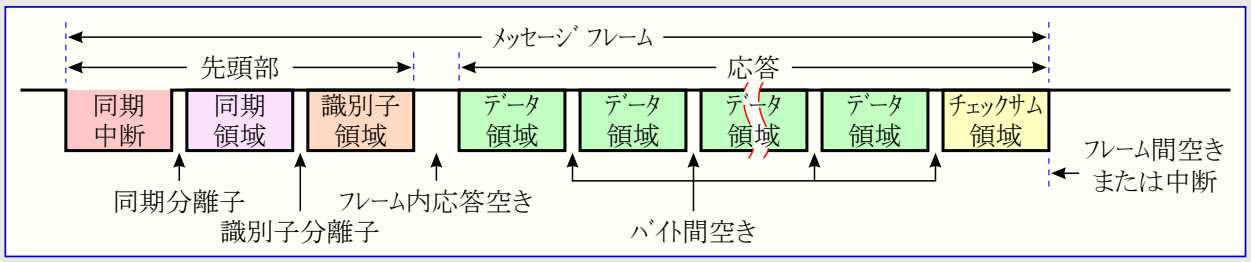
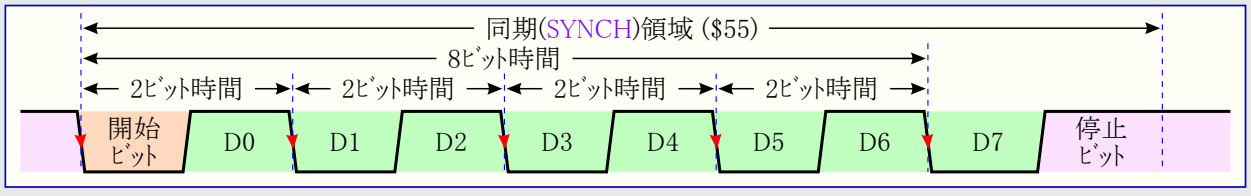


図4. ﾌﾚｰﾑ先頭部内の同期様式($55)



SYNCH(同期)ﾊﾞｲﾄ後に識別子が送信されます。識別子はどの従節点がﾊﾞｽにﾃﾞｰﾀを送信することになっているのかと、何の情報が従節点に要求されたのかを固有に定義します。

未定義のﾃﾞｭｰﾃｨ ｻｲｸﾙの重要性

LIN信号のﾃﾞｭｰﾃｨ ｻｲｸﾙはﾊｰﾄﾞｳｪｱ実装に依存し、このために応用と別の応用で変化するかもしれません。従って、TBus_優性とTBus_劣性に対する値は等しくありません。これは実際、SYNCHﾊﾞｲﾄでの$55を考慮する時に発振器を同期する方法に於いて強い影響があります。SYNCHﾊﾞｲﾄの上昇端と下降端の全ての切り替わりを使う代わりに、連続する上昇端と連続する下降端だけが考慮されなければなりません。

同期の方法

発振器動作中のOSCCAL値変更の可能性は、従節点の元の校正で主フレームに入ることを可能にします。動作の原理はSYNCHバイト中にビット時間を測定し、局所的な電圧や温度の偏差のための局所的な周波数変動から回復するようにOSCCALの値を変更することです。

そのようにするために、2分法の方法は図5.で記述されるように提唱されます。

SYNCHバイトの$55は測定とOSCCAL変更の3つの窓を提供します。より多くの詳細については図6.をご覧ください。

図5. RC発振器同期用自動2分校正法

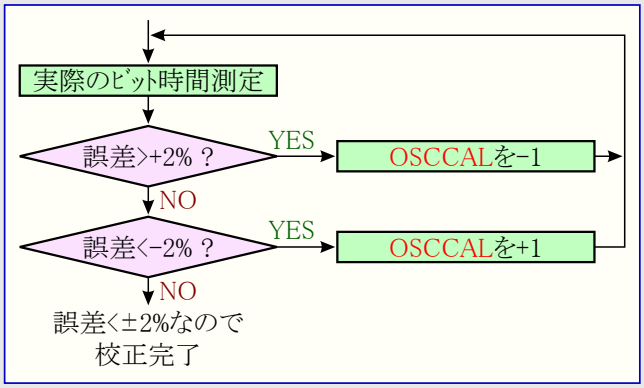
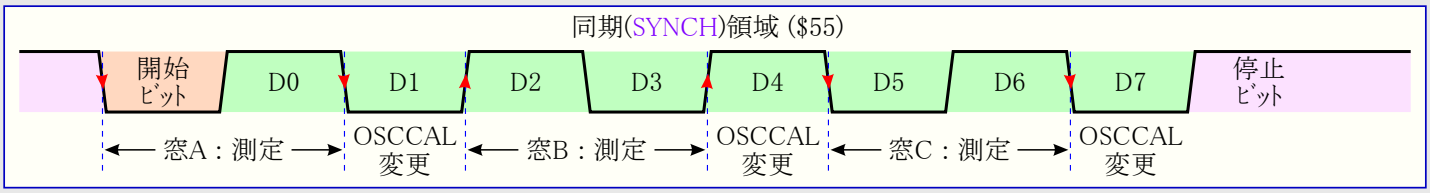


図6. ビット時間測定とOSCCAL変更に使われるSYNCHバイト



開始信号(LINrx)は16ビット タイマ/カウンタの捕獲入力へ送られます。窓Aの間、この信号の最初の下降端が16ビット タイマ/カウンタの値で時間を記されます。そしてLINrxの次の下降端が再び一度このタイマ/カウンタの新しい値で時間を記されます。(2×ビット時間を表す)差が基準と比較され、OSCCALをどう変更するかの判定が行われます。内蔵RC発振器の周波数は次の窓(窓B)が起こる前に安定にされなければなりません(思い出してください、OSCCALが変更された後で周波数が安定するのに最大5μsかかります)。この手順は合計3回(窓A,B,C)まで繰り返せますが、測定されたビット時間が必要とする誤差(±2%)内になれば直ぐに終了することができます。方法とSYNCHバイトの両方のより多くの詳細については図5.と図6.をご覧ください。

最適な2分法の段階

主フレーム開始に対する正しい同期のための最適なOSCCAL値の検索は2分法です。SYNCHバイト(図6.をご覧ください)中に許される3つの検索窓と最悪条件で測定される最大偏差に基き、右の増加/減少が選ばれています。

これらの3つの値は元のOSCCAL値と動作条件が何であれ、いつも動作することを検証された選ばれた値です。

表2. 正しいLIN同期のために選ばれた2分法の増加/減少

測定窓	OSCCAL変更 (段階)
窓A	±16
窓B	±8
窓C	±4

校正と製品デバイスの審査

表1.で示されるように、製品デバイスは8MHzで校正されて出荷されます。校正ルーチンは周囲温度と3Vで実行され、OSCCAL7の結果は'0'または'1'のどちらにもなり得ます。

直線的な校正と正しい同期ルーチンを可能とするため、デバイスはOSCCAL=127とOSCCAL=128に於いてそれらの周波数に関して審査されます。温度と電圧の全範囲に渡る周波数の最大偏差が10%なので、OSCCAL=127で8.8MHz、またはOSCCAL=128で7.2MHzの周波数を持つデバイスだけが許容されます。これらの限度はこの後で記述されるLIN同期法の必要条件を満足するように定義されています。

OSCCALの2分法に対する予防処置

図2.は意図的な不連続を図解します。1つの正しい再同期に関して、その周波数変更は不連続の同じ側(OSCCAL7の無変化)を維持されなければなりません。10%よりも大きく変わった周波数を持つデバイスがないので、故に10%よりも大きな値で周波数を変更する理由は全くありません。従って、OSCCAL値に於いて連続する増加(または減少)のために校正が境界の横断を試みる時に、ルーチンは停止されなければなりません。

例: この曲線の下部で動作するデバイスに対して、新OSCCAL>127の場合は新OSCCAL=127です。不連続の上部で動作するデバイスに関しても同様です。

LIN提唱法との比較

LIN仕様は同期(SYNCH)領域中に1度、局所的なﾋﾞｯﾄ時間の測定に基く同期法を提唱しています。これはﾌﾚｰﾑの残りの部分が解釈される前に局所発振器の修正を可能にします。この応用記述で記述される方法は使用者に3回までの局所RC発振器修正、従ってｼｽﾃﾑの精度改善(例えば、強化)を提供します。この手順でﾌﾚｰﾑがなければ、始動でも同期化の理由のために誤って解釈され得ます。

Cｺｰﾄﾞ例

以降のCｺｰﾄﾞはATmega88を目的対象とする先に記述した方法の実装例を与えます。これはLIN制御器によって呼び出され、その説明はこの応用記述の役割ではありません。

```
/*
**
*****************************************************************************************************
**
**
**         Copyright (c) 2004/2005 - Atmel Corporation
**         所有権情報
**
** 企画         : AVR LIN従装置制御器
** 部署         :
** 説明         : ﾋﾞｯﾄ時間計算用捕獲入力割り込みﾙｰﾁﾝ
** 目的対象     : AVR ATmega48/88/168
** ｺﾝﾊﾟｲﾗ       : IAR Embedded Workbench
**
** 版:  日付:        著者:  注釈:
** 1.0  19.08.2004  E.G.   作成
** 1.1  19.11.2004  E.G.   3周期RC校正 (計算されたOSCCALの±差分)
** 1.2  25.11.2004  E.G.   8.0版RC発振器を克服する3段階分割
** 1.3  08.06.2005  E.G.   周波数不連続版に対する不横断
**
** 承諾 -
**
** Atmel - 2004/2005
** 全てのｿﾌﾄｳｪｱ ﾌﾟﾛｸﾞﾗﾑはどんな種類の保証もなしに"現状そのもの"で提供されます。Atmelはどんな目的に対しても提供する
** 素材の適合性を否認します。Atmelは全ての暗黙的な保証、特定目的への適合性、所有権、合法性を含み、提供するｿﾌﾄｳｪｱ
** に関連する全ての保証と条件をここで否認します。このｿﾌﾄｳｪｱ ﾌﾟﾛｸﾞﾗﾑの使用からの結果であっても、どんな間接的、継続的
** な損害、またはどんな損害賠償に対しても、とにかく全くAtmelに責任がないでしょう。
*****************************************************************************************************
**
*/
//
/*___ ｲﾝｸﾙｰﾄﾞ ____________________________________*/
#include "config.h"
#include "lib_mcu/lin_uart/slave_lin.h"
#include "lib_mcu/lin_uart/runtime_calibration_lib.h"

// LIN走行時内蔵発振器校正に対して3周期に分けて克服します。
// 初回=16段, 第2段階=8段, 第3(最終)段階=4段

// 使用資源      : ﾀｲﾏ/ｶｳﾝﾀ1(16ﾋﾞｯﾄ)+捕獲入力単位部
// ｼｽﾃﾑ周波数 : 8MHz内蔵RC発振器

/*___ 定義 ______________________________________*/
volatile U8 Timer_IC_cnt;
volatile U8 _lin_synchronized;          // 設定の場合にLINが主装置と(2%以内で)同期済み。
volatile U16 TimeStamp_IC1;
volatile U16 TimeStamp_IC2;
```

```
//*****************************************************************************
// タイマ/カウンタ1捕獲入力割り込み処理ルーチン
// (最小細動のために)LIN主装置ビット時間値計算に使用
//*****************************************************************************
#pragma vector= TIMER1_CAPT_vect
__interrupt void TIMER1_CAPT_ISR (void){
#ifdef _RUN_TIME_RC_CALIBRATION_ENABLE
  U8 new_osccal;
  U8 osccal_prior_synchr;
  U8 osccal_step;
  U16 measured_master_tbit;
  signed int tbit_diff;

//****************** 第1周回 (窓A) ******************//
  if (Timer_IC_cnt == 0) {                                  // 第1周回先行端で、
    TimeStamp_IC1 = ICR1;                                   // 第1周回先行(第1下降)端計数値取得/保存
  } else if (Timer_IC_cnt == 1) {                           // 第1周回後行端で、
    TimeStamp_IC2 = ICR1;                                   // 第1周回後行(第2下降)端計数値取得/保存
    TCCR1B = (1<<ICES1) | (1<<CS10);                        // 次回用に上昇端捕獲,前置分周なし動作に変更
    measured_master_tbit = TimeStamp_IC2 - TimeStamp_IC1;   // 測定ビット時間取得
    tbit_diff = EXPECTED_TBIT - measured_master_tbit;       // 誤差値取得
    osccal_prior_synchr = OSCCAL;                           // 現OSCCAL値保存
    osccal_step = 16;                                       // OSCCAL変更段数取得

    if ((tbit_diff <= TBIT_DIFF_THRESHOLD_MAX) && (tbit_diff >= TBIT_DIFF_THRESHOLD_MIN)) {   // 範囲内で、
      Timer_stop_capture();                                 // 自動校正停止(T/C1動作停止)
      lin_synchronized = 1;                                 // LIN従装置正常同期済みﾌﾗｸﾞ設定
    } else {                                                // 範囲外で、
      _lin_synchronized = 0;                                // LIN従装置正常同期済みﾌﾗｸﾞ解除
      if (tbit_diff > 0) {                                  // 低すぎるなら、
        new_osccal = osccal_prior_synchr + osccal_step;     // 新OSCCAL値=旧OSCCAL値+16
      } else {                                              // 高すぎるなら、
        new_osccal = osccal_prior_synchr - osccal_step;     // 新OSCCAL値=旧OSCCAL値-16
      }
      if (new_osccal<128 & osccal_prior_synchr>=128) {      // OSCCAL後半使用で新OSCCAL値が128未満なら、
        new_osccal = 128;                                   // 後半部最低値に変更
      }
      if (new_osccal>=128 & osccal_prior_synchr<128) {      // OSCCAL前半使用で新OSCCAL値が128異常なら、
        new_osccal = 127;                                   // 前半部最高値に変更
      }
      OSCCAL = new_osccal;                                  // OSCCAL値更新(次の上昇端までの1ビット時間で安定保証)
    }

//****************** 第2周回 (窓B)******************//
  } else if (Timer_IC_cnt == 2) {                           // 第2周回先行端で、
    TimeStamp_IC1 = ICR1;                                   // 第2周回先行(第2上昇)端計数値取得/保存
  } else if (Timer_IC_cnt == 3) {                           // 第2周回後行端で、
    TimeStamp_IC2 = ICR1;                                   // 第2周回後行(第3上昇)端計数値取得/保存
    TCCR1B &= ~(1<<ICES1);                                  // 次回用に下降端捕獲,前置分周なし動作に変更
    measured_master_tbit = TimeStamp_IC2 - TimeStamp_IC1;   // 測定ビット時間取得
    tbit_diff = EXPECTED_TBIT - measured_master_tbit;       // 誤差値取得
    osccal_prior_synchr = OSCCAL;                           // 現OSCCAL値保存
    osccal_step = 8;                                        // OSCCAL変更段数取得
```

```
    if ((tbit_diff <= TBIT_DIFF_THRESHOLD_MAX) && (tbit_diff >= TBIT_DIFF_THRESHOLD_MIN)) {   // 範囲内で、
      Timer_stop_capture();                                    // 自動校正停止(T/C1動作停止)
      _lin_synchronized = 1;                                   // LIN従装置正常同期済みﾌﾗｸﾞ設定
    } else {                                                   // 範囲外で、
      _lin_synchronized = 0;                                   // LIN従装置正常同期済みﾌﾗｸﾞ解除
      if (tbit_diff > 0) {                                     // 低すぎるなら、
        new_osccal = osccal_prior_synchr + osccal_step;        // 新OSCCAL値=旧OSCCAL値+8
      } else {                                                 // 高すぎるなら、
        new_osccal = osccal_prior_synchr - osccal_step;        // 新OSCCAL値=旧OSCCAL値-8
      }
      if (new_osccal<128 & osccal_prior_synchr>=128) {         // OSCCAL後半使用で新OSCCAL値が128未満なら、
        new_osccal = 128;                                      // 後半部最低値に変更
      }
      if (new_osccal>=128 & osccal_prior_synchr<128) {         // OSCCAL前半使用で新OSCCAL値が128異常なら、
        new_osccal = 127;                                      // 前半部最高値に変更
      }
      OSCCAL = new_osccal;                                  // OSCCAL値更新(次の上昇端までの1ﾋﾞｯﾄ時間で安定保証)
    }

//******************* 第3周回 (窓C)*******************//
  } else if (Timer_IC_cnt == 4) {                              // 第3周回先行端で、
    TimeStamp_IC1 = ICR1;                                      // 第2周回先行(第4下降)端計数値取得/保存
  } else if (Timer_IC_cnt ==5) {                               // 第3周回後行端で、
    TimeStamp_IC2 = ICR1;                                      // 第3周回後行(第5下降)端計数値取得/保存
    measured_master_tbit = TimeStamp_IC2 - TimeStamp_IC1;      // 測定ﾋﾞｯﾄ時間取得
    tbit_diff = EXPECTED_TBIT - measured_master_tbit;          // 誤差値取得
    osccal_prior_synchr = OSCCAL;                              // 現OSCCAL値保存
    osccal_step = 4;                                           // OSCCAL変更段数取得

    if ((tbit_diff <= TBIT_DIFF_THRESHOLD_MAX) && (tbit_diff >= TBIT_DIFF_THRESHOLD_MIN)) {   // 範囲内で、
    Timer_stop_capture();                                      // 自動校正停止(T/C1動作停止)
    _lin_synchronized = 1;                                     // LIN従装置正常同期済みﾌﾗｸﾞ設定
    } else {                                                   // 範囲外で、
      _lin_synchronized = 0;                                   // LIN従装置正常同期済みﾌﾗｸﾞ解除
      if (tbit_diff > 0) {                                     // 低すぎるなら、
        new_osccal = osccal_prior_synchr + osccal_step;        // 新OSCCAL値=旧OSCCAL値+4
      } else {                                                 // 高すぎるなら、
        new_osccal = osccal_prior_synchr - osccal_step;        // 新OSCCAL値=旧OSCCAL値-4
      }
      if (new_osccal<128 & osccal_prior_synchr>=128) {         // OSCCAL後半使用で新OSCCAL値が128未満なら、
        new_osccal = 128;                                      // 後半部最低値に変更
      }
      if (new_osccal>=128 & osccal_prior_synchr<128) {         // OSCCAL前半使用で新OSCCAL値が128異常なら、
        new_osccal = 127;                                      // 前半部最高値に変更
      }
      OSCCAL = new_osccal;                                  // OSCCAL値更新(次の上昇端までの1ﾋﾞｯﾄ時間で安定保証)
      Timer_stop_capture();                                 // 自動校正停止(T/C1動作停止)
      _lin_synchronized = 1;                                // LIN従装置正常同期済みﾌﾗｸﾞ設定
    }
  }

  Timer_IC_cnt ++ ;                                         // 状態番号進行
#endif
}
```

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AVR140応用記述(doc7653.pdf Rev.7653A-09/06)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。