デジタルノギス データ取り込み
７セグメントＬＥＤ表示 PIC基板設計
Type2 対応

目　　次

[1]　経緯、概要：　3

[2]　4pin（リセット信号）の扱い：　4

[3]　Type2ノギスの隠しコマンド？：　5

[4]　データ、クロック信号の違い：　6

[5]　データフォーマット：　7

[6]　ノギス、PICマイコン間ケーブル：　8

[8]　PICマイコン基板回路図：　9

[9]　7SegmentLED表示基板回路図：　10

[10]　パーツリスト：　11

[11]　PICポートレジスタマップ：　12

[12]　7Segment LED のドライブ：　12

[13]　7Segment LED 各桁のドライブ：　13

[14]　作り出して分った事：　13

[15]　追加した機能、削った機能：　14

[16]　ノギス表示部分の信号引き出し：　15

[17]　ケーブルを接続したところ：　17

[18]　今回作成した基板：　17

[19]　ソフト開発環境について：　18

[1] 経緯、概要：

当初 扱っていたデジタルノギス（150mmのノンブランドデジタルノギス）とは 異なる通信仕様のデジタルノギスが市場にはいくつか存在すると思われます。 たまたま異なる通信仕様のものが手元にありましたので、これにも対応する事にします。

便宜上これら２つを区別するために、**最初に扱った通信仕様の物を Type1** と呼ぶことにします。　今回、**2回目に扱う物を Type2** と呼ぶことにします。

今回の Type2は 下の[画像.1]のようなものです。　ジョー（はさみ口）のところが無くて小型工作機械等に取付ける用途で作られたものと思われます。

Type1、Type2ともにインタフェースコネクタ部分は 同形状の４ピンのエッジコネクタです。　当初、LCD表示部カバーが Type1と全く同じ形状なので Type1と 同じインタフェース仕様と安易に思い込んでました。　コネクタを取付け信号を出してつないでみて動かないので、信号を測定して全く異なる通信仕様のものであることを認識しました。

まず、左から 4番目のピンの結線が異なります。

Type1の 4pinは、GND、　Type2の 4pinは、電池の 1.5Vが出ています。
何で、1.5Vが出ているのか疑問に思いました。

電池無しで外部から 1.5Vを供給する目的なのか。？　と思ったら別に理由がありました。
外部での [RESET]ボタン機能です。　何と、Type2のノギスは、3pinのクロック信号を Hi(1.5V)に吊り上げる事により、リセットの機能を実現していたのです。

それと Type1と Type2では、データ転送速度が全く異なります。
まず Type2は、データ転送のクロック周期が　およそ 13[μs]と かなり高速です。
データ転送のインターバルは、約 320[ms]です。

転送インターバルが十分長いので、こんなに早く転送する必要ないのにと思いますが Type1と異なり、Type2は、データ転送する時だけ、2pin（データ信号）と 3pin（クロック信号）に 電圧を出します。　つまり、データ転送を高速にする事により、インタフェース端子に電圧を出している時間を極力短くして外部に流れ出る電流を抑え、電池の消耗を抑えているのではないかと考えます。
（ 受け側のマイコンにとっては、13[μs]ちょっと厳しい仕様となります。）

データの受け側（表示器）のハードに影響するのは、4pinの扱いと、クロック速度ということになります。

データのフォーマットも Type1と Type2では、全く異なります。
データフォーマットはハードには影響しませんので、フォーマットを調べて、それに合うソフトを作ることになります。

今回作成するデジタルノギス データ表示器は、可能な限り Type1 と Type2 の両方で使える事を目指します。
ショートプラグの差し換え、DIP-SWの設定等切り替え操作は 多少必要になると思います。

[画像.1]

[2] 4pin（リセット信号）の扱い：

まず Type1、Type2ノギスのエッジコネクタの信号線を右に示します。

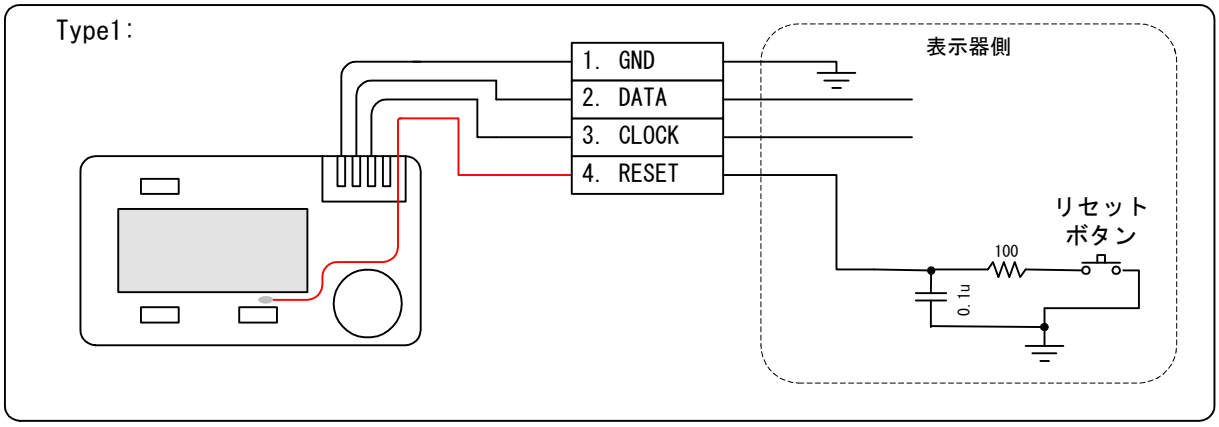
よそのホームページにて、DATAと CLOCKが入れ替わっているノギスが紹介されているのも確認しました。 入れ替わっているだけならケーブルのピンを差し換えるだけで済みますけど...。

ノギス エッジボードコネクタの
ピンアサイン

pin	Type1	Type2
1	GND	GND
2	DATA	DATA
3	CLOCK	CLOCK
4	GND	Vbb+

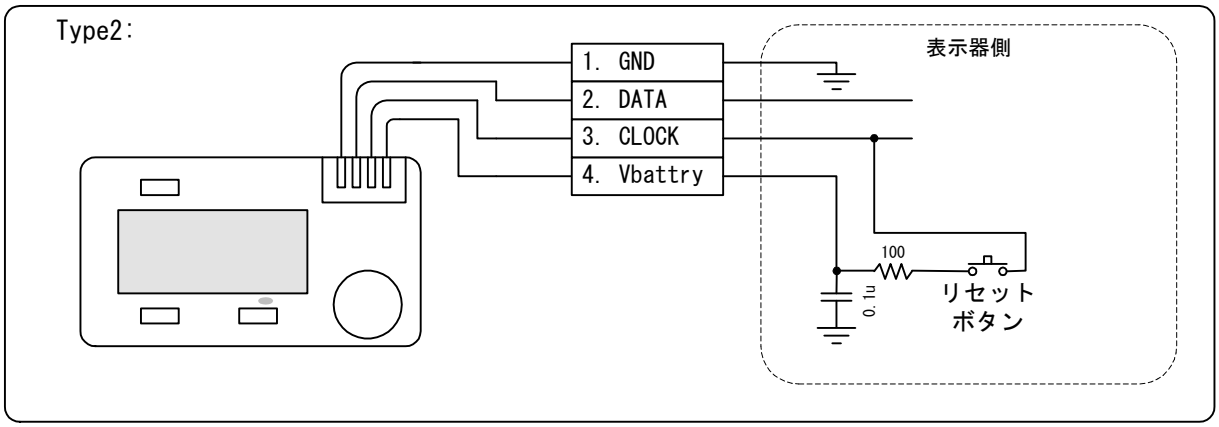
Type1では、リセット機能を外部で実現するためノギスのリセットボタン部分のパターンから信号を引き出し、それを 4pinの信号として扱いました。　表示器側では、4pinの信号をスイッチで GNDに落とす事により リセット機能を実現しました。

Type1:



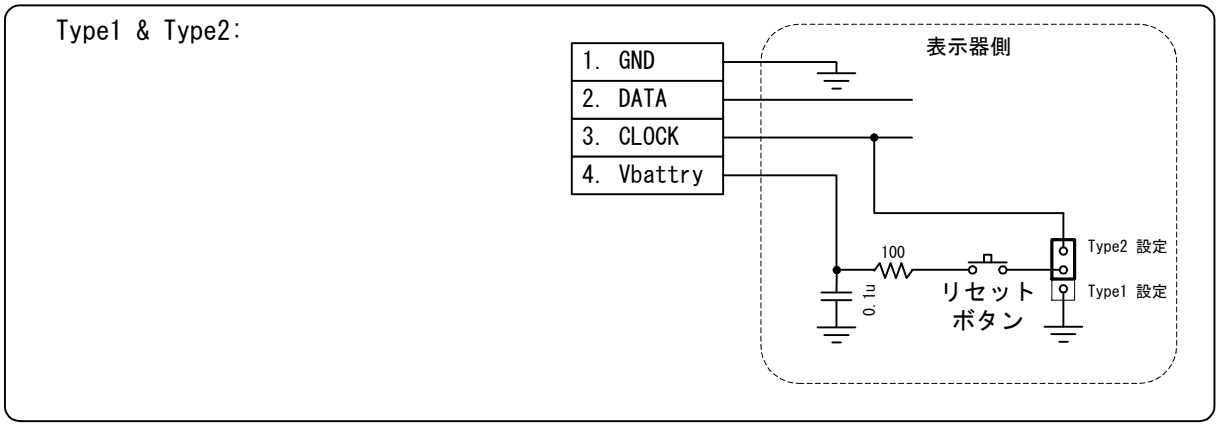
Type2ではノギス側にてエッジコネクタの 4pin（1.5V）を、そのまま表示器に持って行く事になります。　表示器側では、3pinのクロック信号を スイッチで 4pinにショートする事により リセット機能を実現します。

Type2:



表示器側で、Type1 及び Type2の両方に対応するには、以下のように配線して、ショートピンで設定する形にします。

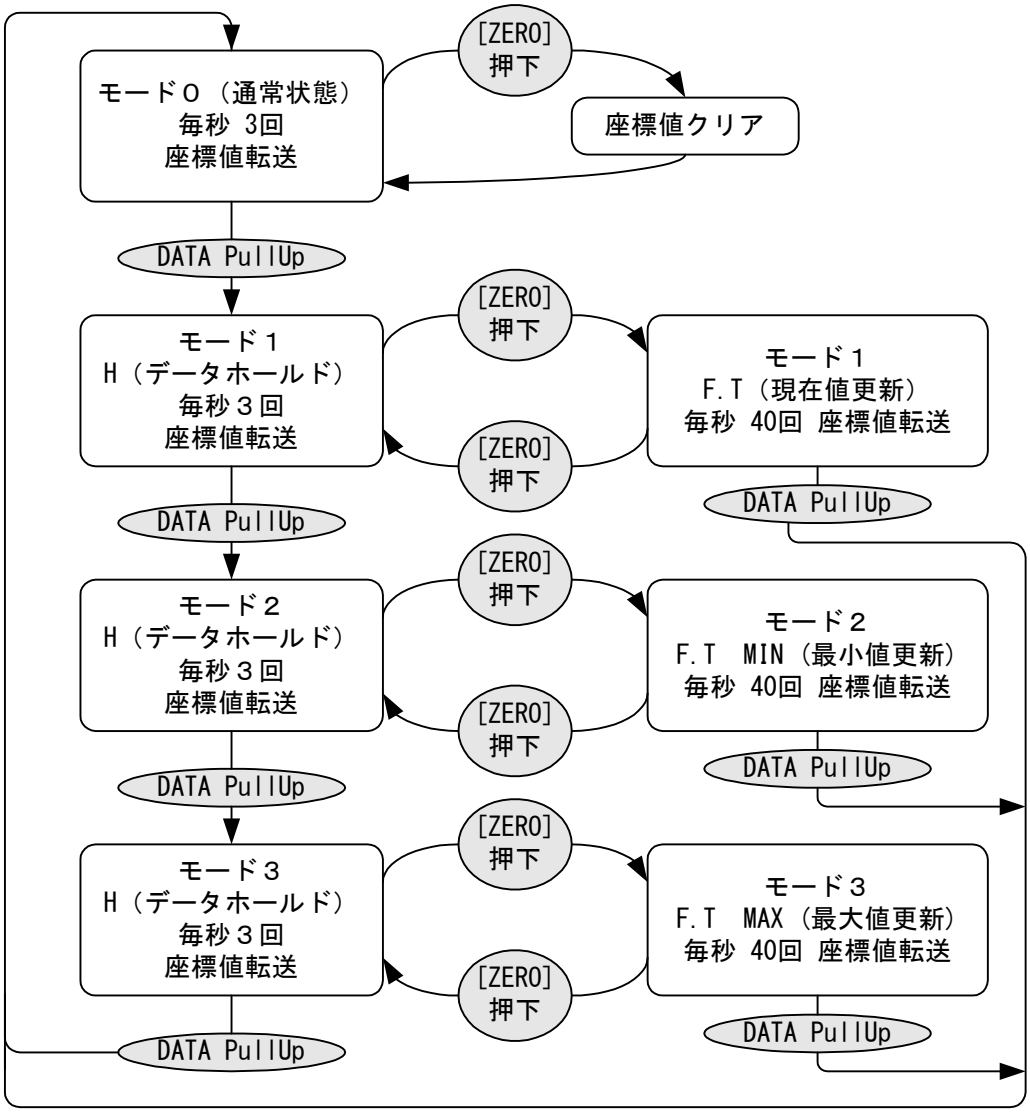
Type1 & Type2:



[3]　Type2ノギスの隠しコマンド？：

今回のType2ノギス対応表示器のプログラムを作っている時、ちょっと気になっている事がありました。　　リセットを 実現するのに CLOCK信号を 電池電圧に PullUp しました。
であれば、DATA信号を PullUpしたらどうなるのだろうと... 。？

この件については、最後に書こうかと思ってましたが、ノギスの機能という事で最初に持ってきました。
PICマイコン基板にタクトスイッチを追加して、DATA線を PullUpする実験の準備をしました。
やってみたところ、最初 何が起こったのだろうと思ってしまいました。
ちょっと悩みましたが、DATA線を PullUpすると、モードの切り替え機能があることがわかりました。
通常の状態を モードＯとします。　この状態でノギスの[ZERO]ボタンを押すと座標値が0.00にクリアされます。　1回、DATAを PullUpすると
モード１に移行します。　ノギスの LCD表示器に H が表示されます。
この状態では 値を ホールドしています。 この状態で [ZERO]ボタンを押すと LCD表示器にFHが 表示され高速（およそ 毎秒 40回）で データ転送が行われます。
応答の速い表示が必要な場合は便利と思います。
但し、最下位行（0.01mmの桁）が 多少ぶれる傾向があります。　それと、FHモードは多分電池の消耗が早いと思われます。　更に 1回、DATAを PullUpするとモード２に移行します。
ちょっと分かりにくいので図示すると以下のようになります。



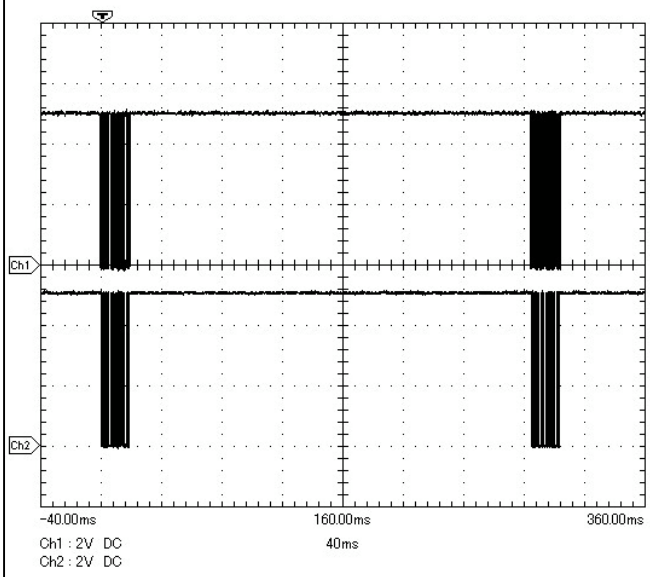
[4] データ、クロック信号の違い：

Type1と Type2の違いですが、Type1は、OFF時（液晶非表示）は、データを出しません。データは出しませんが、DATA、CLOCK信号は Hi(1.5V)状態です。

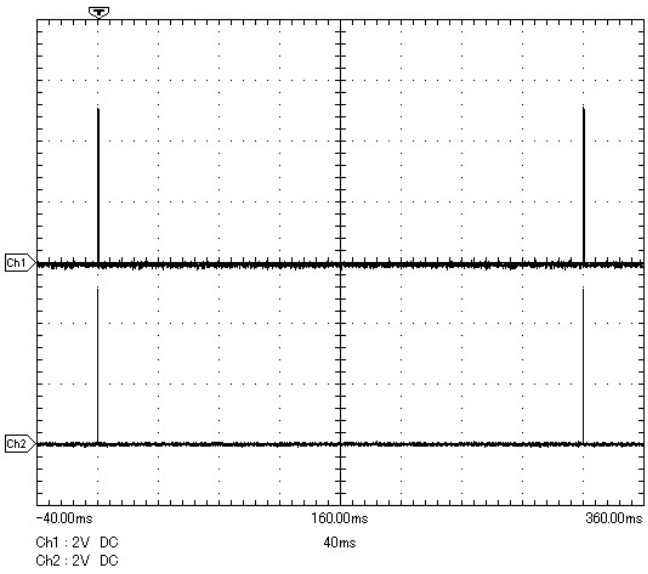
Type2は、OFF状態（液晶非表示）でも、データを出し続けます。　しかしデータを出してない期間は、Lowです。　これが大きな違いの一つです。

Type1、Type2共に、データを送信する間隔は、0.3秒ほどです。
Type2は、１パケットのデータ転送速度が速いので細く見えます。

Type1ノギス DATA CLOCK コンパレータ出力



Type2ノギス DATA CLOCK コンパレータ出力

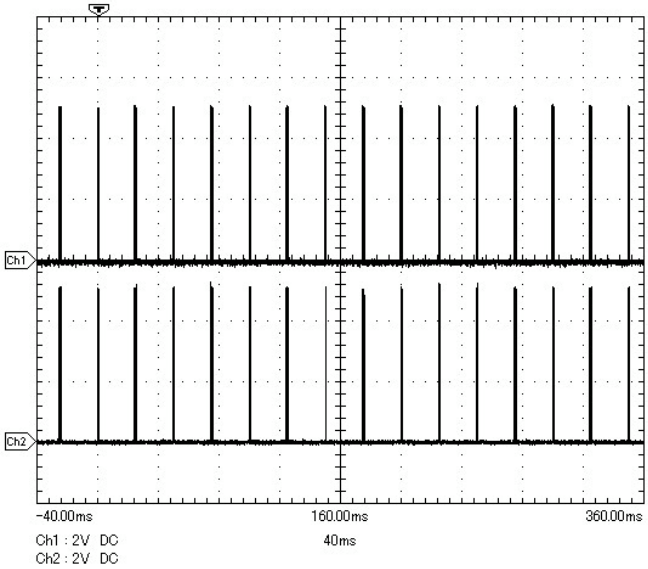


Type2は、FTモードにすると データを毎秒40回ほど転送します。　右のグラフからも頻繁にデータパケットが出ているのが分ります。　このモードで使うとノギスの動きに高速に追従して表示が行われます。

0.01mmの桁が ±1digitバタつきやすい感じですがそう問題はないと思います。

通常モードは送信間隔が 320ms程度に対してFTモードは、送信間隔が 約 25msです。

Type2ノギス FTモード DATA CLOCK コンパレータ出力



ちなみに、時間軸を拡大してクロック周期の短いところを測定したら
Type1が 690us 、Type2が 13usでした。
Type2のクロック周期を PICマイコンの割込み処理で受けようとすると処理時間的に厳しいので
PIC18Fシリーズを用い CPUクロック 40MHzで動作させ、プログラムはアセンブラで記述する事にしました。（ 18Fシリーズは、命令数が 72に増えているので多少はアセンブラが作りやすくなってます。）

[5] データフォーマット：

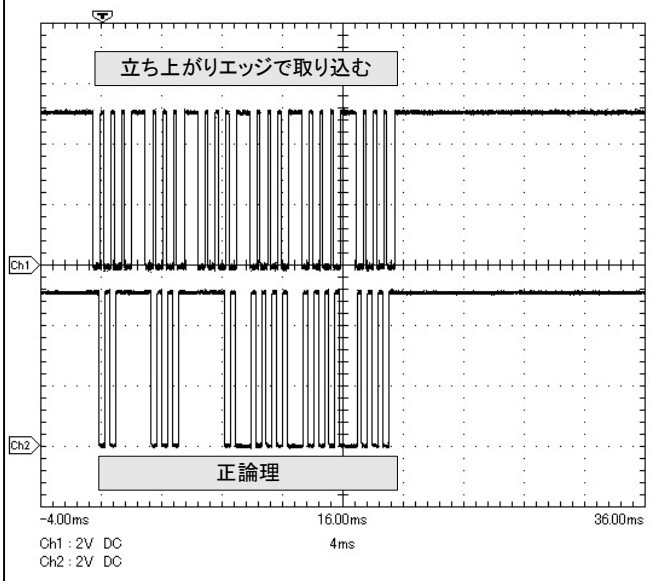
下のグラフは、１パケット全体がちょうど入る時間軸で測定した物です。
（ 上が クロック、下が データ です。 ）
Type1が 4ms/div、Type2が 100us/divで 転送時間の違いが分ると思います。

見た目の波形が Type1と Type2で全然異なりますが、まず１パケットで転送されるデータ量が異なります。 Type1のクロックは、24bit分です。 Type2のクロックは、48bit分です。
それとクロックは、データを取り込むタイミングを規定しますが、Type1では、クロックの立ち上がりタイミングでデータを取り込んでいました。
しかし、Type2ノギスは、クロックの立ち下がりタイミングでデータを取り込みます。

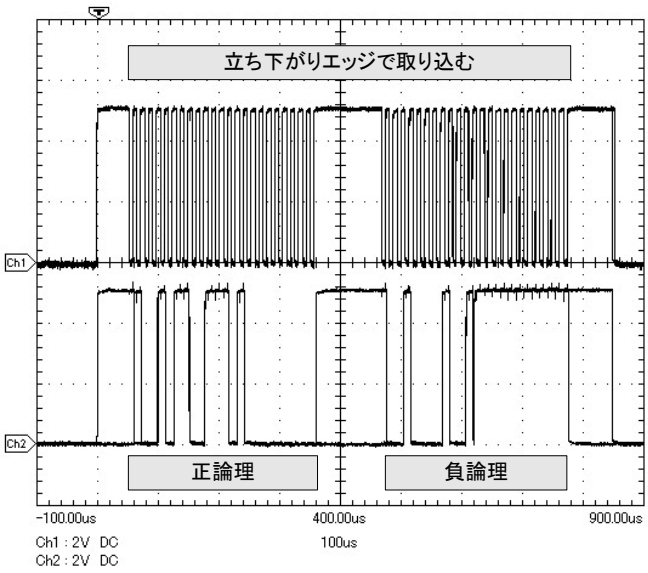
Type1ノギスのデータ線は 正論理でデータを送信します。 Type2ノギスのデータ線は変則的なのですが 前半24bitは正論理で、後半24bitは負論理です。

ちなみに前半24bitが 絶対座標値、後半24bitが 相対座標値となります。
相対座標値は、[ZERO]ボタンを押した位置が 原点になります。
絶対座標値は、[ZERO]ボタンで原点設定出来ません。
最初、定規が固定的な絶対値を持っているのかと思ってましたが、どうもそうではなく電池で保持されてるだけのようです。 電池を外して入れ直すと値が変わります。
電池を入れた位置が原点になるみたい... ？ です。

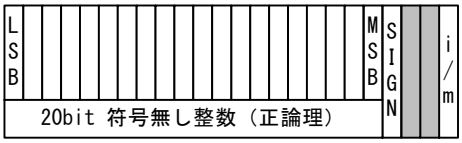
Type1ノギス DATA CLOCK コンパレータ出力



Type2ノギス DATA CLOCK コンパレータ出力

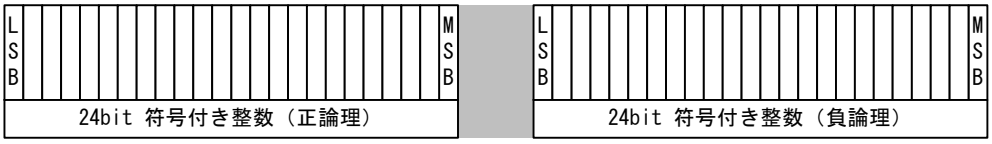


Type1 データフォーマット 24bit



Type1のデータは、1/100mm単位または、1/2000inch単位で送って来ます。最終の i/mbitが 1の時インチ単位のデータとなります。マイナスの値は２の補数表現ではなく、絶対値と SIGN=1 で マイナスの値を表現します。

Type2 データフォーマット 48bit



Type2のデータは、絶対値、相対値共に １インチ 5000H（20480）の単位で送って来ます。
表示器で、1/100mm単位の表示、または 1/1000inch単位の表示を行うには、受けた表示器側で単位変換の演算処理が必要になります。 負の値は、２の補数表現です。

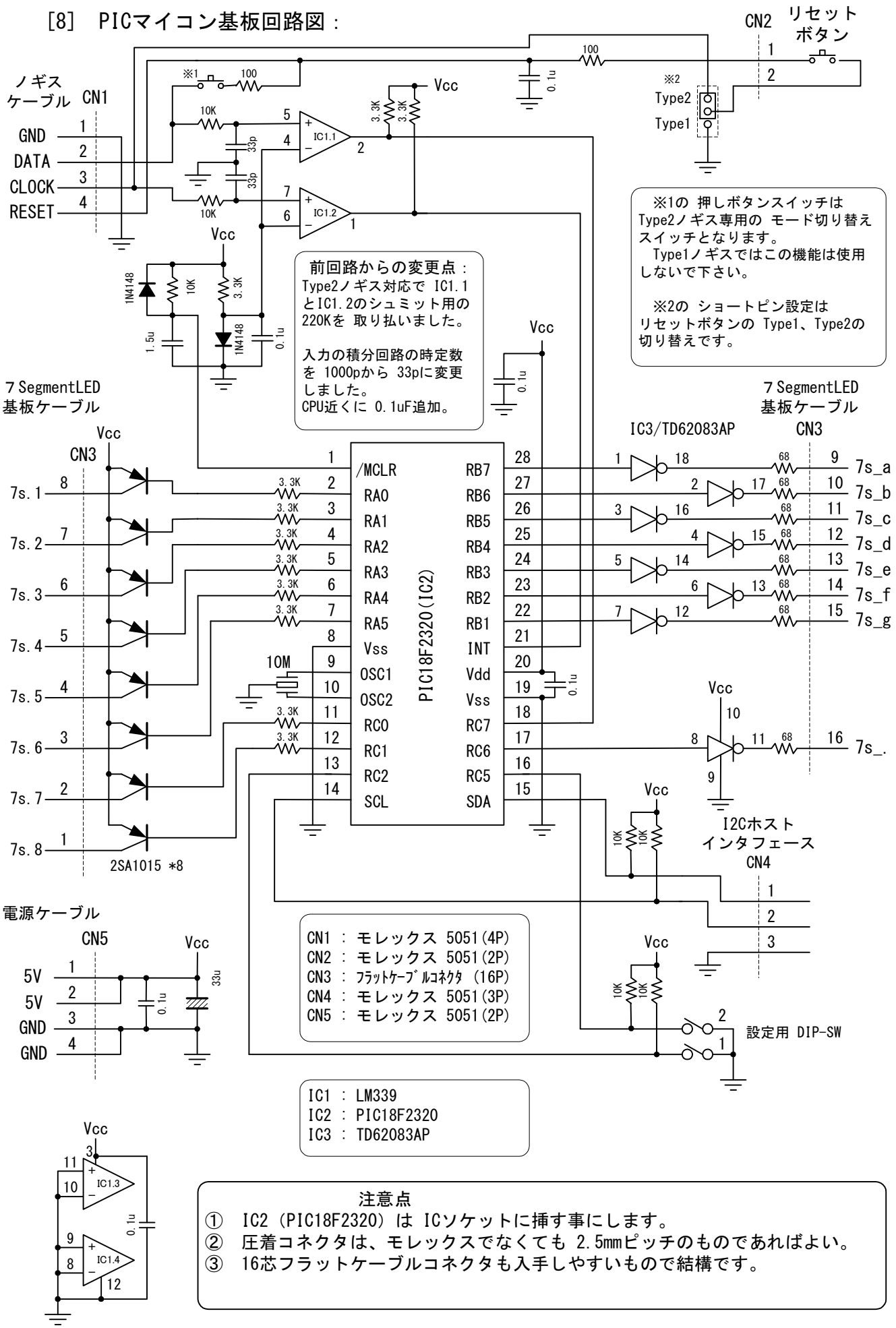
[6] ノギス、PICマイコン間ケーブル：

ケーブルは、前バージョンと 同じです。

[7] 使用PICマイコン（ PIC18F2320 ）ピンアサイン：

マイコンの I/Oピンアサインは基本的に、前バージョンと 同様です。
セラミック振動子が **10MHz**になります。（ PIC内PLLで4倍の 40MHzになります。）

[8] PICマイコン基板回路図：

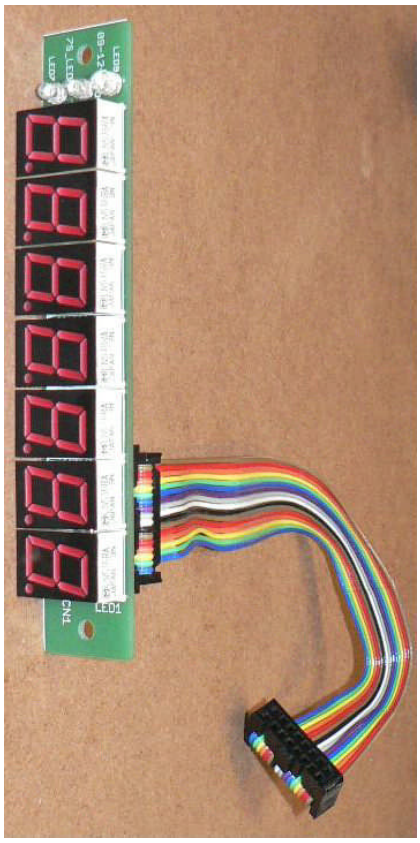


注意点

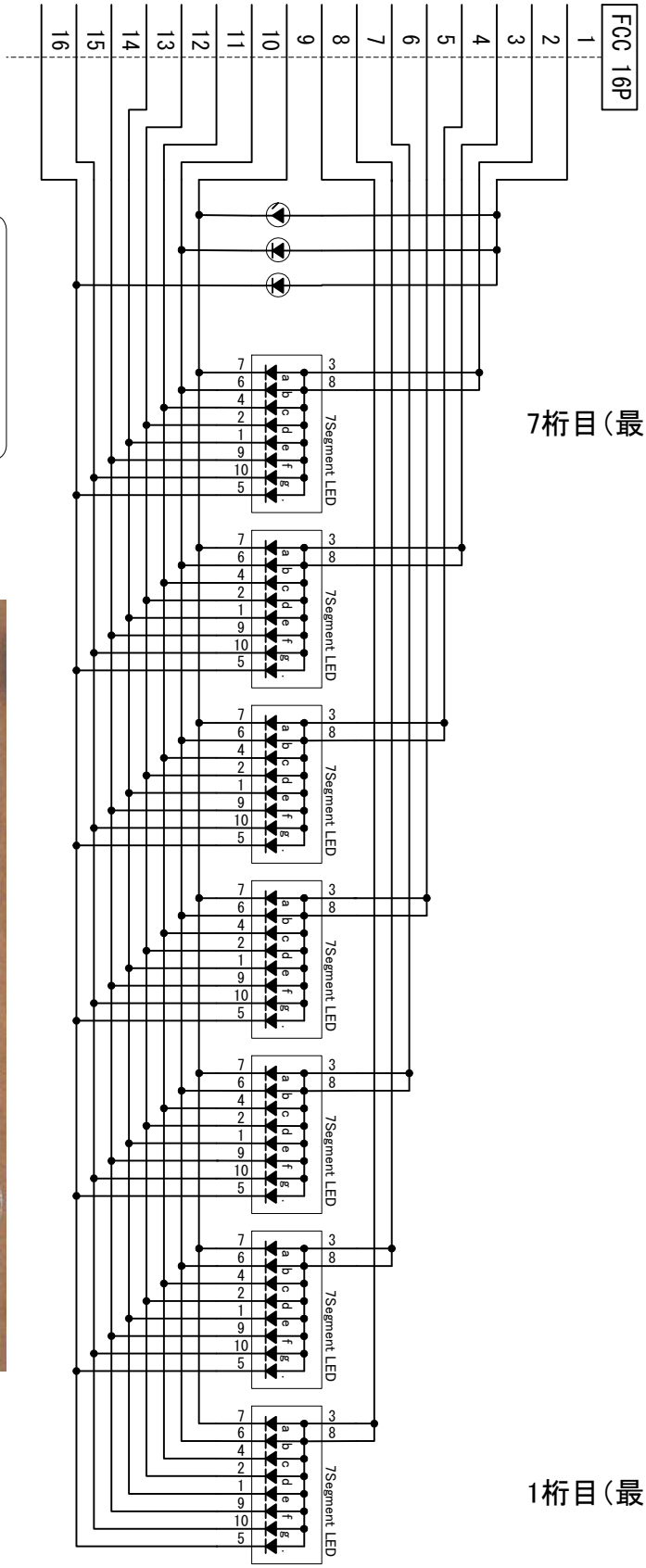
① IC2（PIC18F2320）は ICソケットに挿す事にします。
② 圧着コネクタは、モレックスでなくても 2.5mmピッチのものであればよい。
③ 16芯フラットケーブルコネクタも入手しやすいもので結構です。

[9] 7SegmentLED表示基板回路図：

正面から見て左端に 3mmLED３個が来るように配置してます。
3mmLED３個は、縦配置で上から
青（9番接続）
黄（10番接続）
赤（16番接続）
にしています。



使用 7Segment LED
LN516RA（赤）



[10] パーツリスト：

部品購入先　秋月電子通商
サトーパーツ
オヤイデ電気

PICマイコン基板

No.	部品名	メーカー名	値	個数
1	カーボン抵抗 1/4W		100	2
2	カーボン抵抗 1/4W		68	8
3	カーボン抵抗 1/4W		3.3K	11
4	カーボン抵抗 1/4W		10K	7
5	IC1	NS	LM339	1
6	IC2	MicroChip	PIC18F2320	1
7	IC3	東芝	TD62083AP	1
8	PNPトランジスタ	東芝	2SA1015	8
9	ダイオード		1N4148	2
10	セラミック発振子		10MHz	1
11	セラコン		33p	2
12	積層セラコン		0.1uF	6
13	積層セラコン		1.5uF	1
14	電解コン		33uF	1
15	DIP-SW		2P	1
16	ICソケット		28P	1
17	コネクタ CN1,CN5	モレックス	5045/4P	2
18	コネクタ CN2	モレックス	5045/2P	1
19	コネクタ CN3	ﾌﾗｯﾄｹｰﾌﾞﾙ用	16P	1
20	ﾌﾗｯﾄｹｰﾌﾞﾙ ｹｰﾌﾞﾙ側ｺﾈｸﾀ		16P	1
21	コネクタ CN4	モレックス	5045/3P	1
22	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN1,CN5	モレックス	5051/4P	2
23	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN2	モレックス	5051/2P	1
24	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN4	モレックス	5051/3P	1
25	ｺﾈｸﾀ ピン	モレックス	5159	13
26	タクトスイッチ（小）			1
27	基板			1

7Segment LED表示基板

No.	部品名	メーカー名	値	個数
1	7Segment LED		LN516RA	7
2	φ3mm LED 緑			1
3	φ3mm LED 黄			1
4	φ3mm LED 赤			1
5	ﾌﾗｯﾄｹｰﾌﾞﾙ基板側ｺﾈｸﾀ		16P	1
	ﾌﾗｯﾄｹｰﾌﾞﾙ ｹｰﾌﾞﾙ側ｺﾈｸﾀ		16P	1
6	基板			1

ノギス付加部品

No.	部品名	メーカー名	値	個数
1	本体側コネクタ	モレックス	53253	1
2	ケーブル側コネクタ	モレックス	51065	1
3	コネクタピン	モレックス	50212-8100	4

配線材、その他

No.	部品名	メーカー名	値	個数
1	極細４芯シールドケーブル			1m
2	16芯フラットケーブル			少々
3	その他、0.3SQ程度の配線材			少々

[11] PICポートレジスタマップ：

	b7	b6	b5	b4	b3	b2	b1	b0
PORTA	b7	b6	7seg 6桁目	7seg 5桁目	7seg 4桁目	7seg 3桁目	7seg 2桁目	7seg 1桁目
			(DO)	(DO)	(DO)	(DO)	(DO)	(DO)
PORTB	7seg (a)	7seg (b)	7seg (c)	7seg (d)	7seg (e)	7seg (f)	7seg (g)	ノギス Clock (INT)
	(DO)	(DO)	(DO)	(DO)	(DO)	(DO)	(DO)	
PORTC	ノギス Data	7seg (,)	DIP-SW 2	I2C SDA	I2C SCL	DIP-SW 1	3LED	7seg 7桁目
	(Di)	(DO)	(Di)			(Di)	(DO)	(DO)

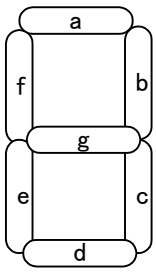
ノギスClockは、PORTB.0を INT信号（Type1=立ち上がりエッジ、Type2=立ち下がりエッジ）として受ける。　INT信号の割り込み処理にて、ノギスDataを読み取る。

PORTAの b6, b7は セラミック振動子を接続するためI/Oポートとして使用できない。
PORTCの b4, b3は I2Cの通信に使用するため I/Oポートとして使用出来ない。

[12] 7Segment LED のドライブ：

今回使用する、7Segment LEDは、アノードコモンの LN516RAである。
複数の 7Segment LEDをダイナミック点灯させる。　8桁あれば、明るさは連続点灯の1/8に減少するので、多少多めに電流を流す必要がある。
PICでの直接ドライブは無理があるので、アノード側 カソード側にドライバを入れる。
a, b, c, d, e, f, g, .の各セグメントは、Lowに落とす事になるので、8bitのトランジスタアレィにて駆動する。　アノード側は 各桁を指定する信号になる。
アノード側は、Hiに引き上げる事になるが PNPのトランジスタアレィは出回って無いのでPNPトランジスタ（ 2SA1015 ）を桁数分並べる事にする。

0～9の数字を点灯させるためには、7SegmentLEDの どの Segmentを点灯させれいいかを整理しておく。
Segment側は Hi Active（ 正論理 ）となる。



値	点灯セグメント	出力データ
0	a, b, c, d, e, f	PORTB = 0xFC;
1	b, c	PORTB = 0x60;
2	a, b, d, e, g	PORTB = 0xDA;
3	a, b, c, d, g	PORTB = 0xF2;
4	b, c, f, g	PORTB = 0x66;
5	a, c, d, f, g	PORTB = 0xB6;
6	a, c, d, e, f, g	PORTB = 0xBE;
7	a, b, c	PORTB = 0xE0;
8	a, b, c, d, e, f, g	PORTB = 0xFE;
9	a, b, c, d, f, g	PORTB = 0xF6;

小数点Segmentの点灯

PORTC.b6 = 1 とする。
PORTCは、7,8桁目ポートと機能が混在しているので注意する事。!!

[13] 7Segment LED 各桁のドライブ：

7Segmentの各桁の指定は、LEDのアノード側を PNPトランジスタでドライブするので Low Active（ 負論理 ）になる。
よって表示させない場合は、全ての桁のポートを Hiにしておく。
そして表示させる桁のポートだけを順次 Lowに落として Activeにしていく。

桁	出力データ
1	PORTA = 0x3E; , PORTC =| 0x03
2	PORTA = 0x3D; , PORTC =| 0x03
3	PORTA = 0x3B; , PORTC =| 0x03
4	PORTA = 0x37; , PORTC =| 0x03
5	PORTA = 0x2F; , PORTC =| 0x03
7	PORTA = 0x1F; , PORTC =| 0x03
6	PORTA = 0x3F; , PORTC =| 0x02
8	PORTA = 0x3F; , PORTC =| 0x01

[14] 作り出して分った事：

Type1ノギスと Type2ノギスの違いは既にページを割いて説明しましたので、その他 Type2ノギスの信号を受けるコンパレータ回路の変更に関して補足しておきます。

コンパレータのシュミット回路を外しました。 これは入力信号が Hiになった時点で DATA信号、CLOCK信号ともに、シュミット回路により更に Hiに引き上げられようとします。 こうなるとノギス側は、[ZERO]ボタン＋「モード切り替えボタン？」を押した状態と誤判定してしまうようです。 よってシュミット回路は外しました。
積分回路の時定数は、当初 10K×1000pでしたが、この時定数だと、13usのノギスCLOCK信号が積分波形となり、振幅が小さくなっていました。 高周波のノイズは取りにくくなりますが 13usのクロックを安定して通すため時定数を小さくしました。

それと、Type2ノギスは、[ZERO]ボタンの機能、及び隠しコマンド？の モード切替えは、通信に使う CLOCKまたは DATA信号を Hi（電池＋側）に PullUpする事により実現するため、ボタンを押した瞬間に壊れた電文を出してしまいます。
受け側の表示器では、Type1と Type2の２つの電文を受けるように作っていたため、Type2の電文が壊れて短くなった場合に Type1の電文と誤判定してしまう現象があり、Inch表示の LEDが点灯してしまうなどの誤動作がありました。 よって受信bit数を厳密にカウントして Type1の場合 24個、Type2の場合 49個（ 割込みが発生する回数は 48ではなく 49になります。 何故、1個多いかは 考えてみて下さい。？ ）以外であれば、壊れた受信電文として無視するようにしました。

その他、Type2特有の処理は、単位変換の演算処理です。 1/100mm でも 1/1000インチでもない単位（ 1インチ 5000H ）で送られて来るので PICマイコン、それもアセンブラでどのように演算処理を実現しようかと悩みました。 浮動小数点演算はやりたくなかったので 整数演算で掛けて割るとかで対処できないか、インチ変換と ミリ変換の係数を眺めてました。 結果としてアセンブラに適した 24bitの足し算＆引き算＆シフト演算で対処できました。 アセンブラのプログラミングについて解説すると長くなるのでここでは書きません。
（いつか気が向いたら、別途で、PICのアセンブラの解説を書くかもしれません。）

[15] 追加した機能、削った機能：

前回作成した基板の一部改造なので、ハード的には殆ど同じです。
追加したのは、モード切り替えのためのタクトスイッチ（マイコン基板上の隙間にマウント）です。
2bitのDIP-SWは、前回 起動時の表示テストをスキップする機能を DIP-SW.1に割り付けていましたが、今回は 表示テストのスキップ機能を削除しました。（単にDIP-SWのビットが足りなかっただけです。） それと、表示テストは、全桁で "0"～"9"まで表示してから、苦肉の "-READY-"表示をしていましたが "-READY-"表示は削除しました。

今回、Type2ノギス対応で、データとして転送されてくる内容に ノギス表示部が現在 mm表示か、inch表示かを識別するビットを送って来ません。ミリでもインチでもない、1インチ 5000Hの数値を送って来るので受け側の表示器で単位変換して表示してます。
その際、インチで表示するのか、ミリで表示するのかの指定を行う必要があったので、それを DIP-SW.1に割り当てました。 DIP-SW1が OFFの時、ミリ表示です。
DIP-SW.1が ONの時、インチ表示となります。 青色LEDが点灯します。
それと、Type2ノギスは 絶対値と相対値の２つのデータを送って来るので、場合によっては絶対値で表示したいという場合もあるのでは... と思い DIP-SW.2を 絶対値、相対値の切り替えに使用しました。
DIP-SW.2が OFFの場合： 相対値表示（ ZEROボタン押下で 座標値が 0.00になる。）
DIP-SW.2が ONの 場合: 絶対値表示（ ZEROボタン押下しても座標値は変わらない。）

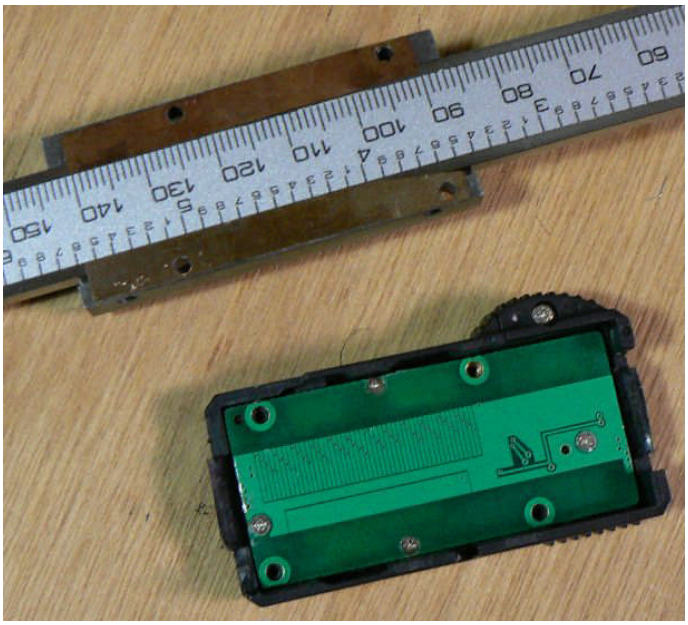
※ Type.2ノギスの絶対値とは、定規がハード的に持っている絶対値という事ではありません。 （ たぶん、電池を装着した時点の位置が 原点になるようです。）

	OFF	ON
DIP-SW.1	Type2ノギス：ミリ表示	Type2ノギス：インチ表示
DIP-SW.2	Type2ノギス：相対値表示	Type2ノギス：絶対値表示

[16]　ノギス表示部分の信号引き出し：

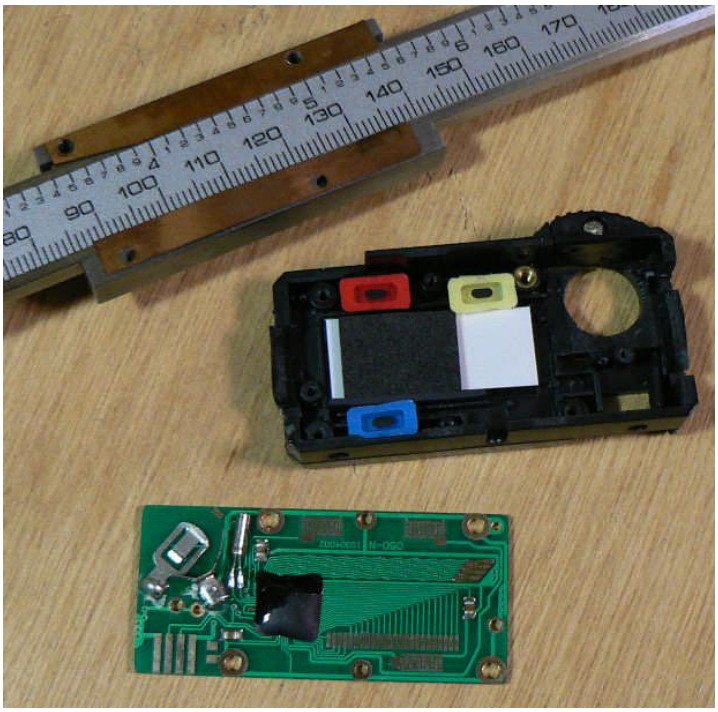
注意：　ノギスを分解して元に戻らなくなっても、私は保証できませんので
あくまで自己責任で行って下さい。

まず、電池を外して下さい。



ノギス裏側の、４つの小さい十字ネジを外します。　これによりステンレスの定規よりプラスチックの表示部分が外れます。
外す時、プラスチック部の左右の定規の当たる部分に小さいゴム製ワイパーが付いているので無くさないようにして下さい。
それと定規 可動部と表示部基板裏の間に薄い銅板製のスペーサが入ってます。
これも扱いに気を付けて下さい。

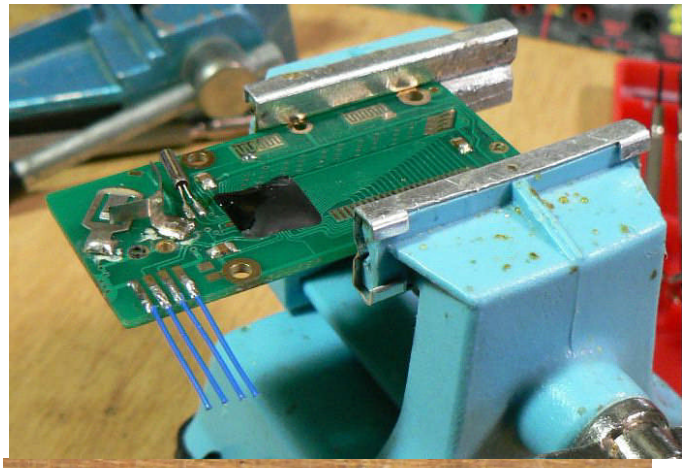
Type1の時もそうでしたが、スライドさせるとちょっとガサツな感じで滑らかさが無いのです。　バラしたついでに、ステンレスの定規と可動部を、金属磨き等で磨いて光らせるとかなり滑らかさが改善されます。



更に裏の基板を止めている４つの皿ビスを外します。　基板を外す時、中のボタンやLCDを押さえているゴムの部品が外れて落ちてくる時があるので気を付けて下さい。

基板を外した状態です。
LCD（液晶表示器）は、何と基板上に乗っているだけです。（線でつながっていません。）すぐ外れます。
プラスチックのカバーにより液晶位置が所定位置に固定されるようになっておりコネクタパターンと基板と接続されるようです。

基板のエッジボード部分に４本リード線をハンダ付けします。
　このリード線の反対側にコネクタを付ける事になりますが、先に基板をプラスチックのカバーに取付けて下さい。　当然、リード線はインターフェースコネクタの穴から外に出しておきます。



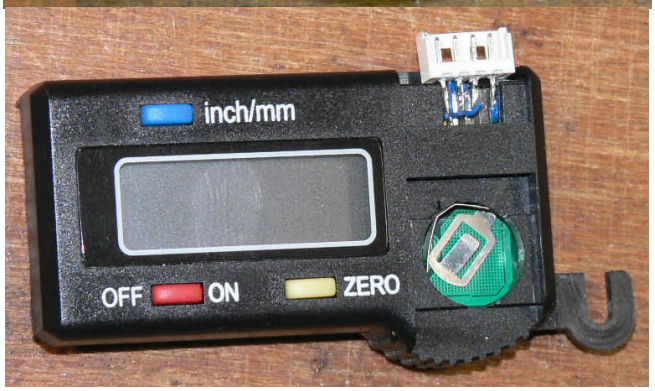
右は、Type1の写真ですが、小さい2㎜ピッチのモレックスのコネクタをハンダ付けします。

コネクタを引っ込めやすいようにリード線をＳ字型に曲げておきます。

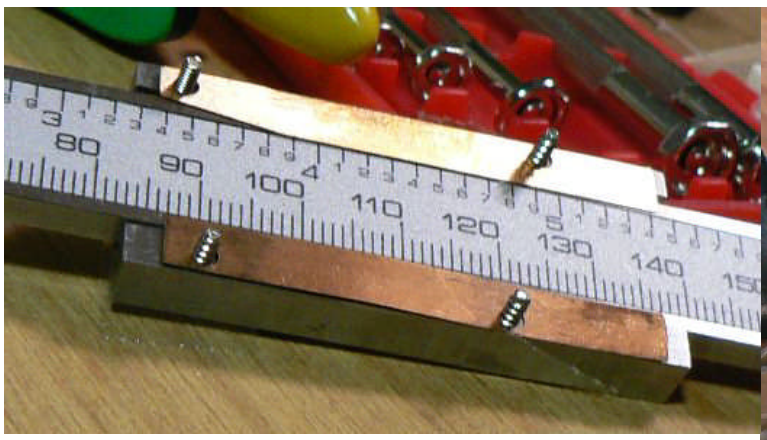
右は、Type1の写真ですが、この程度に引っ込ませたらホットボンドで隙間が出来ないようにコネクタ回りを固めます。

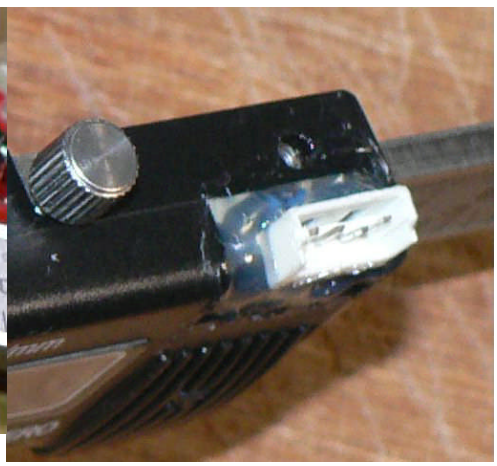
ここまで出来たらステンレスの定規を付けて元通りに組み立てて下さい。

ステンレス定規に取付ける時、薄い銅板の位置に注意して下さい。　ネジ４本を通して落ちないように裏からテープで止めて、ネジに引っかけるような形で銅板の位置合わせをしました。



ハンダ付け時にピンが曲がり、コネクタがすんなり入らない場合があります。　その場合は、先のとがったラジペン等でピンを矯正して下さい。





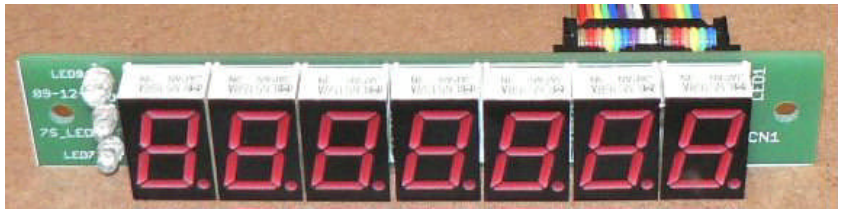
[17]　ケーブルを接続したところ：

　オヤイデ電気より購入した極細４芯シールド線でケーブルを作りました。
ノギス側が 2mmピッチ4Pコネクタです。
PIC基板側は通常の 2.5mmピッチです。
今回長さは、1mほどにしました。

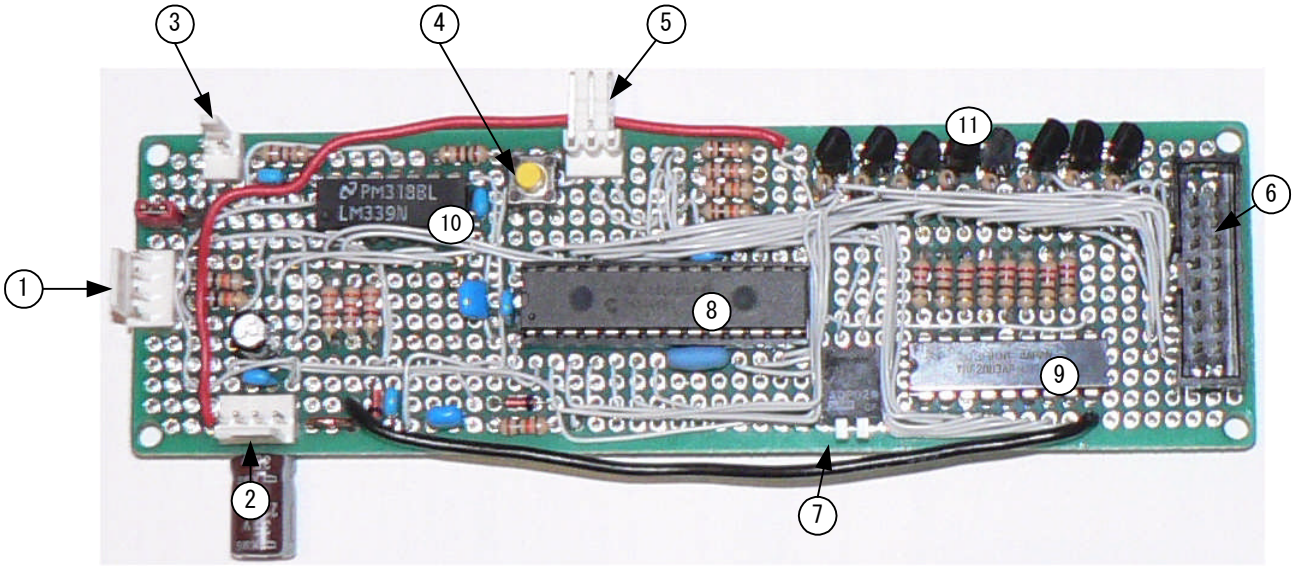
ケーブルが出来たらテスタで導通と隣のピンと接触してないか調べておいて下さい。

[18]　今回作成した基板：

　7Segment LED基板
去年パターンを設計をして別の基板と共に基板業者に製造してもらった物です。



　今回のPICマイコンの基板は、秋月電子のユニバーサルボードにて作成しました。
前回の基板の 一部改造です。



① ノギス信号入力コネクタ
② 電源コネクタ
③ リセット押しボタンスイッチ接続コネクタ
④ モード切り替えタクトスイッチ
⑤ I2Cインタフェース接続用コネクタ
⑥ 7SegmentLED基板接続用コネクタ
⑦ 2P DIP-SW
⑧ PICマイコン PIC18F2320
⑨ トランジスタアレィ TD62083AP
⑩ コンパレータIC LM339
⑪ トランジスタ2SA1015 ×8

[19] ソフト開発環境について：

ハード（基板）は一部改造で簡単に済ませました。
ソフトは、処理速度的な要求から PICマイコンを 16F886から 18F2320に変更した事。
そして、アセンブラで行う事にしたので ゼロからの作り直しとなりました。

とはいえプログラムの仕様、骨組的な部分は把握出来ていたのでその分は楽でした。
開発環境は、Microchip社の **MPLab**環境で **MPASM**（アセンブラ）を使用しました。
フリーでダウンロード出来る標準的な環境なのでその意味では 良かったと思います。

私は、Z80の頃からアセンブラを扱っていた関係で アセンブラで開発する事にさほど抵抗はありませんが、若い技術者の方にアセンブラというと嫌がられてしまいますね。
Ｃで１行で済むところが、アセンブラだと数行になりますし、機械よりの言語なので何をやっているのか可読性が悪いというのはあると思います。

遙か昔は計算機の性能がいまいちで、実装メモリも少なかったため、速度を要求するアプリケーションでは アセンブラは必須でした。　今のパソコン環境は、ＣＰＵの性能が桁外れに高速になり実装メモリも十分過ぎるほどありますから、生産性、メンテ性、移植性の悪いアセンブラを使うメリットが殆どありません。
組み込みマイコンも H8、SHなど高速な物が出てますので、アセンブラは、極く限られた用途に使われるだけです。 それと H8、SHなどは 汎用レジスタを多数持っておりアーキテクチャ的にも、Ｃのような高級言語の Objectを乗せやすいCPUといえます。

ところが、PICマイコンは元々汎用CPU配下で、周辺回路を制御するコントローラとして開発された経緯があり、ちょっと変わったアーキテクチャとなっています。
その関係でＣコンパイラが作りにくい、作ってもObject効率が悪い。等の問題がありPICの世界では、Ｃコンパイラの普及が遅れたのだろうと思います。

運悪く、PICの アセンブラで作る事になった場合、CPUのコストは、16Fシリーズの方が安いのですが、18Fシリーズの方が FSRアクセス時 殆どバンク切り替えを必要としない。RAMメモリが 128byte以内に収まるなら、RAMメモリ空間アクセス時に BANK切り替えを必要としません。　また、命令数も 72に増えていますので 16Fシリーズに比べると 18Fシリーズはアセンブラがだいぶ作りやすくなっています。

あまりシビアなタイミング（応答性）を要求しないのであれば、PICもＣ言語で作った方が楽です。　用途によって使い分けて下さい。

尚、今回のプログラムについての解説はしません。
興味のある方は、ハード仕様は説明はしましたので自分で解読して下さい。