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

目　次

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

[1] 概要：

デジタルノギスのデータの垂れ流しを取り込み、７セグメントＬＥＤに表示を行うPICマイコン基板を作成する事を目的とする。
拡張機能として、ホスト通信も検討する。
用途としては、卓上ミニ工作機械のＤＲＯ用途（フライス盤のＸ，Ｙ，Ｚ軸の座標表示）を想定する。

[2] 回路の検討：

１枚の基板に、1本のデジタルノギスを対応させる。
測定軸が、３軸必要であれば、３枚基板を用意するものとする。
７セグメントの表示基板も、１軸１枚（横に細長い基板）として扱う。

① デジタルノギスの信号取り込み：

デジタルノギスの出力信号は、データとクロックの２つの信号である。
デジタルノギスは、電源 1.5[V]なので通常のマイコンで使用する 5[V]の回路に直接接続する事が出来ないため、レベル変換回路が必要となる。
またデジタルノギスとマイコン基板との間が多少離れている場合も考えられるので、ノイズ対策も考慮しておいた方が望ましい。
考えられるのは、トランジスタで受ける回路、またはコンパレータICで受ける回路が考えられる。　ノイズ対策としては読み取りに支障が出ない程度に、CRによる積分回路を入れる。　及びシュッミットトリガ入力にする事も考えられる。
シュッミットトリガは、正帰還となるのでトランジスタアンプでは２段にする必要がある。　トランジスタ回路では周辺のバイアス抵抗もちょこちょこ必要となるので単純な扱いのコンパレータIC（LM339）を使用してシュッミットトリガを実現してみる事にする。　コンパレータの敷居値は、ダイオードの順電圧 約 0.7[V]を使用する。

② 使用PICマイコン：

デジタル信号入力に、DI 2bit必要。 ７セグメントドライブに a, b, c, d, e, f, g, dotの各セグメントに 8bit、それと桁数分の DOが必要となる。　今回は拡張性を考慮して桁に 8bit用意する。
よって、７セグメントドライブには、DOが 16bit必要となる。
18pinのPICでは、この時点で足が足りないので入手のしやすさと余裕を考え28pinのPIC（ PIC16F886 ）を想定する。

③ ７セグメント表示基板：

表示させる桁数でかわるが、７セグメントドライブにやや信号線が必要となる。
７セグメントはアノードコモンを使用して a, b, c, d, e, f, g, dotの各セグメントは、8bitのトランジスタアレィ（TD62083AP）を使用する。
デジタルノギス用途では、６桁表示で足りると思われるが、拡張性を考慮して8本の DOを用意する事にする。　アノード側のドライブは 2SA1015を必要個数並べる事にする。　表示基板は、Ｘ、Ｙ、Ｚ表示等で縦に３枚並べて使う事が考えられるため、詰めて並べられる様に、７セグメントＬＥＤだけにして他の電子部品は乗せない事にする。

④　デジタルノギスのリモートリセット：

デジタルノギスは、工作機械の可動部の脇に取り付けたりすると、見るのも困難であるが、リセットボタンを押すのも厄介である。
基準位置に移動して座標値をリセットしたいという使い方が当然あり得るのでリモートリセット出来ないか検討する。
デジタルノギスの通信機能でリセットを受け付けるコマンドが用意されているのかは全く分らない。　ノギス側で一方的に垂れ流し的にマスタクロックを出しているので通信仕様的に相手側からのコマンドを受ける事は難しいように思われる。
じゃ出来ないのか。？
　一つ実現する方法としてノギスの表示部をバラしてリセットボタンの配線を引き出して使用する事が考えられる。　ノイズに弱くなる事は考えられるが他に方法は無い。
ノギスと マイコンとの間は細いシールド線を用いる方が望ましいと思われる。

リモートリセットについては上記の方法で成功した。
ノイズ対策としてシールド線の反対側に 0.1uFのコンデンサを付けてみたが特に支障はなかった。　よってこの方法で、ＯＫと判断する。

[3]　信号の引き出し：

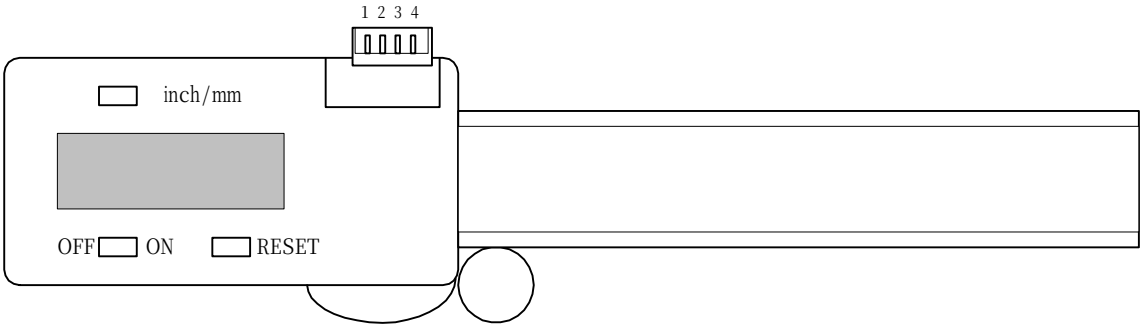
　極細の４芯シールド線を用いて信号を引き出す。　長さ１m程度とする。
デジタルノギス側のコネクタは、元々付いているエッジボードコネクタに合うコネクタがないので、市販品のコネクタを取り付ける事で対応する。
　今回、デジタルノギスのコネクタ部から、あまり大きく飛び出さないように 2㎜ピッチのコネクタを付けた。　左から

1．GND
2．DATA
3．CLOCK
4．RESET

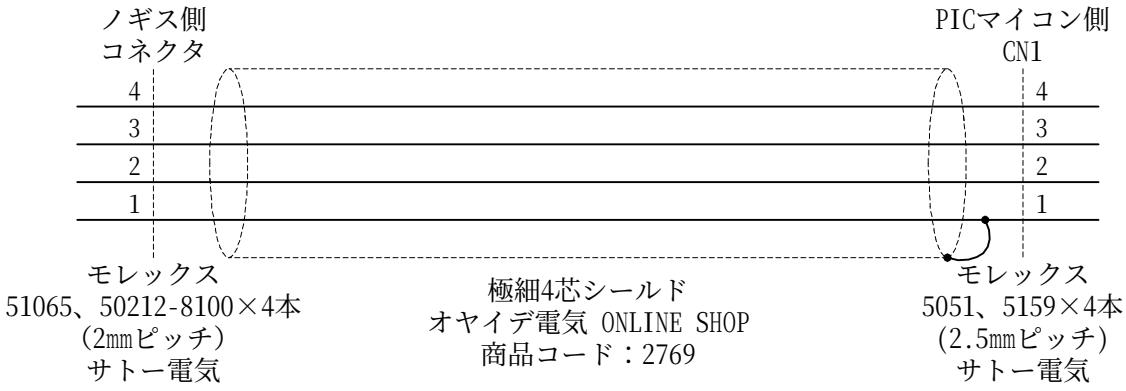
のピンアサインとした。
　ノギス本体とコネクタ間の配線は、0.28㎜の単線にて行った。
１、2、3は、エッジボードから引き出し、４は、RESETボタンのパターン部にハンダ着けして引き出した。　コネクタの固定はホットボンドにて行った。

　2㎜のコネクタはやや小さいので、ピンへのハンダ付けがやりにくい。
長く熱しているとピンを止めている樹脂が柔らかくなってピンが斜めになる事もある。
（その場合ラジペンで逆方向に曲げて直す。）
ターミナル側も専用の圧着ペンチが無いのでハンダ付けしてラジペンで丸く曲げた。
小さいのでハンダ付けがやりにくい。

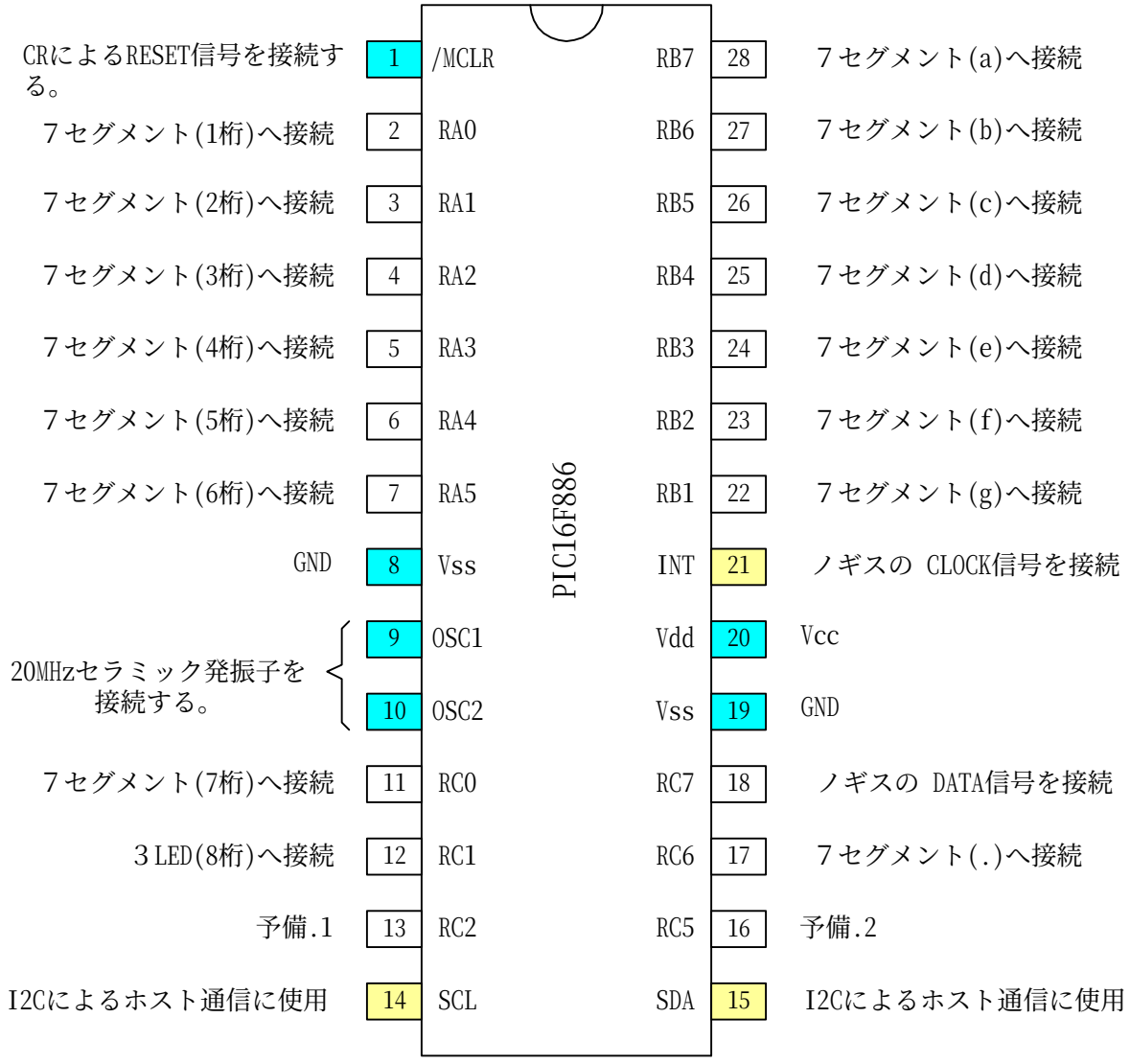
使用コネクタ：モレックス 53253（ケーブル側：51065、ターミナル：50212-8100）
入手先：サトー電気



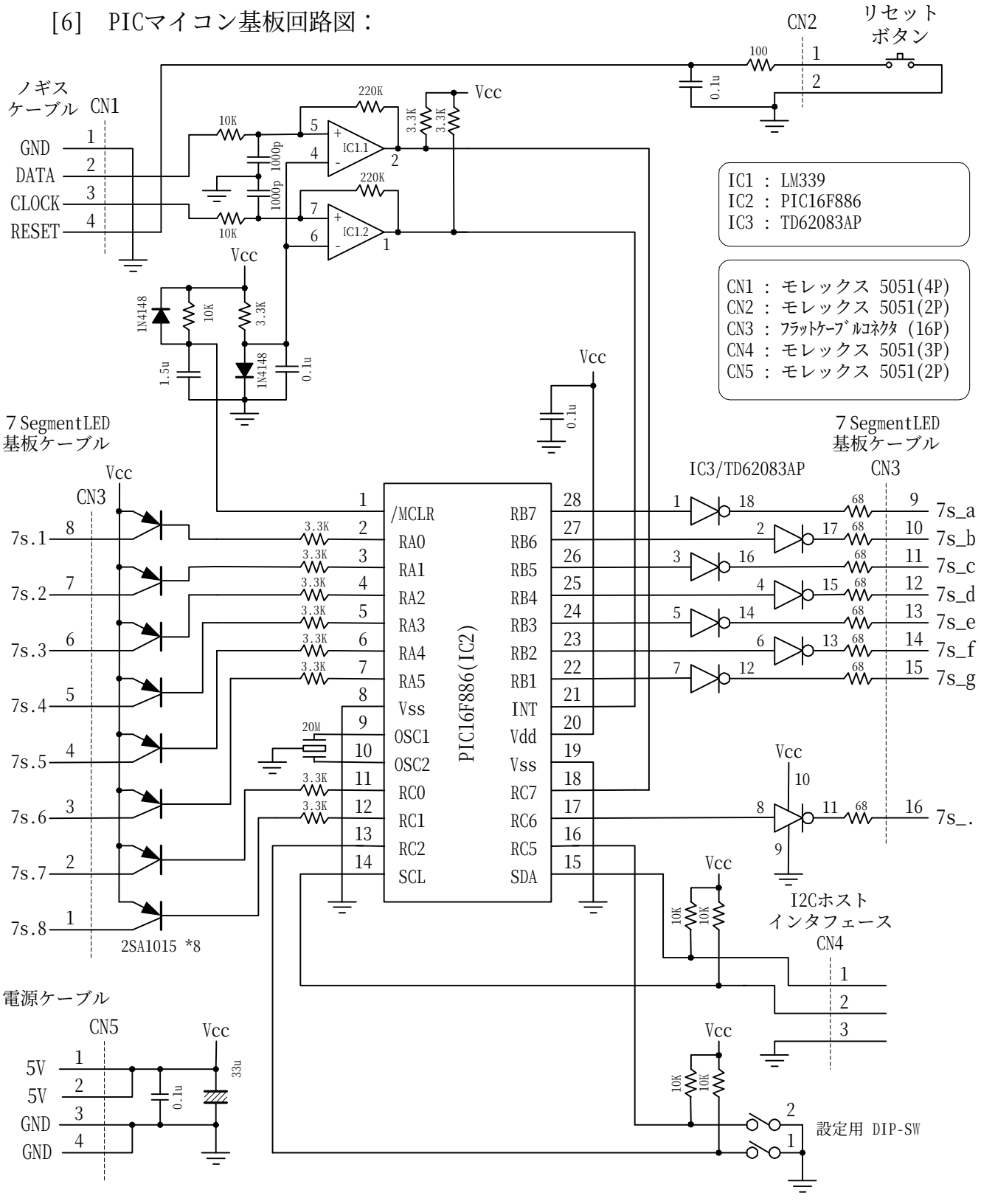
[4]　ノギス、PICマイコン間ケーブル：



[5]　使用PICマイコン（ PIC16F886 ）ピンアサイン：



[6] PICマイコン基板回路図：

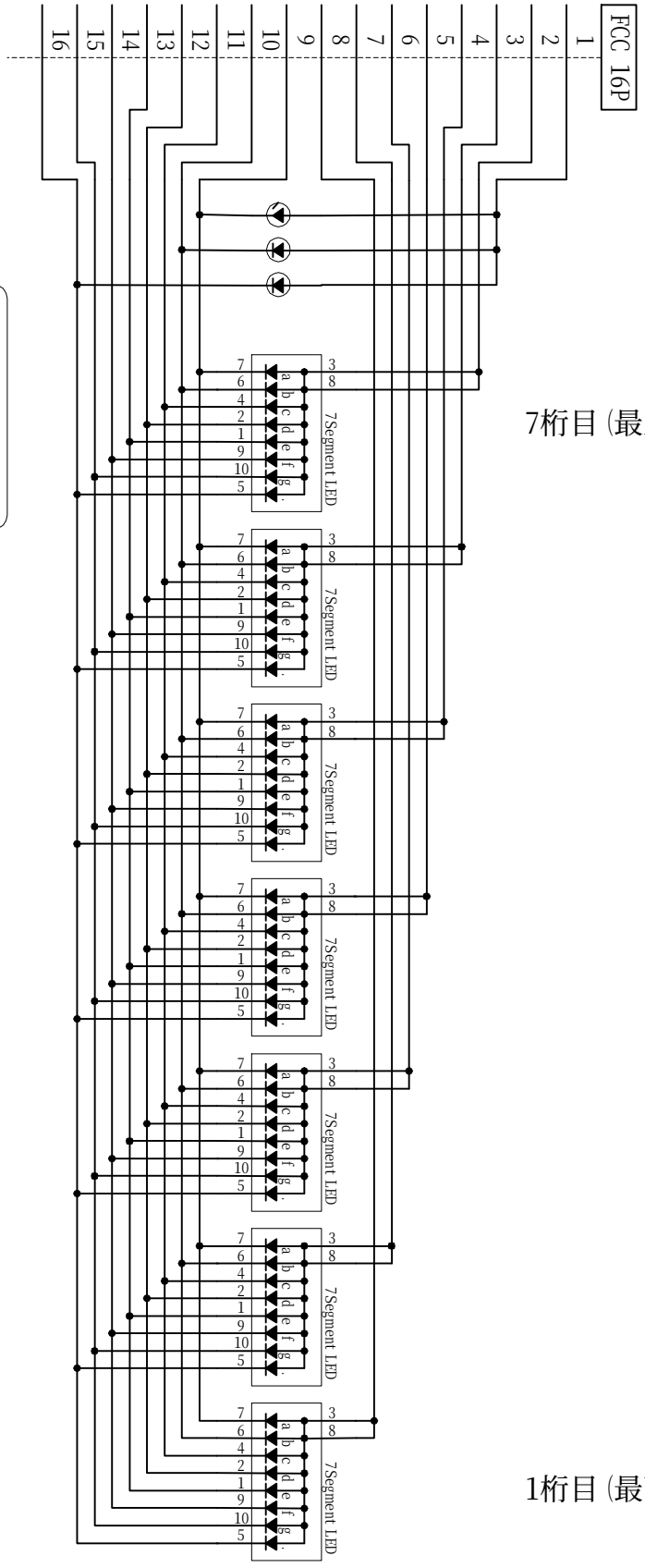


注意点

① IC2（PIC16F886）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 2.5㎜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。

[7]　7SegmentLED表示基板回路図：

正面から見て左端に 3㎜LED３個
が来るように配置してます。
3㎜LED３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。



7桁目（最上位桁）

1桁目（最下位桁）

使用 7Segment LED
LN516RA（赤）

[8] 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	1
2	カーボン抵抗 1/4W		68	8
3	カーボン抵抗 1/4W		2.2K	2
4	カーボン抵抗 1/4W		3.3K	11
5	カーボン抵抗 1/4W		10K	7
6	カーボン抵抗 1/4W		220K	2
7	IC1	NS	LM339	1
8	IC2	MicroChip	PIC16F886	1
9	IC3	東芝	TD62083AP	1
10	PNPトランジスタ	東芝	2SA1015	8
11	ダイオード		1N4148	2
12	セラミック発振子		20MHz	1
13	セラコン		1000p	2
14	積層セラコン		0.1uF	5
15	積層セラコン		1.5uF	1
16	電解コン		33uF	1
17	DIP-SW		2P	1
18	ICソケット		28P	1
19	コネクタ CN1,CN5	モレックス	5045/4P	2
20	コネクタ CN2	モレックス	5045/2P	1
21	コネクタ CN3	ﾌﾗｯﾄｹｰﾌﾞﾙ用	16P	1
22	ﾌﾗｯﾄｹｰﾌﾞﾙ ｹｰﾌﾞﾙ側ｺﾈｸﾀ		16P	1
23	コネクタ CN4	モレックス	5045/3P	1
24	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN1,CN5	モレックス	5051/4P	2
25	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN2	モレックス	5051/2P	1
26	ｺﾈｸﾀ ｹｰﾌﾞﾙ側CN4	モレックス	5051/3P	1
27	ｺﾈｸﾀ ピン	モレックス	5159	13
28	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

[9] 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)

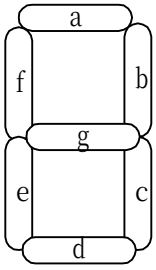
ノギスClockは、PORTB.0を INT信号（立ち上がりエッジ）として受ける。
INT信号の割り込み処理にて、ノギスDataを読み取る。

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。

[10] 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を点灯させれば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。!!

[11] 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

[12] 作り出して分った事：

今回のデジタルノギスは、mmと inchの２つの表示モードが有るが、mmとinchの区別はどうなっているのか疑問が出てきた。 出力データ上で見分けが付かないと mmのデータを送っているのか、inchのデータを送っているのか分らない。
今回、受信したデータ 4bit 6個のデータをそのまま7SegmentLEDに表示して分った事は最大値bitというか、最終bit（先頭から24番目）のbitが 1であれば inchであることが判明した。
それと、ノギスで inchモードで表示していると 最下位に 小さな 5 が表示されることがある。 これは、データ上ではどのように扱われているのか。？
これについても分った。 最小分解能が、mmモードの場合 1/100mmで 12.5mmであれば 1250 というデータ値になる。
inchモードの場合、 1.200inchの場合、倍の 2400という値で送信される。
1.200 5 inchと ノギスの液晶に表示されている場合は、 2401という値で送信される。
値が２倍されて表示されると扱いにくいので、今回は、inchモードの場合、受信したデータを 5倍して表示することにした。

[13] 今回作成したデジタルノギスDRO基板のソフト機能：

① 電源ＯＮN時、7Segment LEDの表示テスト（全桁 0～9の表示）を行う。
その後 苦肉の -READY- 表示を行う。 (^_^;
② データ受信中は、黄色LEDを点灯させる。
③ インチモードの場合は、青色LEDを点灯させる。
④ ミリモード、インチモードにて、少数点表示位置を適切に表示する。
⑤ 上位桁のゼロサプレスを行う。
⑥ 負の値を表示する時は、マイナス（-）表示を絶対値に応じて表示位置をシフトさせる。
⑦ 初期表示テストが不要な場合は、DIP-SW.1を ON（Lowに落とす）事により
表示テストを行わないようにする。
⑧ ソフトではないが、リモートリセット機能有り。

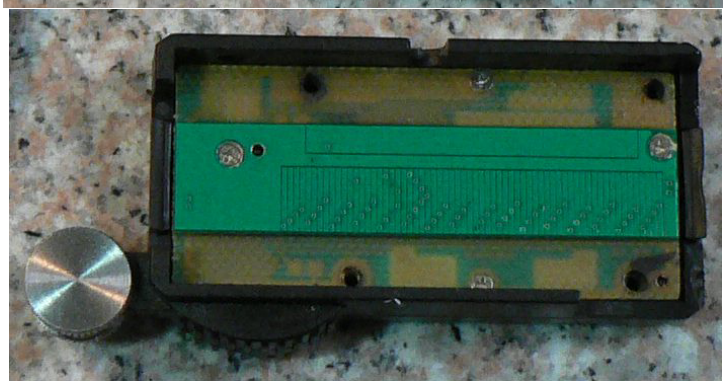
[14]　ノギス表示部分の信号引き出し：

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。

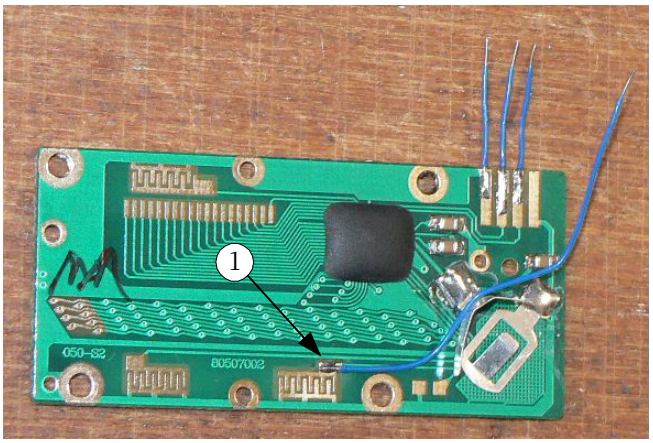
ノギス裏側のシールを剥がし、４つのネジを外します。　これによりステンレスの定規よりプラスチックの表示部分が外れます。

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。

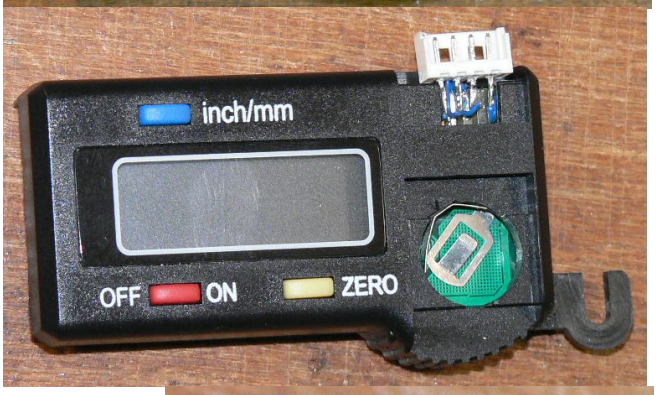
LCD（液晶表示器）を外し
　リセットボタン部分に細い
単線のリード線①を ハンダ付けします。
　カバーをかぶせる時に、カバー内側の
ネジ支柱で、リセットのリード線を踏まない
ように注意してカバー内に固定して下さい。
　押しボタンのゴム、液晶表示器、それを固定
するゴムは、最初にカバー内に入れておいて
上から基板を被せるような感じで入れます。
　当然 4本のリード線はカバーのコネクタ穴か
ら外に出るように挿入して下さい。



カバーを被せてからコネクタをハンダ付けしま
す。 ちょっとハンダ付けが
やりにくいですけど...
　コネクタの樹脂が柔らかくなりピンが曲がっ
てくるのでなるべく早くハンダ付けした方が
いいです。

　ハンダ付けしたら、リード線を折りたためる
ようにＳ字に曲げます。

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。

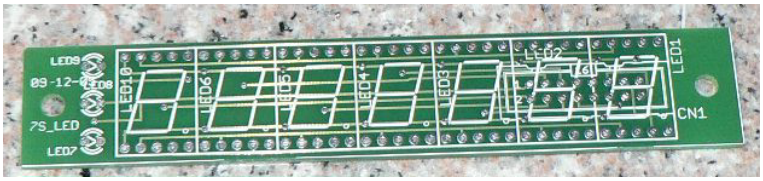
[15]　ケーブルを接続したところ：

　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 2㎜ピッチ4Pコネクタです。
PIC基板側は通常の 2.5㎜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。

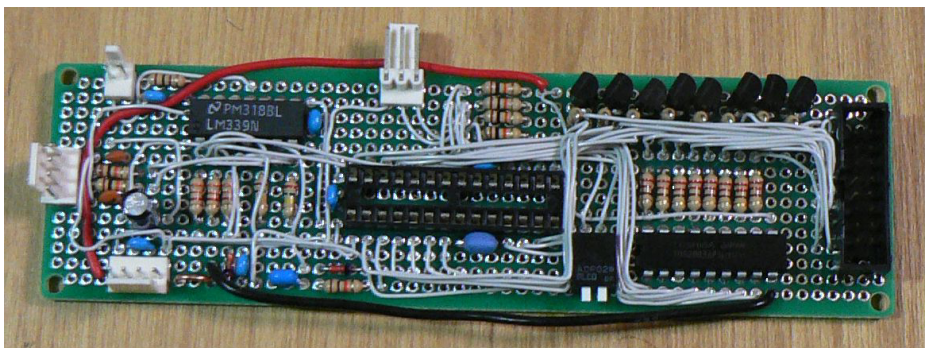
[16]　今回作成した基板：

　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。
一枚だけならいいけど何枚も作ることになるとうんざりですよね。

　ちゃんと動作することが確認できたので、お金に余裕が有るとき!! という条件付きでこの基板もパターン起そうかな。？ と思っております。（ いつの事やら... ）

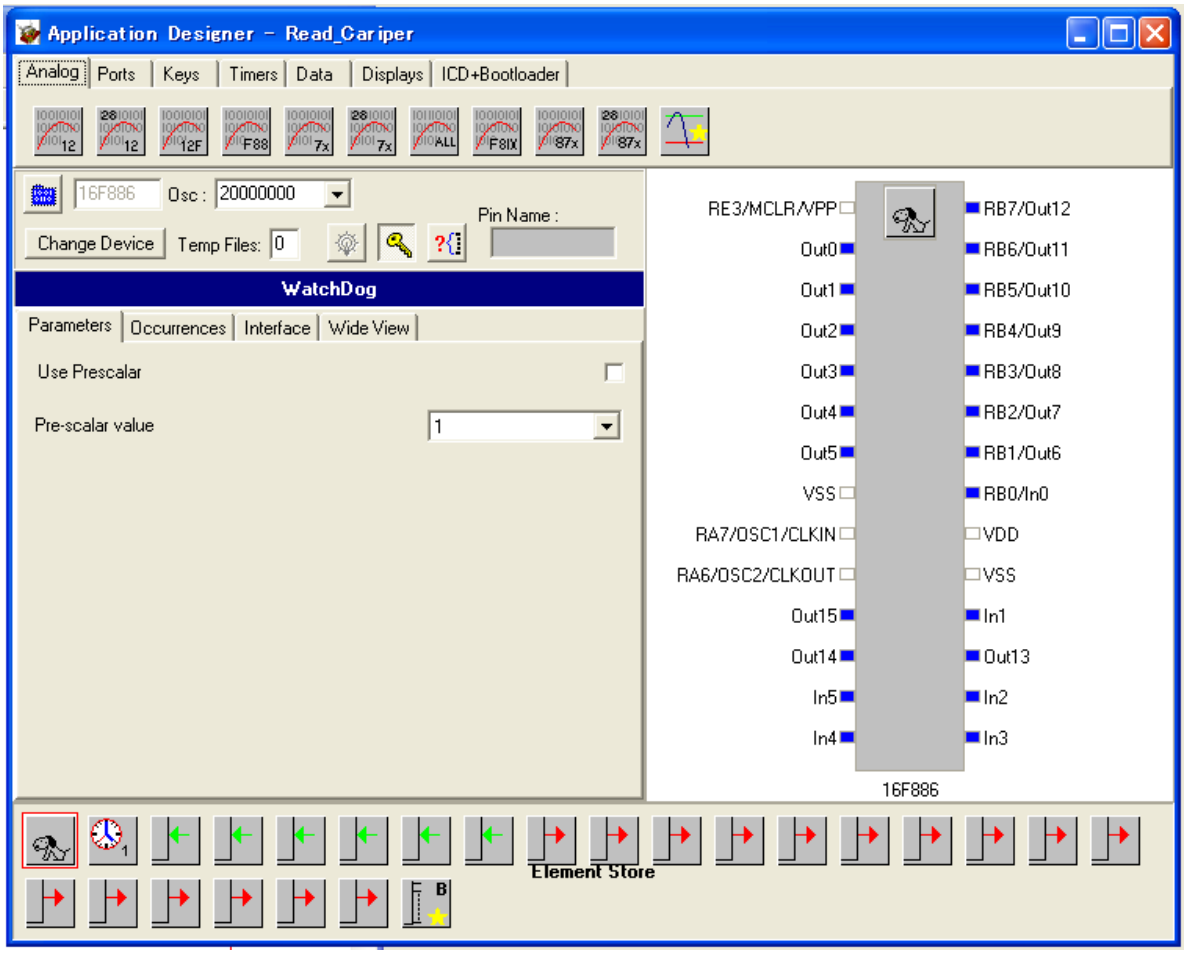


[17]　ソフト開発環境について：

　私は、半年ぐらい前までは、PICのプログラムは、ずっとMPLABの MPASM（アセンブラ）で開発していました。
　アセンブラで作るのは、コンパクトで高速なオブジェクトを作るのは可能ですが、使用するPICのデータシートとにらめっこしながら煩雑なコードを記述しなければならないため ちょっと面倒でした。
　Ｃの開発環境に移行したいと思っておりましたが、使い方がよく分らないせいか、不可解な現象で悩まされ、しばらくはＣの環境は放置しておりました。　巷では、PICのＣコンパイラは、バグがある。　クセがある。　との噂があり 私が使おうとしていた WIZ-Cも潜在的なバグがあるのでは... と思ってました。 半年前ぐらいに WIZ-Cのバージョンアップ版を購入しました。（ Version 16.00です。）
少しは良くなっているのでは... と思い使い始めました。

試行錯誤しながら、仕事で小さいプログラムを２本ほど作成しました。
今回が、３本目です。　で、慣れてくると、アセンブラに比べ すごく楽です。
特に、WIZ-Cの場合、ちょどWindows開発環境の VBのようにエレメントと呼ばれるアイコンの機能部品（PICの周辺機能をラップしている。）をフォームにぺたぺた貼り付ける感覚で扱え、プロパティシートの設定で細かいパラメータ（例えばシリアル通信のボーレイト等）が設定出来るので、ほとんどの周辺回路の初期化処理をコードで書いたことがありません。
　これはすごく楽です。　更に応答速度にシビアにこだわらなければ、割込み機能を持ったエレメントを貼り付けた際に オカーレンス関数（割込みが発生したときに、呼び出されるイベント処理関数）の登録が出来ます。 これも便利で、タイマー割込み、外部割込み、通信割込みのイベント処理を VBのイベント処理的な感覚で扱える物です。
　ただ組み込み用開発環境として見るとコード記述を中心とした他の処理系と比べ、ビジュアルに設定出来る事が戸惑う要素なのかもしれません。
今となっては、この便利さになじんでしまって、もうアセンブラには戻る気になりません。

下の画像は、WIZ-Cのアプリケーションデザイナの画面です。
犬のアイコンはドッグタイマのリフレッシュ機能、時計はタイマー機能、→は出力、←は入力といった具合で、どのピンを入力、あるいは出力にするかをビジュアルに設定します。



気を付けなければならない事も有ります。
Ｃ言語の場合、ハードの細かい事をあまり意識せずにコーディング出来ますが、唯一 PICのハード仕様を意識しないといけない部分としてコンフィグレーションビットの設定が有ります。
これがちょっと厄介です。　PICの品種毎にコンフィグレーションビットが微妙に異なるため、使用するPICのデータシートのコンフィグレーションビットの説明を見て WIZ-Cの設定と見比べて下さい。
初期に経験した不可解な現象の原因は、殆どがこれでした。
それと何もしないと I/Oピンがアナログ入力に初期化される場合があるので注意が必要です。　PIC16F886の場合は、全てデジタルI/Oで使用する時には初期化処理として ANSEL、ANSELH 共に 0 を代入する必要があります。

それとコンフィグレーションの設定と、アプリケーションデザイナに貼り付けるエレメントの連携も意識する必要があります。
例えば、コンフィグレーションビットで WDTを有効にしたら、Watch Dogのエレメントを貼り付ける必要があります。　これをやらないと途中でリセットが毎回かかるので、ちょっと進んで頭に戻るような状態でループします。

それと WIZ-Cは、イギリスで開発された開発環境なので日本語の対応がいまいちなのです。
デフォルトでは、開発環境内のエディタで漢字を入力する事が出来ません。
半角英数記号のみです。　インストール時に漢字をサポートする設定が出てくるのでそれを有効にすると確かに漢字は入力出来るのですが、そのソースファイルは、半角英数も 2byte表記の ユニコード文字なので コンパイラが直接受け付けません。
よって 漢字を ?? に変換したASCII文字のソースファイルを別途 自動生成してコンパイラが読み込みます。　何か ややこしい話です。
またユニコードのファイルは ソースファイルとして他の処理系に持って行く場合も扱いにくいので何とかならないかと思っていました。

その後、一ついい方法を思いつきました。
自分が使いなれたテキストエディタ（ MIFESでも秀丸でもかまいません。）で ソースを作成するのです。 WIZ-Cのコンパイラは、 //のコメント部分に 漢字コードが含まれていても無視するだけなので問題ありません。
WIZ-C内で 別のエディタを呼び出すのは、WIZ-Cの Toolsに登録は出来ます。
別のエディタの呼び出しは出来ますが自動的にプロジェクトのソースを読み込むまでは出来ません。　しかし、エディタの履歴に最近呼び出したファイル名が残っているので不自由は有りません。　　これで自分好み？ で 使い勝手が良くなった気がします。
この場合の注意：WIZ-Cのエディタ側で編集中のファイルを表示しないこと。!!
ユーザが作成するファイル名は ?????_user.c と最後に user.cが付く仕様になっています。
このファイルのみ外部のエディタで編集するということです。
そしてコンパイル前に必ず保存する事。

WIZ-Cに関しては、また改めて説明のページを作ろうかと思います。