簡単ロジック・スコープ用
テスト・ツール
ユーザーズ・ガイド

初版

2009年7月

よし　ひろし

目次

1 はじめに

このドキュメントでは、簡単ロジック・スコープの動作を確認したり、使い方を学習するためのツールについて説明します。

（1）周期テスト
ロジック・スコープは、内部的にはソフトウェアのタイミングでサンプリング周期を決定しています。
そのために、サイクル数のちょっとした違いでサンプリング周期に誤差が出ます。
ファームウェアに手を加えたときには、周期テスト・ファームウェアで生成する信号を測定してサンプリング周期に著しい違いが無いことを確認します。

（2）TCN75A 温度計
I2C 通信とシリアル通信の動作確認を通して、ロジック・スコープの使い方について、少しだけ触れます。

2 周期テスト信号発生器

2.1 動作

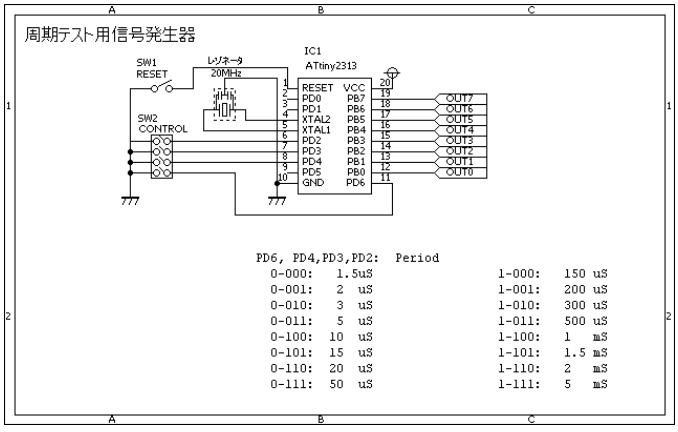
指定したクロック周波数で動作する 8 ビットカウンタです。
ここでは、1.5uS を周波数に直すと半端な数値となるので、クロック周期を用います。
4 ビットの DIPS スィッチの設定によって、クロック周期を変更します。
上位２ビットが周波数マルチプライアで、下位２ビットが周期です。
出力信号の最下位ビット（PB0）が、ＤＩＰスィッチで指定した周期で反転します。上位ビット（~PB7）はその周期の倍、倍、と長くなっていきます。

出力信号の周期

下位２ビット	設定値	上位２ビット			
		0	1	2	3
		×１μS	×１０μS	×１００μS	×１ｍS
０　００	１．５	１．５μＳ	１５μS	１５０μＳ	１．５ｍS
１　０１	２．０	２．０μＳ	２０μS	２００μＳ	２．０ｍS
２　１０	３．０	３．０μＳ	３０μS	３００μＳ	３．０ｍS
３　１１	５．０	５．０μＳ	４０μS	５００μＳ	５．０ｍS

注：ＤＩＰスィッチの設定を動作に反映させるには、
　①リセットする
　②ＤＩＰスィッチの下位２ビットを最後に変更
のどちらかを行ってください。

2.2 回路図



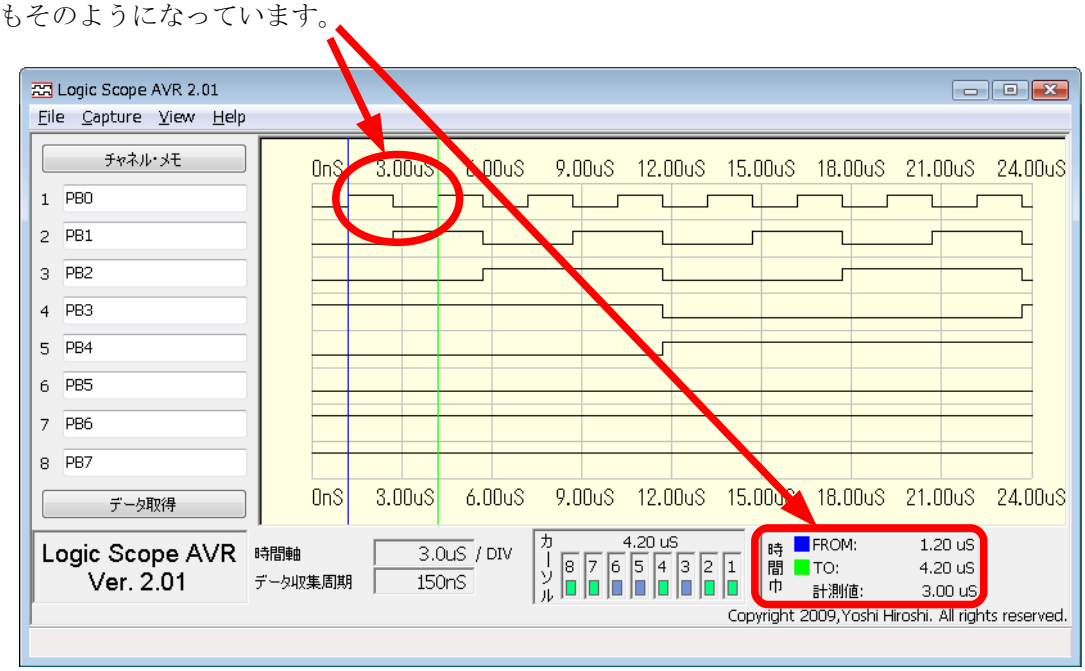
注：ＤＩＰスィッチの変更取込みは、PD2、PD3 の外部割込みを利用しているので、上位 2 ビットの変更を自動で取込むことはできない。

2.3 使い方

ＰＢ０～ＰＢ７の出力にプローブをあてて、信号を測定します。

プローブ ＣＨ．１　→　テスト信号発生器のＯＵＴ０
プローブ ＣＨ．２　→　テスト信号発生器のＯＵＴ１
プローブ ＣＨ．３　→　テスト信号発生器のＯＵＴ２
プローブ ＣＨ．４　→　テスト信号発生器のＯＵＴ３
プローブ ＣＨ．５　→　テスト信号発生器のＯＵＴ４
プローブ ＣＨ．６　→　テスト信号発生器のＯＵＴ５
プローブ ＣＨ．７　→　テスト信号発生器のＯＵＴ６
プローブ ＣＨ．８　→　テスト信号発生器のＯＵＴ７

例えば、周期設定を１．５μＳとしたとき、次のような測定結果を得られます。
信号発生器の最下位ビットは１．５μＳで反転するので、１サイクル３μＳです。
計測結果もそのようになっています。



ここでは、１周期３μＳですから、１５０ｎＳでサンプリングしたデータが２０サンプル必要です。
このとき、もしロジック・スコープのサンプリング周期が１クロック余分にかかっていた場合、すなわち、１５０ｎＳのサンプリング周期が実際は２００ｎＳになっていた場合、２０サンプル分すなわち４μＳのデータを３μＳとして表示します。
実際の３μＳのデータは３μＳ÷２００ｎＳ＝１５サンプルなので、３μＳ×（１５÷２０）＝２．５μＳとなりますが、１５０ｎＳの整数倍になっていないので、それに近い値として表示されます。
（実際の話はもっとややこしいです。そもそもテスト信号の反転周期が２００ｎＳの整数倍になっていないため測定された波形のＤＵＴＹは５０％ではない上に、測定上の誤差も加わっています。なので、以下省略。）

ロジック・スコープとテスト信号発生器のクロック周波数の誤差も測定には影響しますので、「そこそこ」の精度を確認する程度にご使用ください。

3 TCN75A 温度計

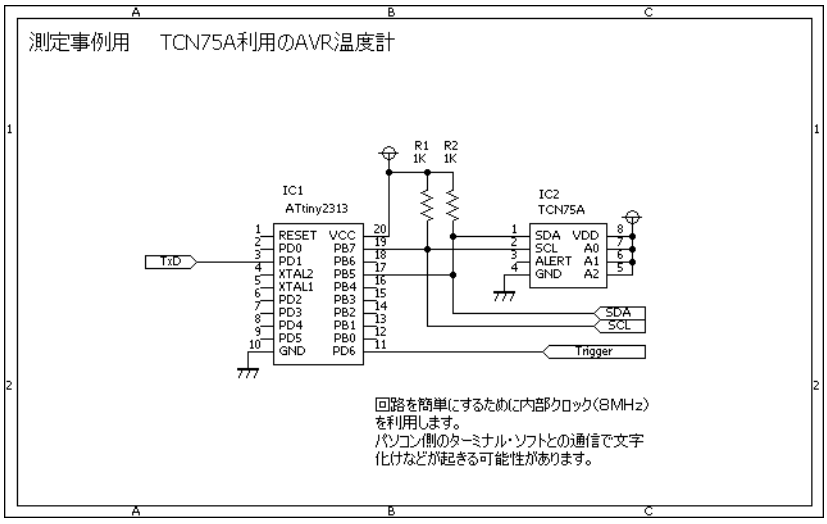
3.1 動作

I2C 通信で接続した TCN75A で温度を計測し、測定結果を文字列に編集してシリアル通信経由で送信します。

項番	項目	概要
1	測定周期	およそ 500mS
2	I2C 通信速度	100Kbps
3	シリアル通信速度	38.4Kbps
4	送信メッセージ・フォーマット	“xx.x\r” 計測値を摂氏の 0.1 度まで表示。 文字列の最後は改行コード （0x0D）で終端。

シリアル通信速度の変更は、UART. h で定義してある BAUD を変更して下さい。

3.2 回路図

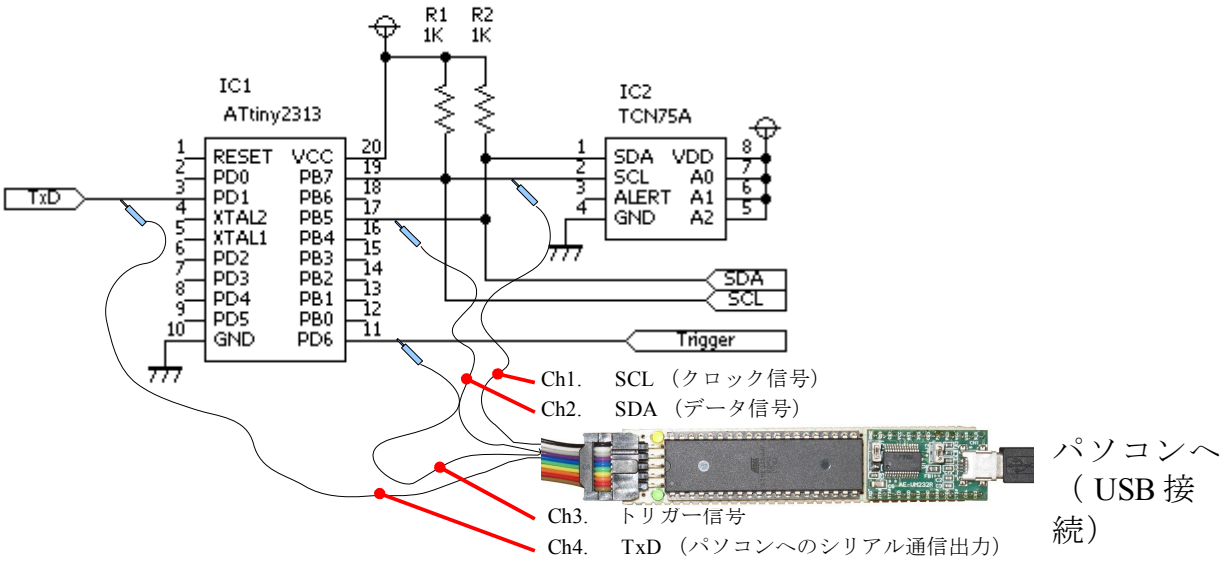


レゾネータなどを使ってクロック周波数を変更した場合、usitwi.h で定義してある
#define F_CPU　　8000000
を変更して下さい。
ここで、8000000 はクロック周波数です。例えば 20MHz にしたときは、
#define F_CPU　　20000000
です。

3.3 使い方

3.3.1 準備

ファームウェアには、I2C 通信を開始する前に、PD6 を 1 にセットするように細工を入れます。
プローブを次のようにあてて、信号を測定する準備をします。
I2C は 2 ワイヤでの通信なので、２本だけ接続すればデータは取れますが、トリガー直前のデータを取得できないので、PD6 にトリガーのための信号を出力するようにします。



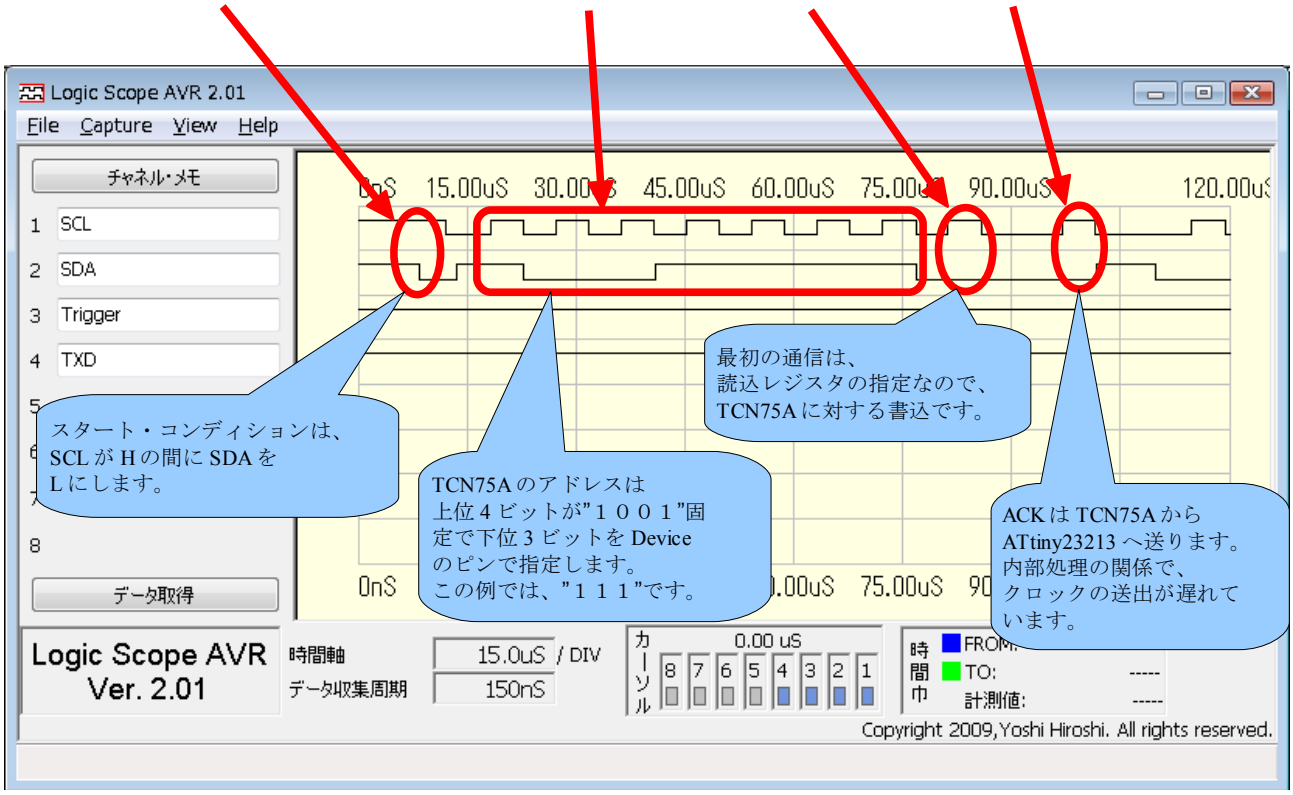
3.3.2 トリガの設定

500mS 間隔でシリアル通信を含めても数 mS しかデータが出力されないので、データが出力されているときだけピンポイントでデータを収集します。
そのためには、SCL や SDA 信号線を使う方法が一番簡単ですが、簡単ロジック・スコープではトリガ条件が成立してからデータ収集を開始するので、トリガ直前までのデータ（ならびにトリガに使用したデータも）は取込めません。
そのために、特別な信号（Trigger：PD6）を用意しました。この信号線が”１”になったときにデータ収集を開始します。
チャネル４~５はいらないので、チャネル番号のチェック・ボックスををクリックして「収集しない」としておきます。

3.3.3 データの解析－その１

見やすい大きさになるよう、ズーム機能で拡大します。”↑” 矢印キーでズームインです。
サンプリング周期を 150nS としたので、かなり詳細なデータをとることができました。
まずは。DeviceAddress の送出部分です。

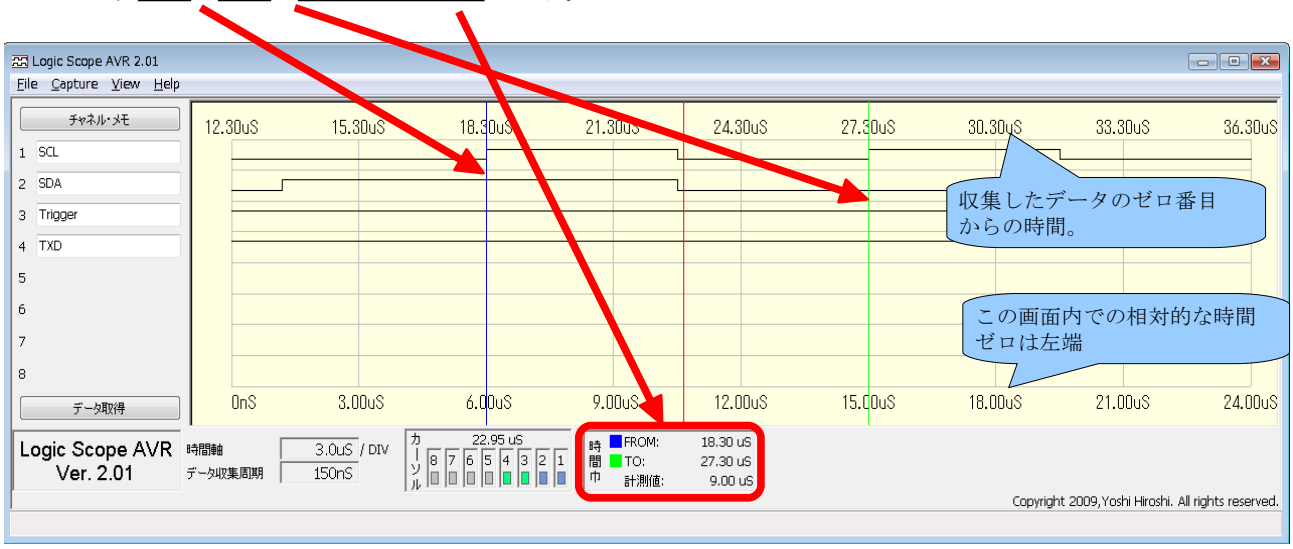
スタート・コンディションや、アドレスの出力状況、R/W ならびに ACK もはっきりと確認できます。



ちょっと見て、信号がそこそこ出ていて、それなりに通信ができていると、とりあえず安心します。
しかし、ACK 前後でデータを送出する側が入れ替わるタイミングでひげが出るのが気になります。また、
ここでは説明しませんが、ストップ・コンディションを送るときも、ちょっとひげが出て、気になります。
などなど、いろいろな問題を見つけ出すためにも、このようなツールは重要な役割を果たします。

3.3.4 データの解析－その２

動作は大体合っていましたが、100Kbps のつもりで作ったタイミングはどうでしょう・。
クロック巾を測定してみます。
時間幅を計測しやすいようにさらに拡大します。また、拡大していくと測定対象が画面の外側に出てしまうので、スクロールして画面内に収めます。
下の図で、青線と緑線の時間間隔を測定します。



クロックピリオドが 9μS では、ちょっと違反しています。でも、TCN75A は 400Kbps 対応なので、最小クロックピリオドは 2.5μS となっているので動作していますが、100Kbps までしか対応していないデバイスではトラブルを起こします。低速デバイスはクロックの引き延ばしで対応できるはずですが、うまく動作しないこともありますので、要注意です。

3.3.5 I2C 通信の不良例

TCN75A との I2C 通信がうまくいかないときは、TCN75A からの ACK が戻らないので、次のような信号が出力されます。
TCN75A のアドレス設定が、ファームウェアの設定と異なる、あるいは、TCN75A が接続されていない場合にこのような状態になります。

読込レジスタのアドレス書込みに失敗（最初のシーケンス。書込み要求時のアドレス送出で ACK が返らない)したにもかかわらず、読込を行おうとして、こちらもアドレス送出で失敗している。
最初の書込みエラーで処理を中止せずに読出しのシーケンスを起動するという、内部ロジックの手抜きが露見してしまいました。