平成22年度

卒 業 研 究 報 告

研究題目

ステッピングモータを使った光軸調整システムの製作

指導教員

由井 四海

著者

皆越 慎吾

平成22年 3月 9日提出

独立行政法人国立高等専門学校機構

富山高等専門学校 電子制御工学科

目次

1.　序論

1．1　背景

私たちが生活している現代社会では技術力、産業が発展し生活が便利になってきている。しかし、その反面で化石燃料やエネルギーを大量に消費することにより地球温暖化や、大気汚染が進み地球規模の問題となっている。その中の大気汚染で影響を与えているものの一つとしてガスや浮遊粒子状物質などがある。これらのものを短期的、中間的、長期的にデータを観測しそれを比較することにより**地球温暖化メカニズムの解明や**総合的解析やプロジェクトの研究などに使われている。広い範囲でそして時間帯ごとに測定を行うことにより測定区間の大気に含まれる物質の時間変化による値を調べることができる。それらを分析しその結果から将来の気候を予想する研究を支えるものとなっている。これらの活動から地球環境保全のための有効な対策がとられている。

１．２　光学的観測

大気観測をする方法の一つとして光学的方法が挙げられる。この方法では望遠鏡を用いるものがあり、大気を通過してきた光を集めそれを測定しデータを習得している。その光をフォトダイオードなどに受光しそのデータから光の吸収スペクトルや光の減衰などを測定し大気中に存在している物質を調べるという方法である。光学的観測ではオゾンや雲の成分検出も行えるため広い範囲の測定ができる。ニュートン式望遠鏡を用いた測定方法の概要を図１－１に示す。

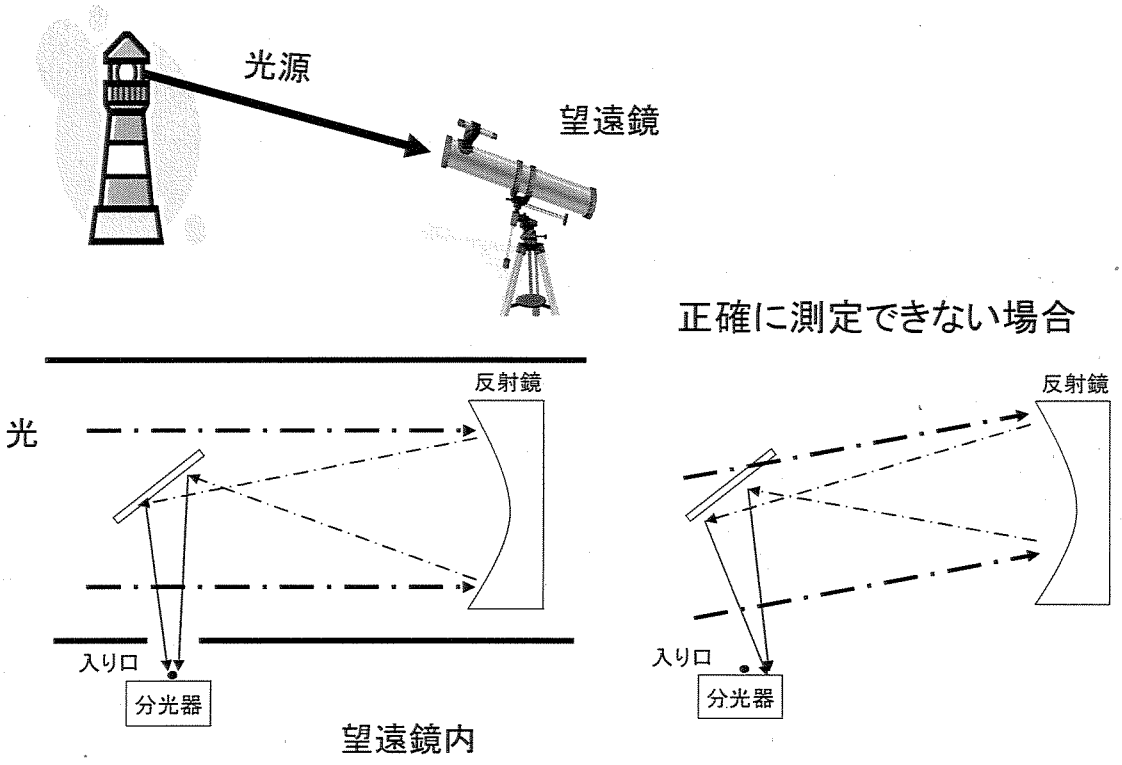

図１－１　ニュートン式望遠鏡を使った光学的観測方法

１．３　目的

光源からの光を望遠鏡で集めていた時、建物が温度の変化により歪んだり何らかの事態により観測位置から微小にずれてしまった場合、再び実験データを得られるように修正を行わなければならない。修正手段として凹面鏡（反射鏡とする）を用いて望遠鏡内に入った光の位置修正を行い観測点に集めている。観測位置からずれた場合、反射鏡を調節する際に人が毎回調節することもできるが労力がそれなりに必要になる。また、微調節が必要になり手動での調整は困難になる。そこで本研究では、修正したい位置を入力するとその場所に反射光がいく位置調節を行うための調節システムの製作を目的とした研究を行った。

１．４　PIC18F2550 使用 USB マイコンボード

本研究では、ワンチップマイコンの中で A/D 変換が行え USB ポートを搭載している秋月電子通信社製の PIC18Ｆ2550 使用 USB マイコンボード（MICROCHIP18Ｆ2550）を使用した。図１－２にマイコンボードの回路を示し特徴を以下に示す。

- 基盤サイズ 40mm×18mm で、28 ピン IC ソケットに装着可能
- プログラムメモリタイプはフラッシュメモリ 32 キロバイト
- 20MHz クリスタルを搭載して、内部 PLL で 48MHz で高速動作
- 電源は、USB からのバスパワー、外部からの５V 電源、基盤搭載の５V レギュレータ使用の３タイプの方式に対応
- ソフトの開発は、マイクロチップ社無償版 C18 コンパイラ(StudentEdition)が使用可能
- ソフトの書き込みは、AKI-PIC プログラム Ver4 に対応
- 直接的、間接的あるいは相対的なアドレッシングモードを持つ。
- A/D 変換の分解能は 10 チャンネル、10bit である
- 動作電圧範囲は 2~5.5(V)である
- ２サイクルである分岐命令を除くすべての命令は１サイクルで実行される

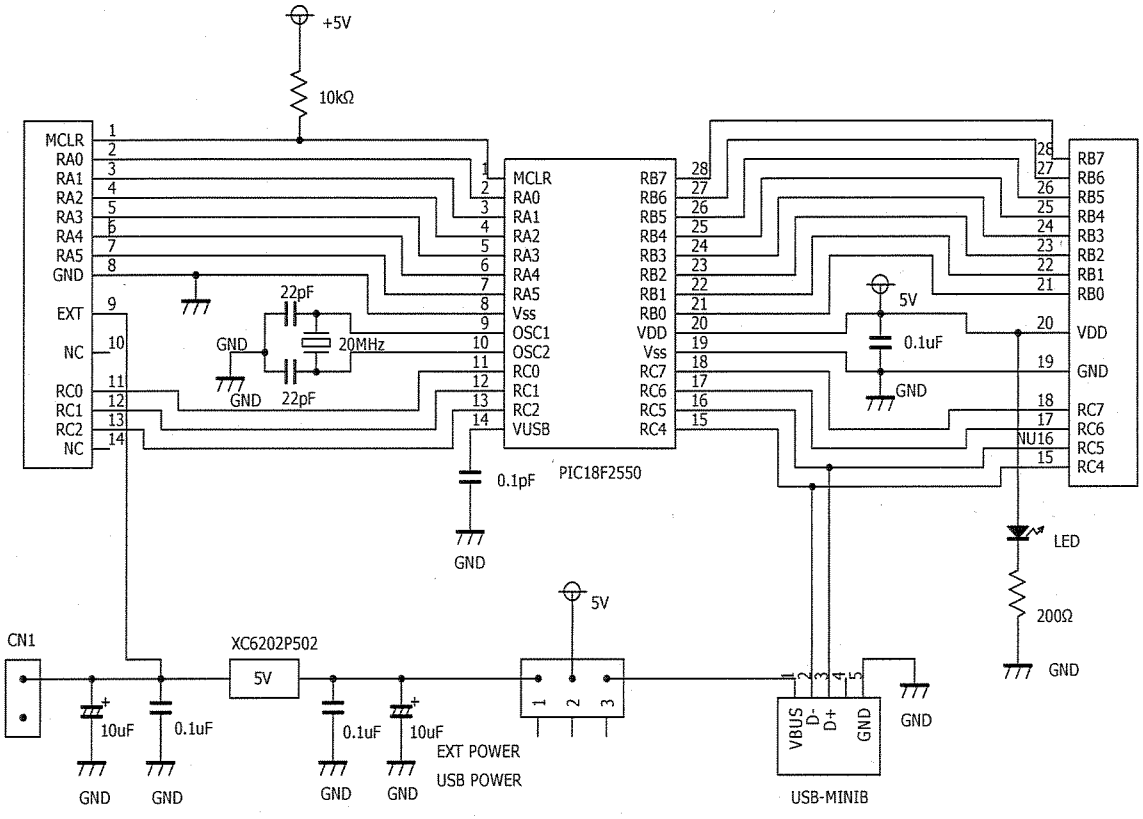

図１－２　MICROCHIP18F2550

１．５　PICkit2

PIC18f2550 にプログラムを書き込むために、MICROCHIP 社製の PICkit2 を使用した。図１－３にプログラムを書き込むために製作した回路を示す。以下に特徴を示す。

- インサーキット書き込みでユーザー基盤に直接書き込み可能
- USB インターフェースによる書き込み品装備
- 専用プログラマ(MPLAB とは別)
- 独立な書き込み器としても使用可能
- 電源は USB 供給可能
- PIC16・PIC18・PIC24・dsPIC30F・dsPIC33F など多くのデバイスをサポート
- モジュラ変換コネクタで MPLAB IDE 2 コネクタにダイレクトレイン
- MAL のほとんどのボードに対応

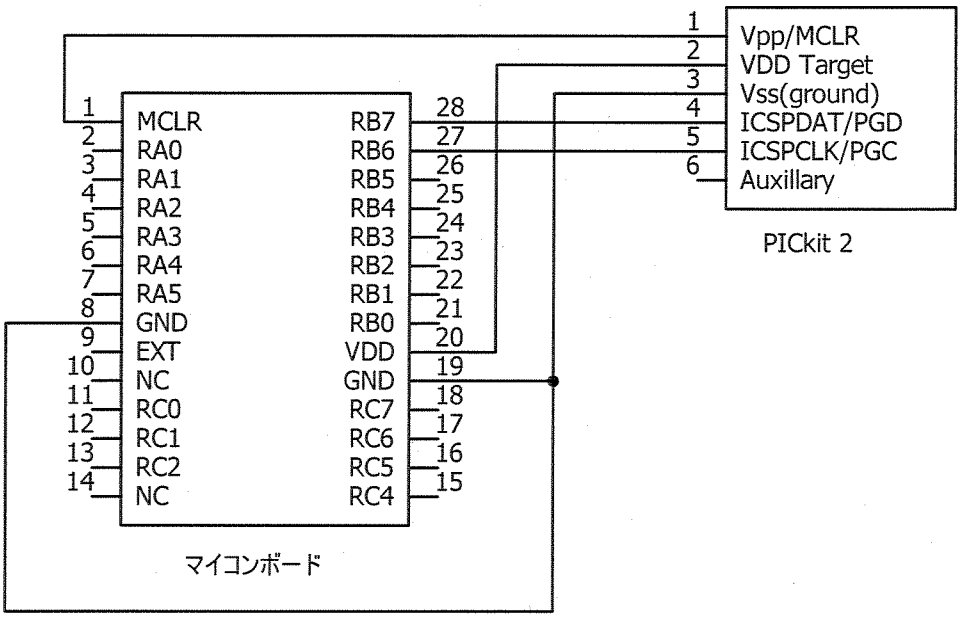

図１－３　PICkit2 とマイコンボードの接続

２．　反射鏡傾き調節機器の製作

２．１　反射鏡の調節機構

製作した概要図を図２－１に示す。今回はステッピングモータが回転すると、カップリングにより連結されたネジも回転するとようにした。ネジは反射鏡を固定している板に巻かれていくので, その動作によりネジの長さ L1、L2、L3 の長さが変化するようにした。またバネを用いることによりネジを固定している板と反射鏡固定板との距離関係が保たれている。ステッピングモータを３箇所とも板で固定することによりカップリングにかかる負荷を少なくした。

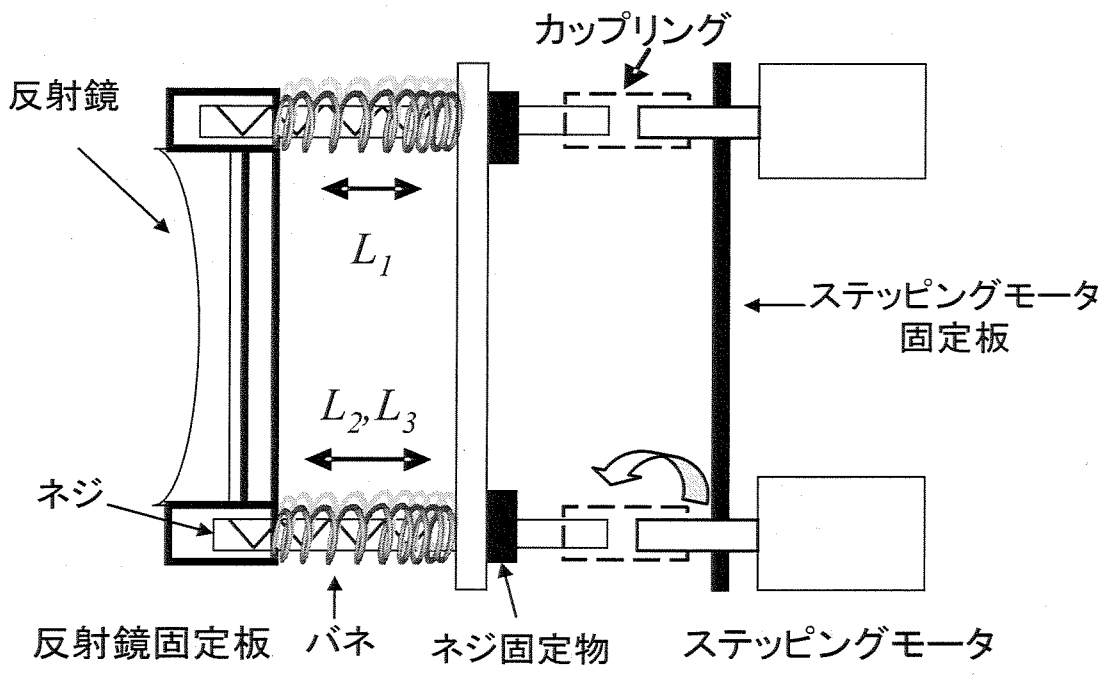

図２－１　反射鏡調節機器の概要

以下にこの機器で用いた製品を表２－１に示す。

表２－１　反射鏡調節機器で使用した部品

カップリング MiSUMi 製	
型式	MCJN15
許容トルク	0.12N・m
許容偏角	2.5°
許容偏心	0.2mm
静的ねじり	12N・m/rad
慣性モーメント	1×10^{-7}
質量	4g

ワッシャ MiSUMi 製	
型式	SSWA

バネ MiSUMi 製	
型式	NW-5.5
自由長	30mm
バネ定数	10%
許容たわみ量	12%
密着長	10.9mm
Nmax	13.7kgf
N/mm	1.1

２．２　ステッピングモータ固定機器の製作

先ほど説明した三点にあるステッピングモータを固定する機器の製作を行った。製作するにあたり、正三角形の位置関係になるように固定した。また固定するときステッピングモータの軸が少しでも正三角形の位置よりずれてしまうとネジを固定している箇所からずれてしまい、カップリングが曲がった状態になり余計な負荷がかかってしまう。製作した機器は微調整可能なように図２－２のようにし移動可能なものを製作した。

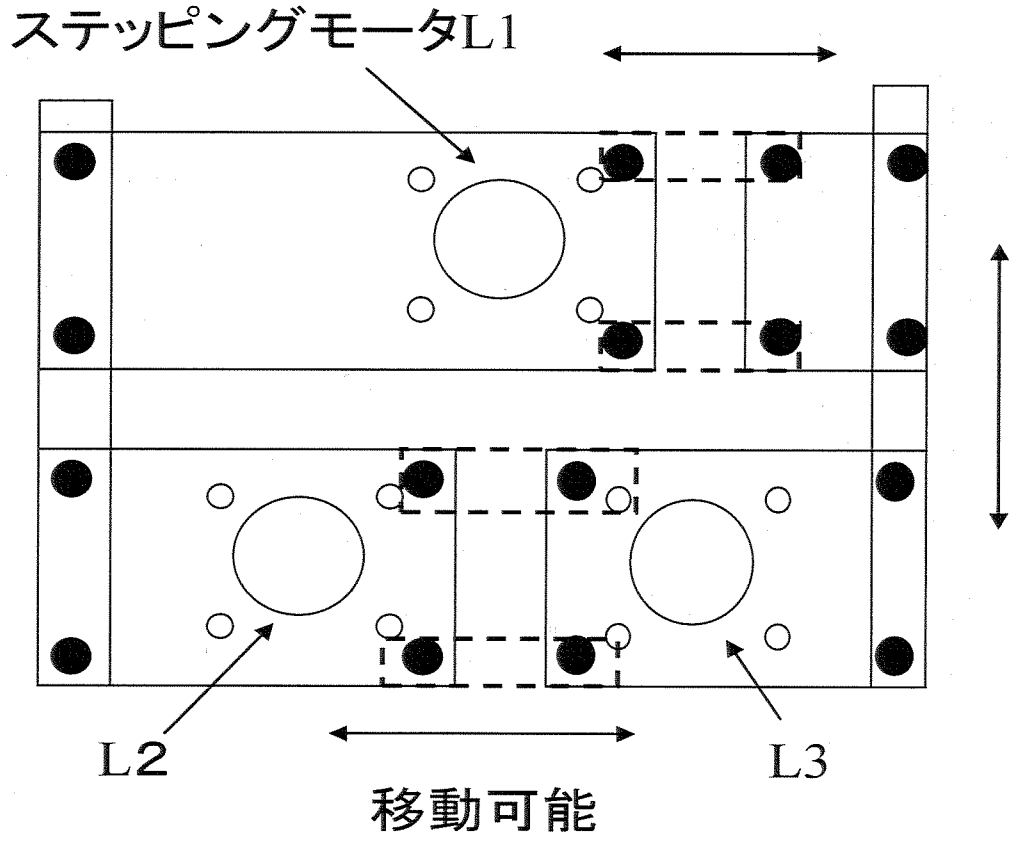

図２－２　ステッピングモータ固定機器

２．３　ステッピングモータ固定板とネジ固定板の設置

製作したステッピングモータ固定板とネジ固定板とを接続する際に、この二つを支えているものはネジとモータ軸をつなぐカップリングのみである。このままではステッピングモータの重さにより下方向に負荷が大きくかかる。カップリングに大きな負荷がかかってしまい、モータの回転する力が足りなくなり動かなくなる可能性がある。そのため図２－３のようなステッピングモータ固定板とネジ固定板を支える固定器を製作し取り付けた。このことにより余計な負荷がかからずネジが回りやすくなった。

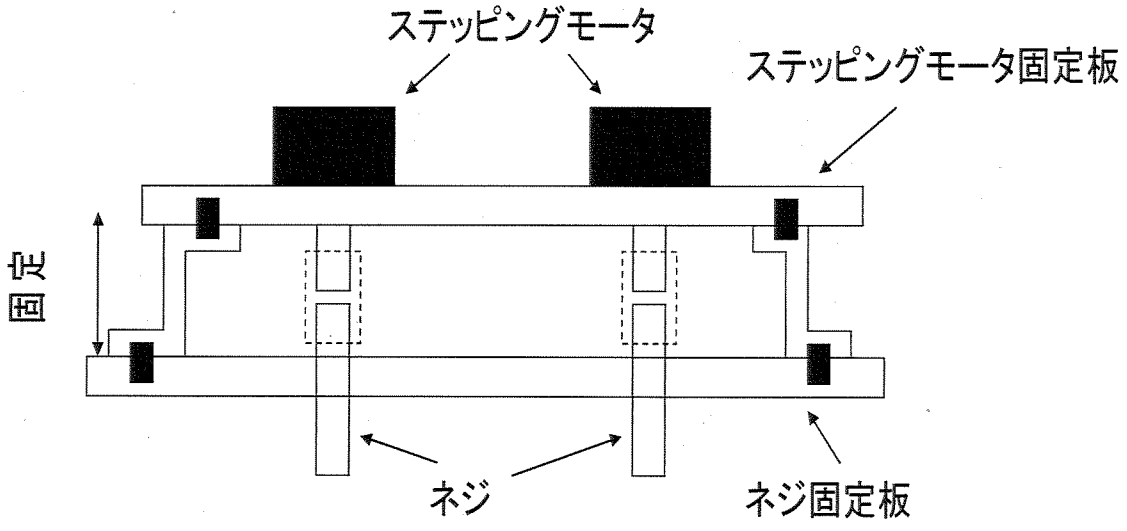

図２－３　ステッピングモータ固定版とネジ固定版の固定

２．４　反射鏡調節機器の製作

これまでに製作した機器を組み合わせて、反射鏡調節機器とそれを取り付ける土台の作成を行った。今回の調節機器はステッピングモータが動作中にずれたりしてはいけないので動かないように固定した。図２－４が製作した固定機器である。これを用いて反射光の調節を行った。

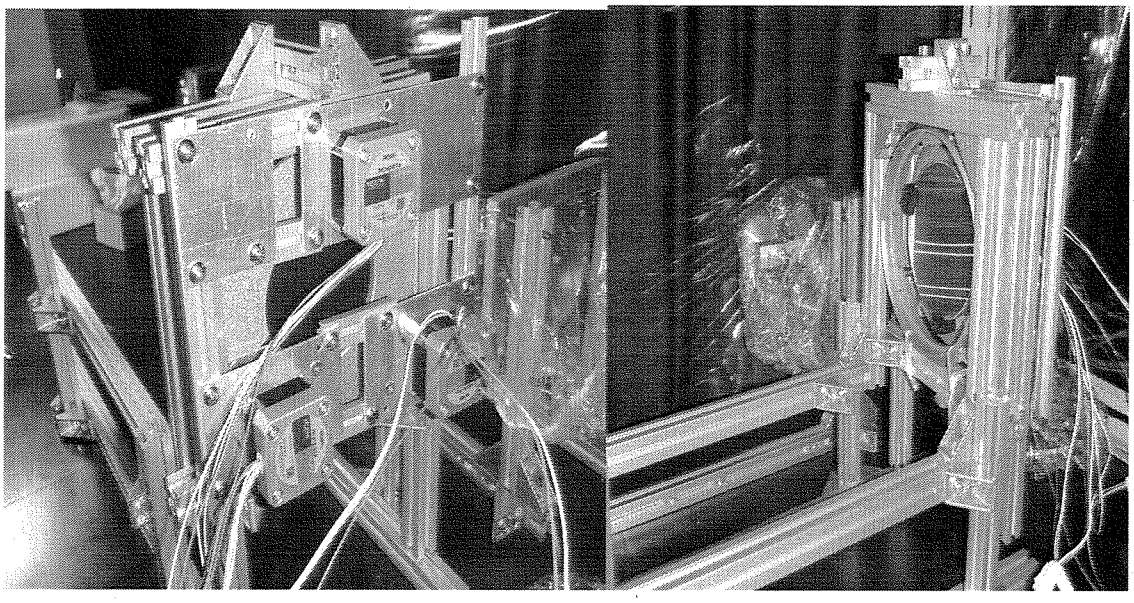

図２－４　実際に製作した反射鏡調節装置

３．　ステッピングモータ

３．１　ステッピングモータの特性

ステッピングモータはパルスモータとも呼ばれていて、電気的なパルス信号を適切な順序でコイルに与えると動く。パルス信号を与えるとモータの軸がステッピングモータ固有のステップ角で回転する。回転角度と回転速度が与えるパルスの信号の回数と周期によって決定する。

回転角度＝ステップ角（度）×パルス数

回転速度[rpm]=（ステップ角（度）/360）×パルス速度[Hz]×60[s]

通電状態（励磁状態）で大きな静止トルクが得られ、通電しない状態（無励磁状態）でも小さな静止トルクが得られる。また、ギヤを取り付けてある物を使用すると許容トルクや最大トルクが大きくなる他、基本ステップ角を細かくすることが可能である。このような正確な回転角度制御のできる特性があるため位置制御の用途で多用されている。また、パルスとは電圧の ON/OFF が繰り返される電気信号である。ON/OFF の１サイクルを１パ

ルスと数え、１パルスが命令されると基本ステップ角だけモータ軸が回転する。その他には、ブラシレスモータなので機械的な接触子がなく長寿命である。

今回使用したステッピングモータはユニポーラ型で二相励磁方式での基本ステップ角は１．８°の物を使用した。200 パルス与えるとモータの軸が一回転することとなる。ユニポーラ型は二本の電源端子と、４本の信号端子があり周囲のコイルに対して順番にパルス状の電流を流すことにより回転磁界を発生させマグネットローターが回転する。以下の表３－１に使用したモータの仕様を示し、図３－１に今回使用したステッピングモータとユニポーラ型ステッピングモータの構造を示す。

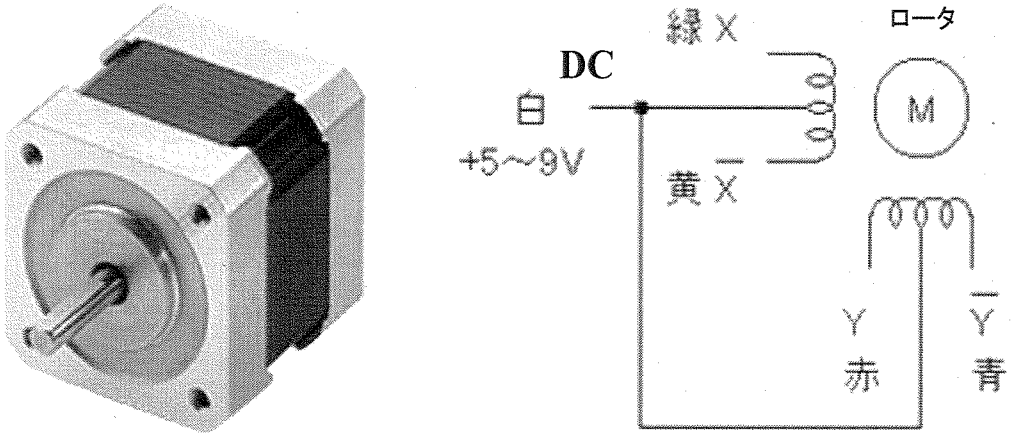

図３－１　ユニポーラ型ステッピングモータ

表３－１　本研究で使用したステッピングモータ

表３－１　本研究で使用したステッピングモータ

オリエンタルモータ株式会社	
二相ステッピングモータ(6本リード線)	
形式	PK243-01A
励磁最大静止トルク	0.16N・m
ローター慣性モーメント	35×10^{-7}kg・m^2
定格電流	0.95A/相
電圧	4V
巻線抵抗	4.2Ω/相
基本ステップ角	1.8°
電源入力	単相 100V＋－15%
	50/60Hz 1A
質量	モータ 0.21kg
	ドライバ 0.47kg

３．２　ステッピングモータ動作特性

電気的なパルス信号を回転させたい分送ることによりモータ軸の回転角度がその送られたパルス数だけ基本ステップ角回転し止まる。複数のコイルに位相が異なる周期的な電流を流すと、合成される磁束の向きを回転させることができる。これが回転磁界でありこれを発生させ、磁力によりローターを引き付けて回転する。ローターが回転することによりモータ軸も回転する。回転軸を正転（軸を上から見て時計回り）、逆転（反時計回り）させたい時は各コイルに流すパルス信号の順番を変えることにより簡単に制御できる。正転の場合は図３－２のような励磁シーケンスのように、$\overline{Y}X$, XY , $Y\overline{X}$, $\overline{X}\overline{Y}$ の順にパルス信号を入力する。逆転の場合は図３－３のように$\overline{Y}X$, $\overline{X}\overline{Y}$, $Y\overline{X}$, XY という順にパルス信号を入

力する。この励磁方式は二相励磁方式のステッピングモータ励磁シーケンスであり、図に示した流れのとおり１の部分でステッピングモータドライブ回路に電圧を印加することによりステッピングモータを正転、逆転することができる。このようにパルス信号を与えるたびに決められたステップ角だけ回転する。今回使用した二相励磁方式は二相の信号を同時に励磁するので消費電力はあがるが、得られるトルクが大きくなる。

時間	0	1	2	3	4	5	6	7	8
X	1	1	0	0	1	1	0	0	1
Y	0	1	1	0	0	1	1	0	0
$\overline{X}$	0	0	1	1	0	0	1	1	0
$\overline{Y}$	1	0	0	1	1	0	0	1	1

図　３－２　正転の場合の励磁シーケンス

時間	0	1	2	3	4	5	6	7	8
X	1	0	0	1	1	1	0	0	1
Y	0	0	1	1	0	1	1	0	0
$\overline{X}$	0	1	1	0	0	0	1	1	0
$\overline{Y}$	1	1	0	0	1	0	0	1	1

図　３－２　逆転の場合の励磁シーケンス

４．　ステッピングモータ駆動プログラムの作成

４．１　MPLAB の概要

プログラムを作成するにあたり、使用するステッピングモータドライブ回路には電気的なパルス信号を出力する必要がある。今回はワンチップマイコンをもちいてその電圧出力をプログラムにより実行した。MPLAB C18 を使用してプログラムの作成を行った。

MPLAB C18 は米マイクロチップ社が提供する、組み込みシステム用統合開発環境ソフトウェアのことである。Windows95/98/Me/2000/XP/Vista/7 上で動作するテキストエディタとコンパイラを持ち合わせている。全 PIC シリーズに書き込むプログラムを作成でき、C コンパイラを統合して書いたプログラムをコンパイルできる。また、コンパイルしたファイルをＰIC に書き込むなどができる。開発言語もアセンブラや MPLIB そして C 言語などがある。今回は C 言語を用いてプログラムの作成を行った。

４．２　ステッピングモータを正、逆回転させるプログラム

ステッピングモータを回転させるには電圧出力をステップ信号としなければならない。正転させる場合のフローチャートを図４－１に示す。動作確認のために LED 点灯回路を作成した。LED の点灯の順番と二相励磁方式の特徴である二点が同時に点灯しているかを確認した。図４－２に LED 点灯回路を示す。このフローチャートの PORTA =12 の出力と PORTA =3 の出力を入れ替えると逆転を行うフローチャートとなる。ステッピングモータを正転させるプログラムをプログラムリスト１に示す。

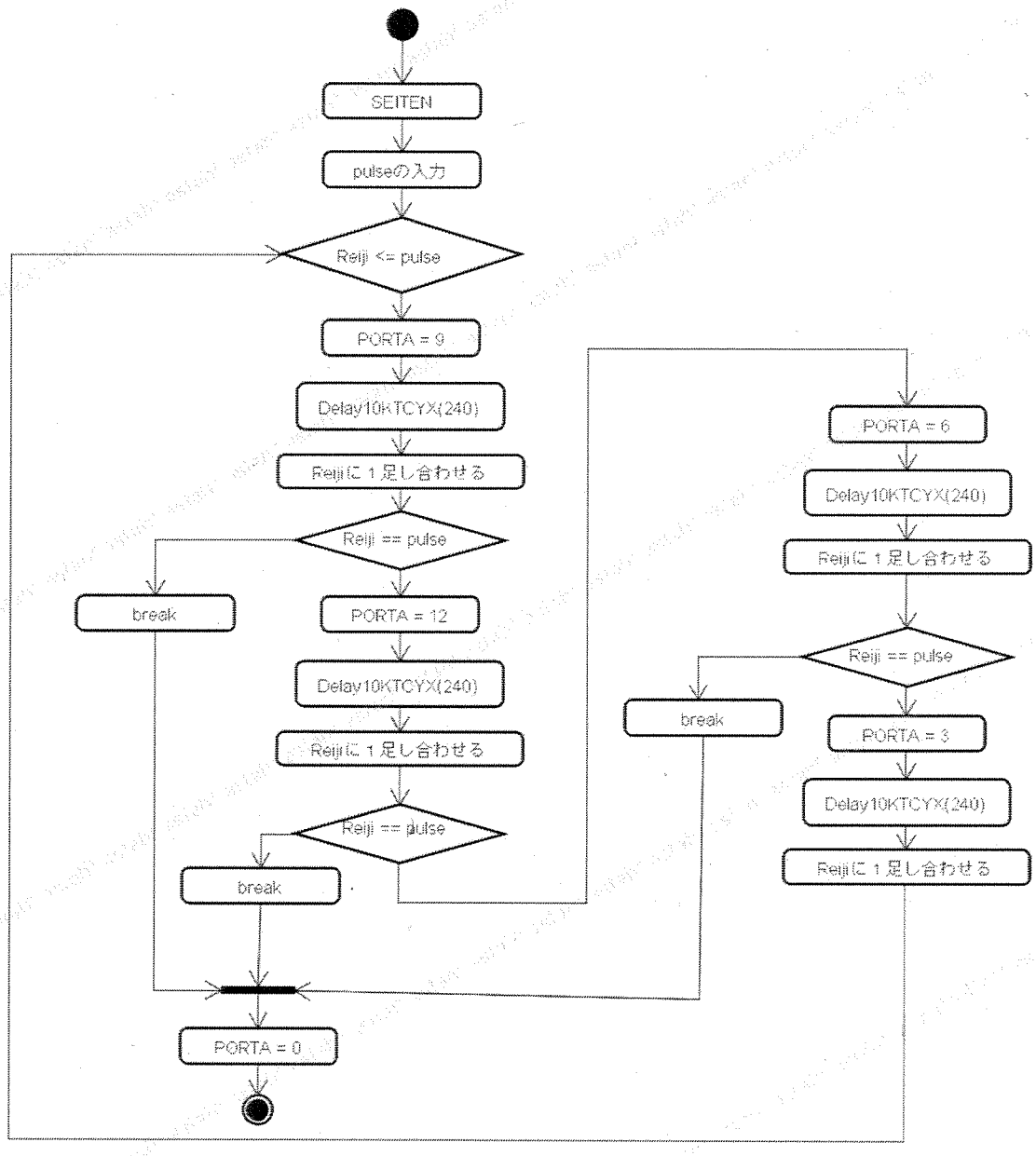

図４－１　正転用パルス信号確認の LED 点灯フローチャート

LED を点灯させるフローチャートの SEITEN では、与えられたパルス信号数が正の数であるとこの動作に入る。これはステッピングモータ軸を正転させる用のフローチャートとなる。Reiji ではステッピングモータのコイルが何回励磁を行ったかを示す変数である。これを与えられた pulse の数だけ励磁を行いステッピングモータを回転させる。PORTA=9 などで各ピ

ンに出力を行わせている。図３－１の正転の場合の励磁シーケンスピンの０時間をみると１００１というように出力を行っていて、それをPICのピンの出力で行わせるため、RA0からRA3までの４ピンを使用して行わせた。二進数の１００１を１０進数に変換すると９となる。PORTA＝９はRA０とRA3を出力するということになる。Delay10KTCYX(240)という関数では、時間待機をさせるものなのでこれを用いてパルス幅を設定することができる。10KTCY(120)では200msec待機させることができる。10KT=0.05μsec×４×1000=2msecとなる。Breakは関数であり、繰り返し処理を終わらせるために使用した。

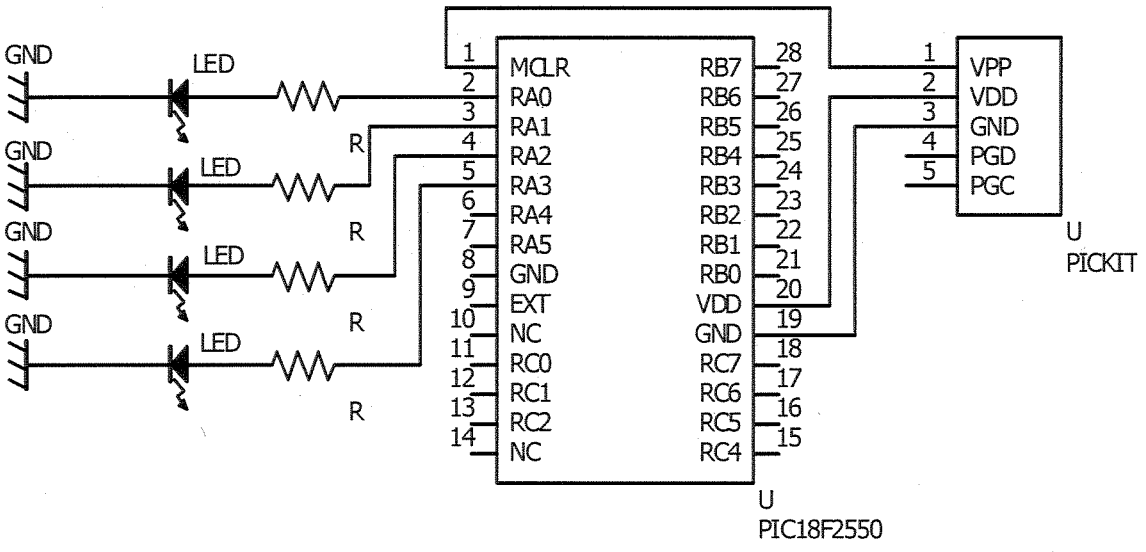

図４－２　LED点灯回路

フローチャートのプログラムを実行した結果により点灯したLEDは図４－３のようになった。黒く塗りつぶしてあるLEDが点灯したということとする。この結果より正転の場合の励磁シーケンス通りにLEDを点灯させることができた。

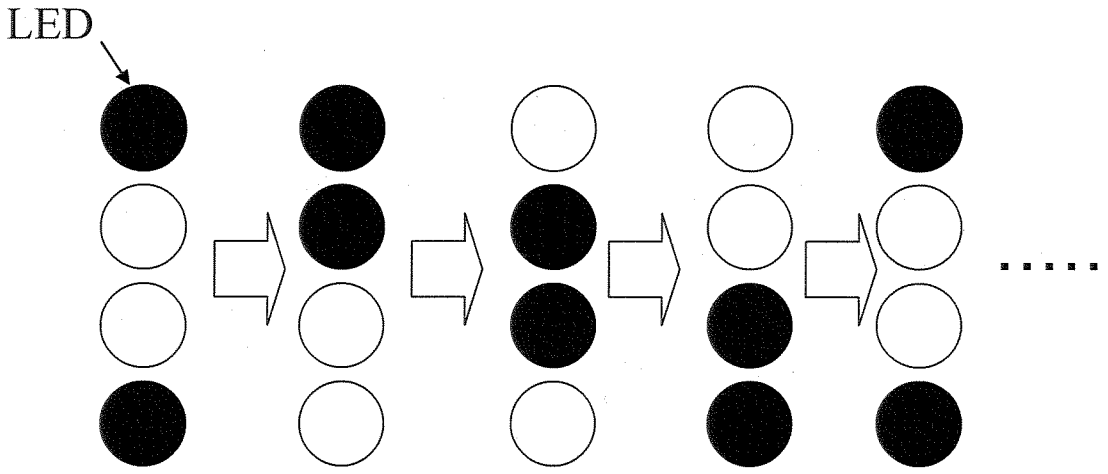

図４－３　正転用の LED の点灯順番

次に逆転用のプログラムで LED 点灯を確認したところ、逆転の場合の励磁シーケンス通りに点灯できた。

この結果から正、逆回転を行わせる場合の PIC の各ピンが行う電圧出力を横軸を経過時間、縦軸を出力電圧として図４－４に正転の場合、図４－５に逆転の場合を示す。

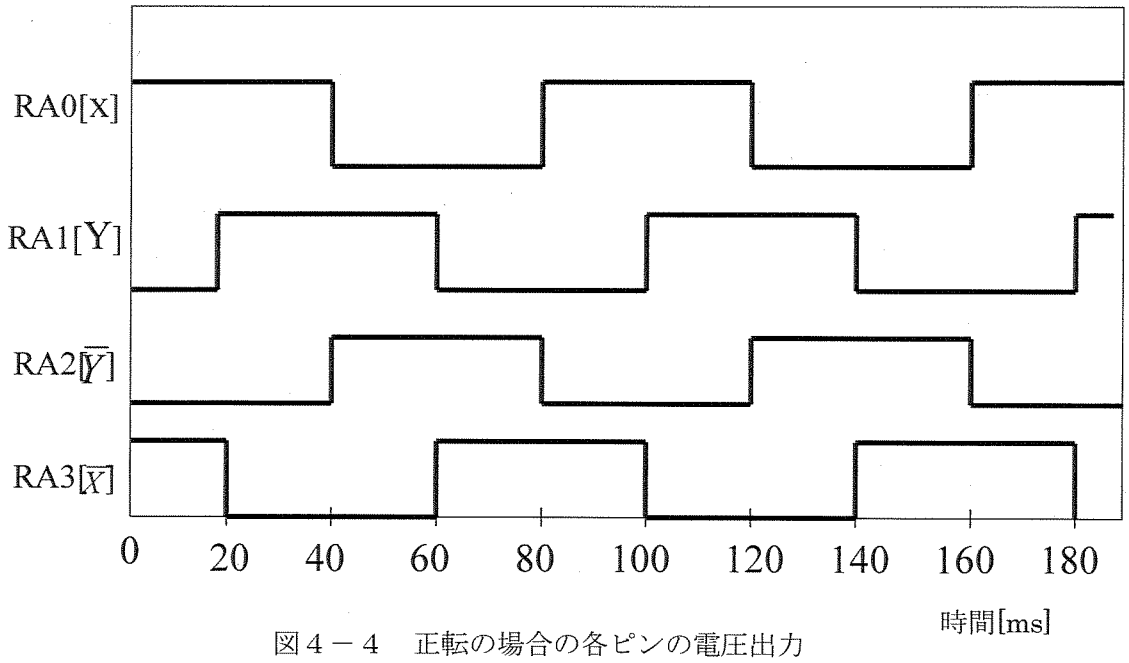

図４－４　正転の場合の各ピンの電圧出力

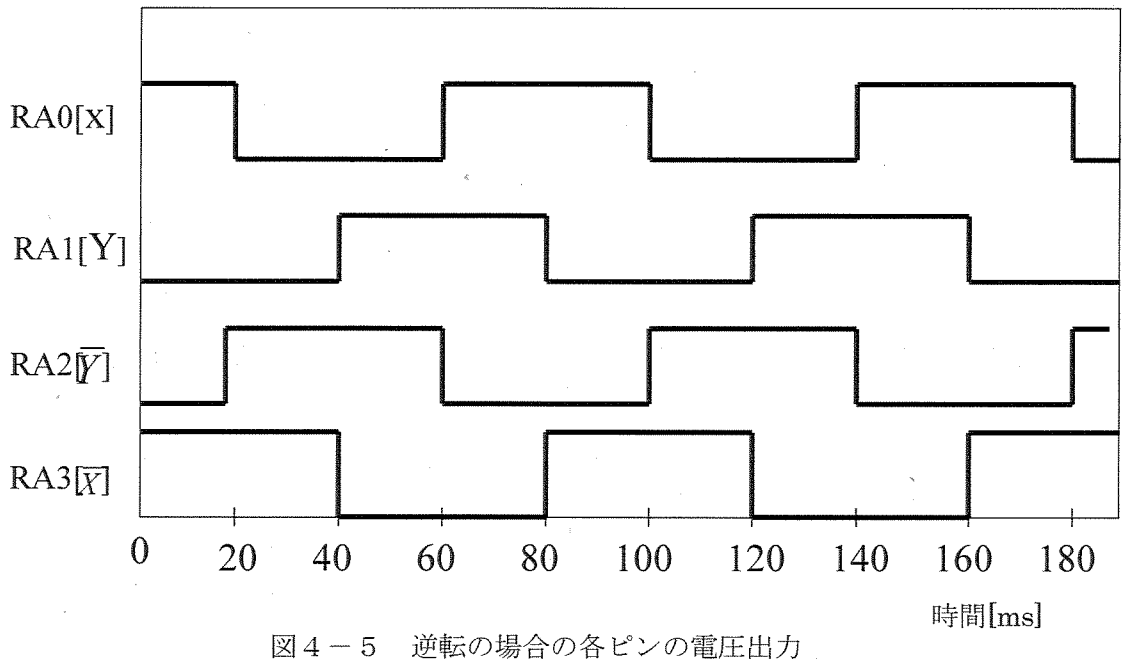

図４－５　逆転の場合の各ピンの電圧出力

以上の結果から電気的なステップ信号を出力できた。また位相が異なったパルス信号を作り出すことができ、二相励磁方式の信号を作成することができた。このプログラムを利用して次にステッピングモータを回転させた。ステッピングモータドライブ回路とワンチップマイコンを接続した回路図を図４－６に示す。使用した FET にはモータ駆動のために大電流が必要になるためパワーMOSFET を使用した。他のパワーデバイスと比較してもスイッチング速度が速いうえ低電圧領域での変換効率が高い。

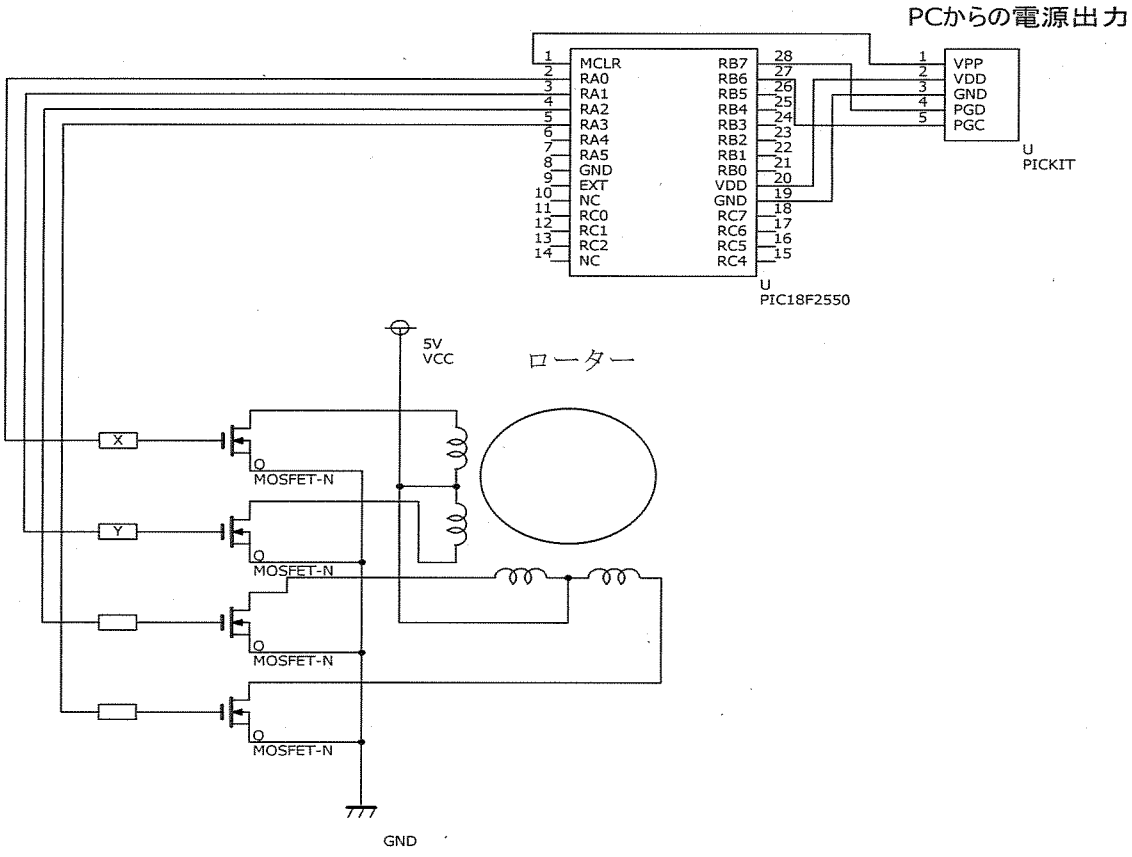

図４－６　ステッピングモータドライブ回路

５．　光軸調整に使用する式

５．１　反射鏡の角度調節

反射鏡の傾きを変更する場合、反射鏡固定版に三点の基準をとりその位置にネジを付けた。そのネジをステッピングモータで回転させることにより角度調節を行えるようにした。その基準点は図２－４の右図のステッピングモータの位置に正三角形の位置関係となるようにする。なお角度調節を行う際に反射鏡の中心部がずれないように三点を基点として角度変更を行えるように関係式の導出を行った。

反射鏡の中心部に光が当たるとして、それを反射させ壁に当てる。反射鏡の傾きをかえ反射光の移動が円の形をした軌道になる角度関係式の導出をした。図5－1にその概要図を示す。X方向に傾くときの角度をα、Y方向に傾くときの角度をβとした、各ステッピングモータの軸の中心から

反射鏡の中心までの長さをLとした。反射光が壁に当たるまでの距離をZとし、描く円の半径をrとして式の導出を行った。反射鏡が角度α、β変化して傾いたときの反射光が移動する距離の関係を図5－2に示す。

X方向に反射鏡がα傾いたときの、壁にあたった反射光の移動距離はZ tan α

Y方向に反射鏡がβ傾いたときの、壁にあたった反射光の移動距離はZ tan β

三平方の定理より

$$r^2 = (Z\tan\alpha)^2 + (Z\tan\beta)^2 \quad \cdots(1)$$

この式からαが変化したときのβの値が算出できるようにする。以下にその計算過程とその関係式を示す。

$$Z^2 \tan^2\alpha + Z^2 \tan^2\beta = r^2 \quad \cdots(2)$$

$$\tan^2\beta = \frac{r^2}{Z^2} - \tan^2\alpha \quad \cdots(3)$$

$$\beta = \tan^{-1}\sqrt{\frac{r^2}{z^2} - \tan^2\alpha} \quad \cdots(4)$$

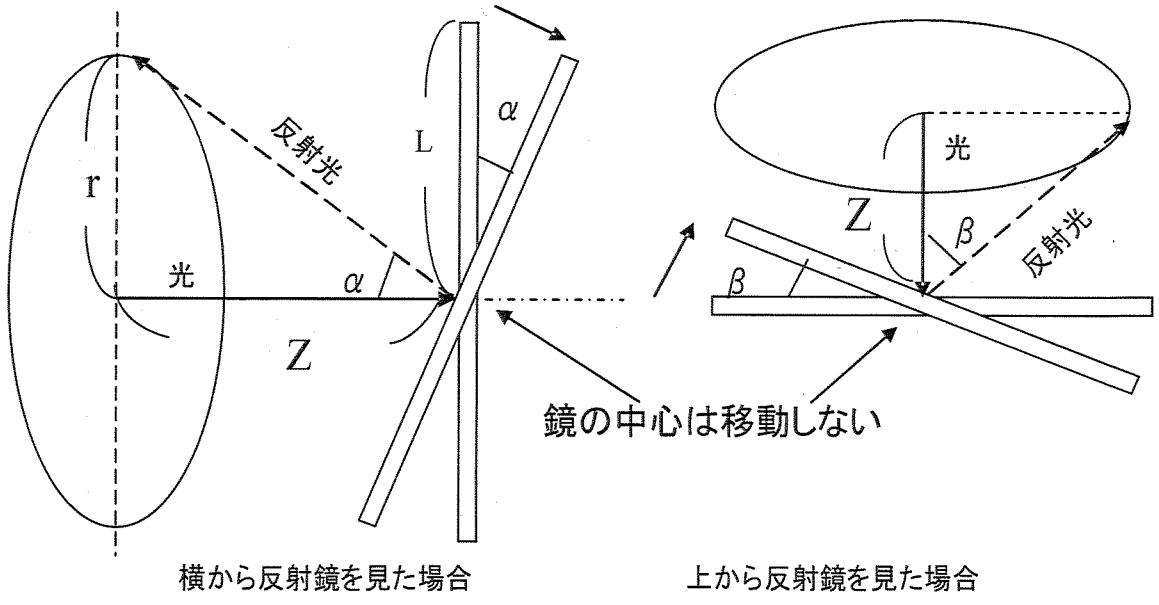

図　5－1　反射鏡を用いた光軸調節の概要図

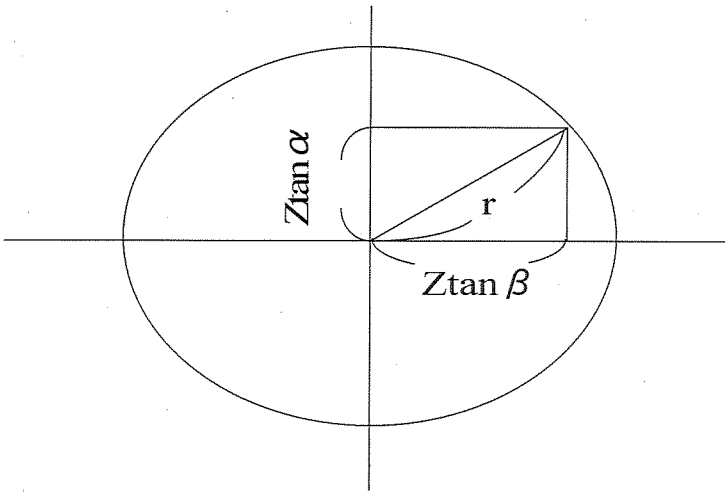

図　5－2　反射鏡の角度変化による反射光移動

この求めた円を描く時のαとβの関係式にrとzに適当な数値を代入し反射光がどのような動きをするかを数値として確認した。αが0～30°変化した場合、それに対応したβの値を計算し表にまとめた。そのデータを用いて図5－2のようにαとβのそれぞれの角度が与えられて移動する距離を求めグラフにした。図5－3に横軸をβ変化したときの光の移動距離、縦軸にα変化したときの光の移動距離として示す。

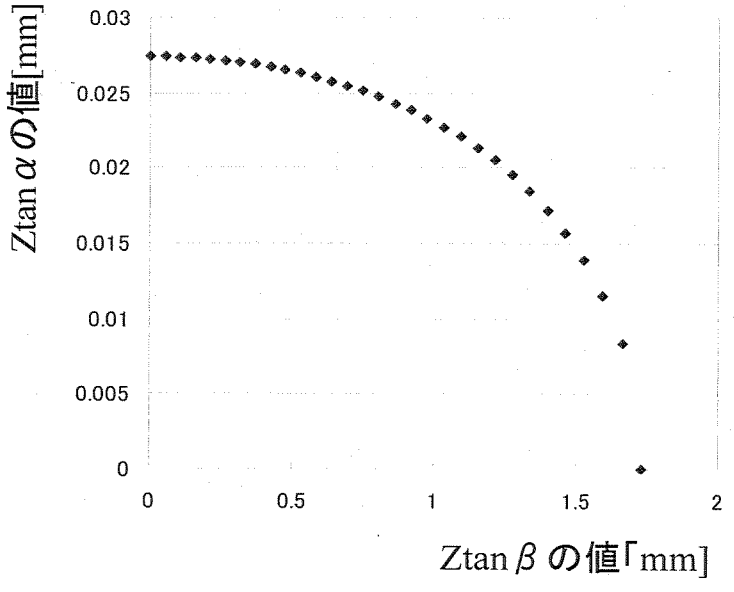

図5－3　計算によるαとβに対応した光の移動距離

5．2　反射鏡の傾きによる移動距離

本研究では反射鏡を傾かせる方法として、ステッピングモータを回転させることにより、カップリングにより連結されたネジが回転する、ネジが回転を行いそれによる移動により反射鏡を傾かせる。よってネジの移動距離によって反射鏡の傾きを変えることができる。そのために指定されたαやそこから求められたβの角度が与えられた時、その角度になるようにそれぞれのステッピングモータを回転さるためのネジの移動距離を求めた。

図5－4に各ステッピングモータから反射鏡固定板の位置関係を示す。反射鏡固定板に正三角形の位置関係になるようにステッピングモータを設置している。反射鏡の中心の位置からステッピングモータL1までの距離をLとした場合、中心点からX軸とY軸をとりその他2つの位置関係を求めた。

正三角形の中心点から下の辺までは1/2なのでこのようになり、Y軸からステッピングモータL2の長さの算出の仕方は三平方の定理より求めた。

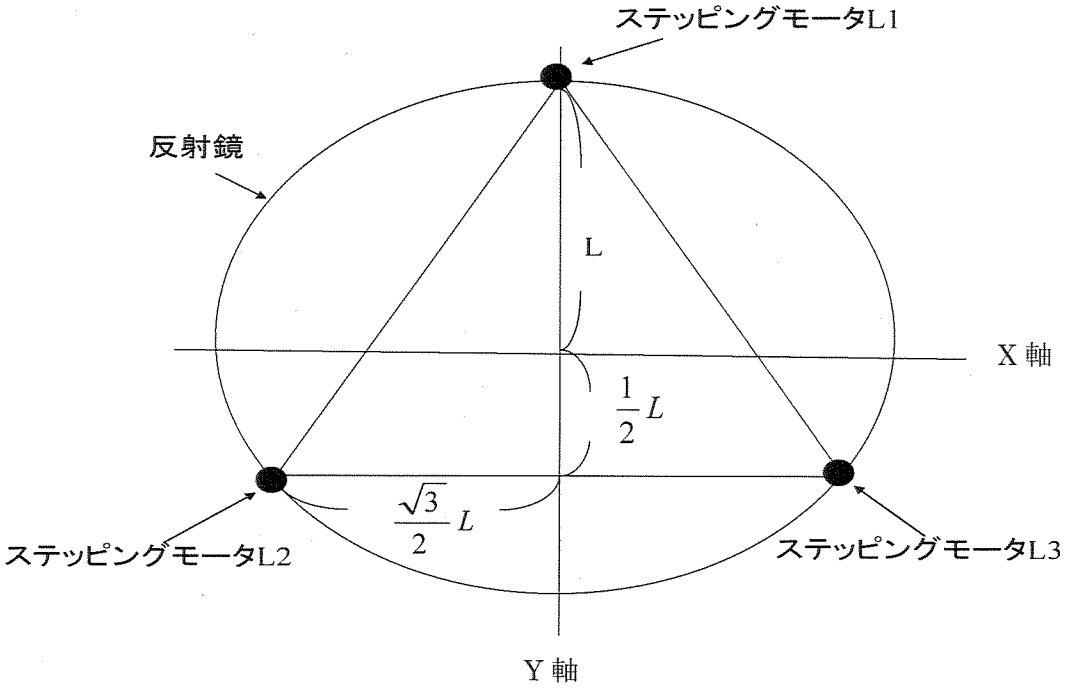

図５－４　反射鏡固定板から各ステッピングモータの位置関係

この値を使用して角度変化に対応したネジの移動距離の算出を行った。図５－１の横から反射鏡を見た場合のように傾く時、角度変化するのはαであるその時のそれぞれのネジの移動距離を以下のようにした。

$$L1 = L \tan \alpha \quad \cdots(5)$$

$$L2, L3 = -\frac{L}{2} \tan \alpha \quad \cdots(6)$$

また図５－１の上から反射鏡を見た場合のように傾かせた時、変化するのはβである。それぞれのネジの移動距離の算出式を求めた。このように傾かせる場合、Ｌ１は使用せずにL2とＬ３の伸び縮みにより傾かせる。以下にその関係式を示す。

$$L2 = \frac{\sqrt{3}}{2} L \tan \beta \quad \cdots(7)$$

$$L3 = -\frac{\sqrt{3}}{2} L \tan \beta \quad \cdots(8)$$

以上の式を利用してαを円を描くように変化させていく場合、それぞれのステッピングモータは計算上ではどのような動きをしてネジの回転移動をしているかをＬとＺにてきとうな値をいれて確認した。図５－５に横軸に変化時間、縦軸に垂直にあった状態から傾いた移動距離とした。反射鏡が傾いた場合、当たる壁に近づいてゆく時を正とし壁とは反対方向にゆく場合を負とした。

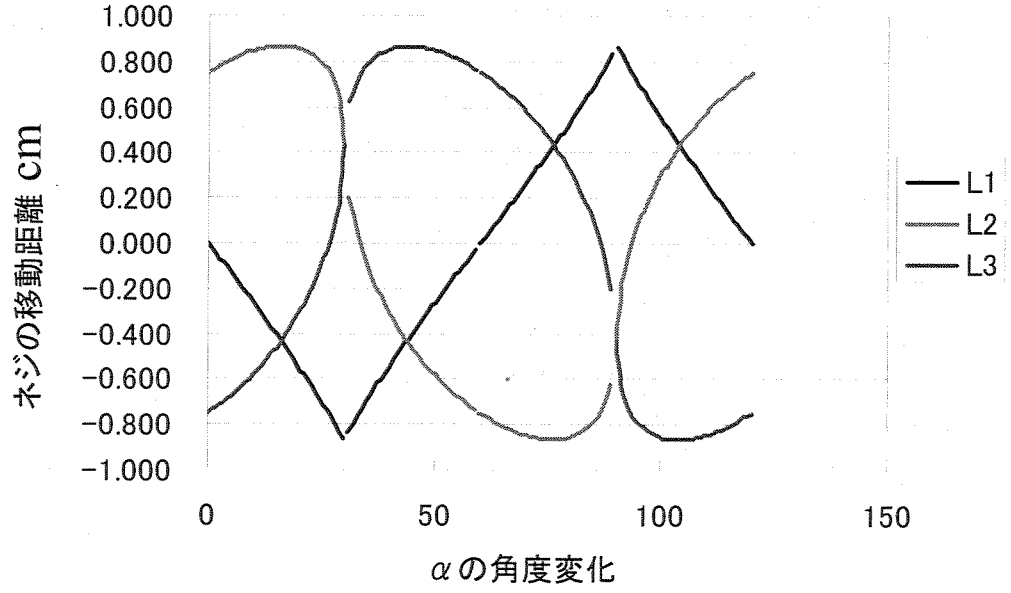

図５－５　円を描く場合の各ネジの回転移動の変化

６．　反射鏡の傾き調節プログラム

６．１　反射光による縦方向、横方向移動

ステッピングモータをプログラミングされたPICをもちいて回転させ、反射鏡の傾きを制御した。５．２で求めた式を利用してαとβを入力するとそれに応じてネジ移動距離を算出し、そのように動くようパルス信号数を決定し正転もしくは反転するプログラムの作成を行った。パソコンからαとβの値を入力すると、それに対応した角度調節の動きをするようにした。

今回はパソコンからUSBケーブルを通しRS-２３２Cにデータを送りさらにそこからレベルコンバータを通して信号を整理し、PICにパソコンで入力したデータを読み取らせるという工程を取った。値はαとβの値を入力しその角度になるようにステッピングモータを動かした。図６－１にそのデータ移動の様子を示す。

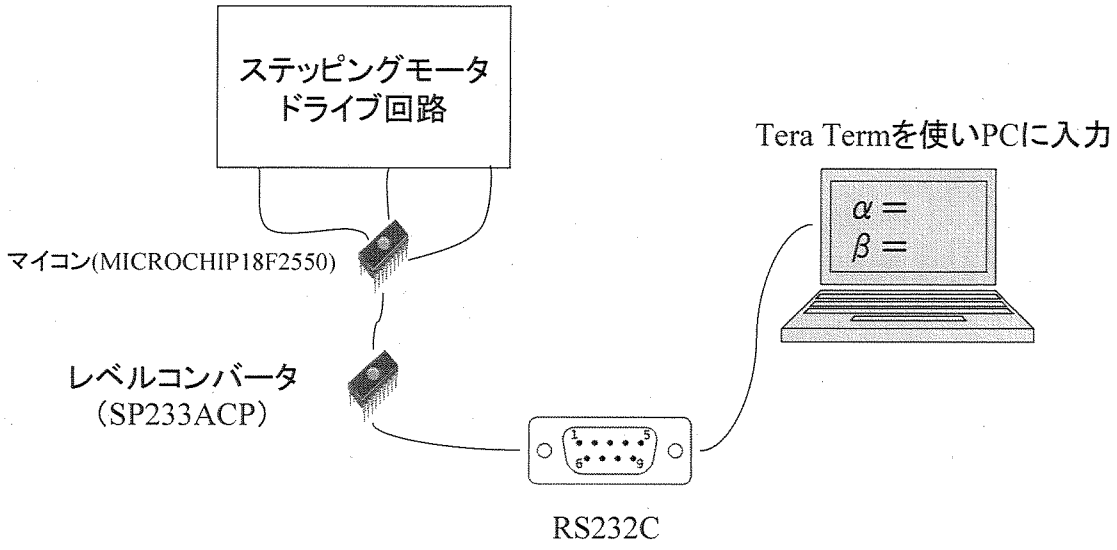

図６－１　パソコンによるステッピングモータ制御の通信方法

６．１．１　RS232C

今回使用したRS232C(Recommended Standard 232)はシリアル通信方式のインターフェースの一つである。インターフェースはポートとも呼ばれるため、シリアルポートとも呼ばれている。パソコンなどのデータ端末装置とデータ回路終端装置とを接続してデータ伝送を行うための電気的、機械的な特性を定義したものである。パソコンで Tera Term(通信ソフト)を用いて入力したい数字をいれＵＳＢコネクタからＲＳ２３２Ｃに変換するケーブルを用いてデータを送信した。９ピンシリアルインターポートを使用した以下に使用した変換ケーブルとＲＳ２３２Ｃを図６－１に示す。

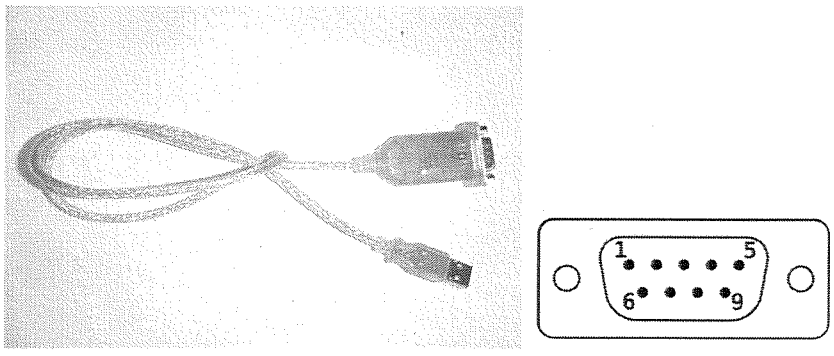

図６－１　RS２３２C と USB 変換ケーブル

６．１．２　SP233ACP

パソコンからマイコン（今回はMICROCHIP18F2550）へ信号を送りたい場合それらを直結することはできない。理由として、RS232C の電圧レベル（＋－９Ｖ）とマイコンの電圧レベル（＋５Ｖ）が異なるためである。その異なる電圧を変換して、パソコンとマイコンを直結できるようにするためにSP233ACP レベルコンバータを使用した。以下にその回路図を図６－２に示す。

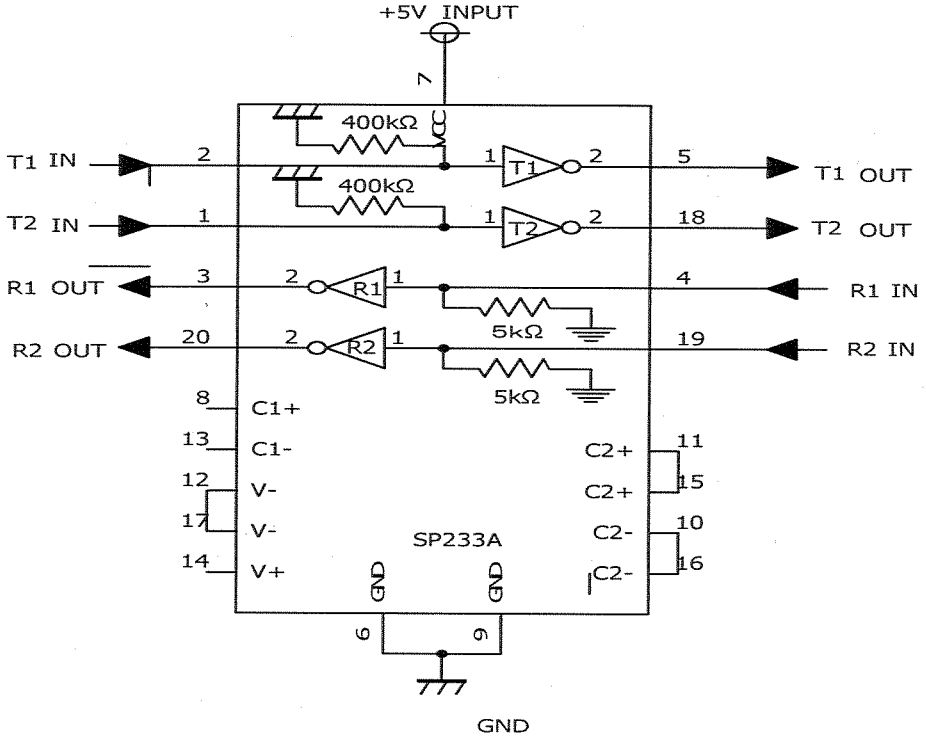

図６－２　SP233ACP レベルコンバータ

６．１．３　データ受信プログラム

上記の通信システムを用いてパソコンの入力画面でαとβの値を入力するとそのデータ信号をＰＩＣに送りそのように動くプログラムを作成した。Tera Term を使いCOM ポートによるシリアル接続を可能にしてパソコン通信を行なった。図６－３の回路のように結線しパソコンからのデータをマイコンで読み取り、ステッピングモータを駆動させた。

結線を行なうにあたり SP233ACP からくるデータを MICROCHIP18F2550 で受け取る際にデータの送受信を行えるピンが決まっている。外部からマイコンに受け取る場合は RC7 のピンを使用し、マイコンから送信する場合は RC6 のピンを使用した。

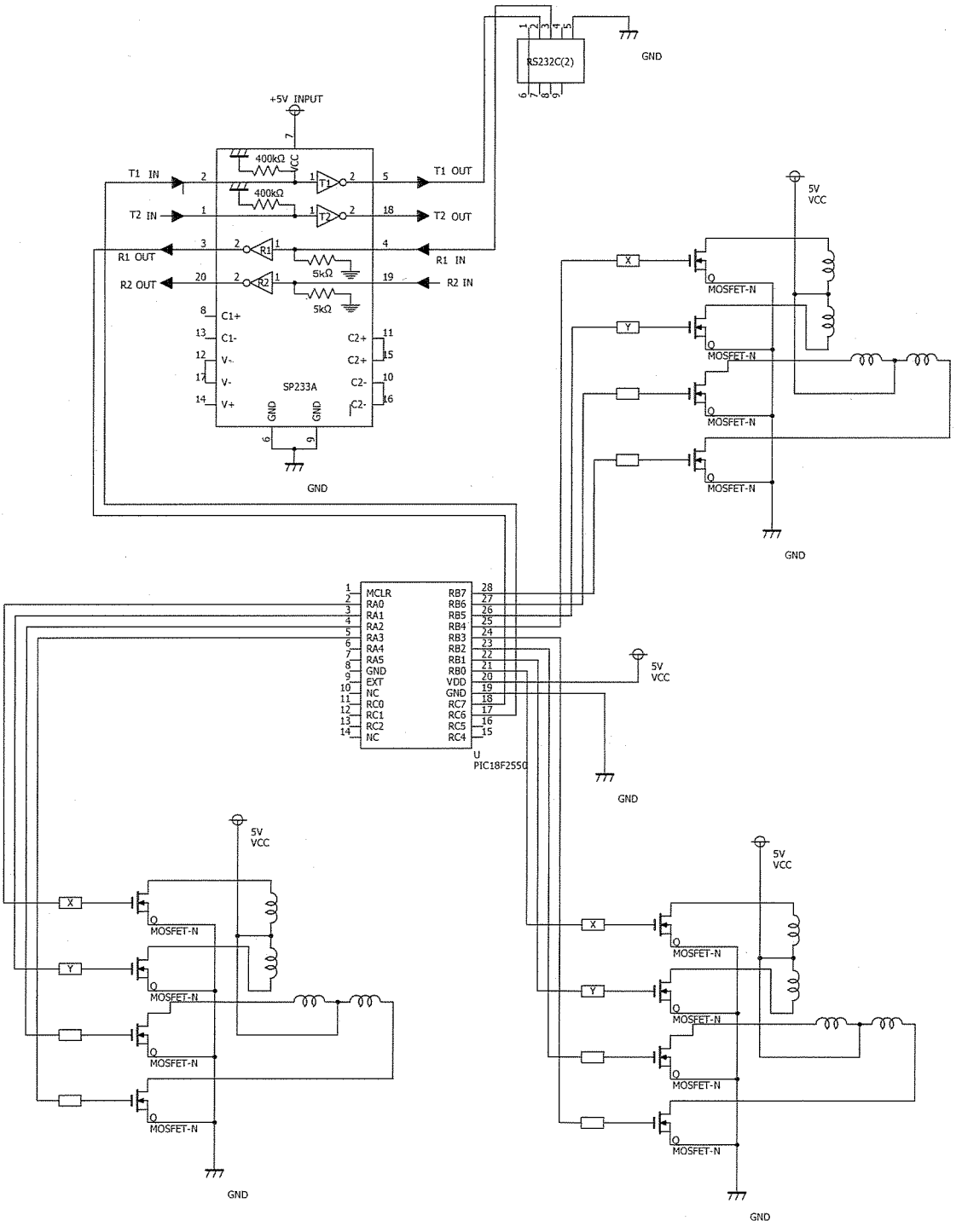

図６－３　パソコンからのデータ通信回路

Tera　Term から送信したデータを変換してマイコンで使用するのだが、レベルコンバータ（SP233ACP）から送られるデータは文字列であり送られてきたデータを直接使用できない。そのため文字列を数値に変換して使用した。パソコンから送信される文字数が何文字でも可能なようにプログラムを作成した。また、送信されたデータをパソコンに送り返してTera　Term で表示させた。以下にそのフローチャートを図６－４に示しプログラムをプログラムリスト２に示す。

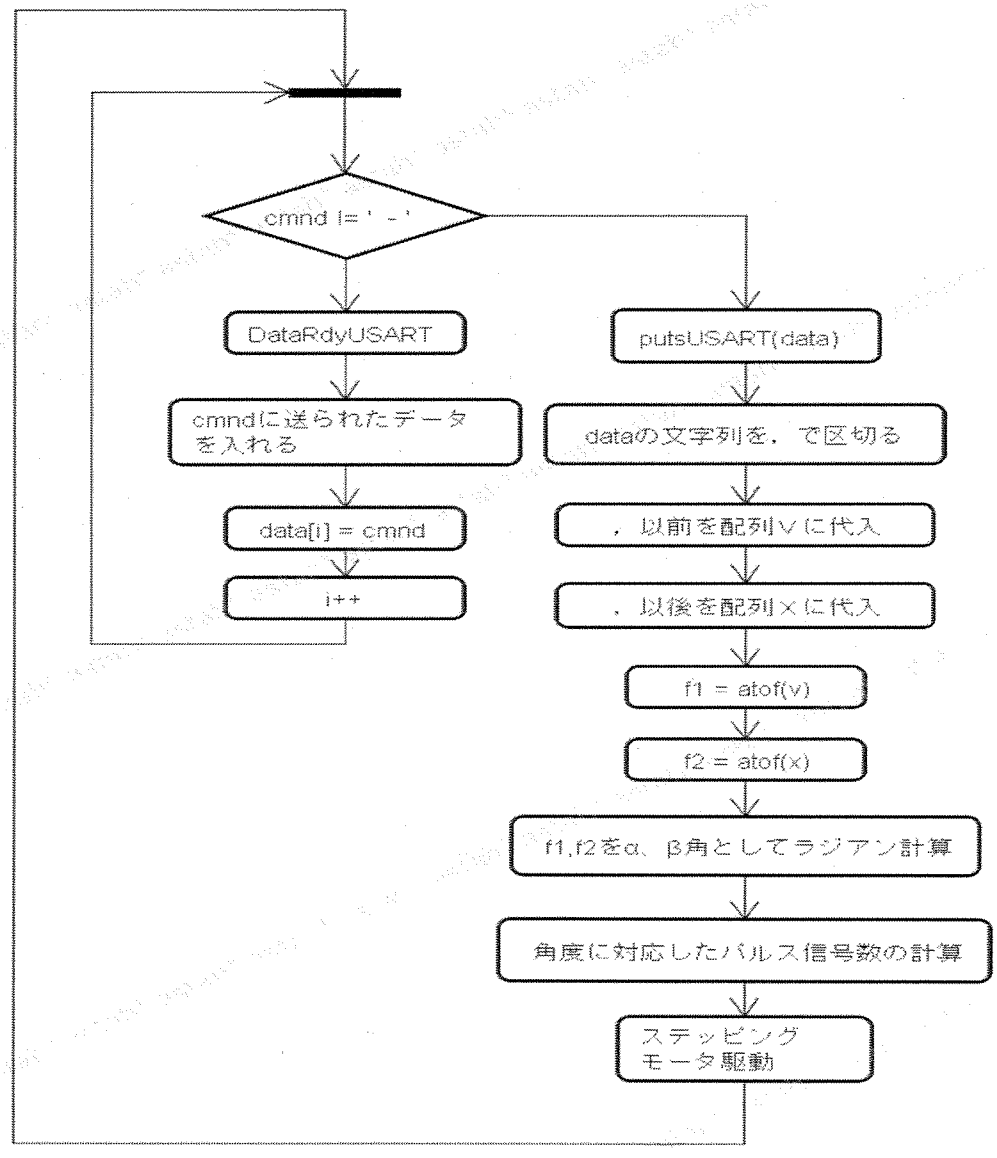

図６－４　データ通信用のフローチャート

今回はパソコンから送信するデータをネジの回転移動計算で用いるαとβの値とした。パソコンから送られてくる文字データをDataReadyUSARTという関数を用いて一文字ずつ受信できる。また、その送られてくる文字をdataという配列に代入していき’-‘がくるまでデータの受信をするようにした。putsUSARTではdataの中に入っている文字列をパソコンに送り返すことができTera termで表示させることが可能である。Dataに代入された文字列を　，がある前と後とで別々の配列に代入しなおす。そしてその配列をatofにより数値にする。関数であるatofでは文字列の数字の部分を数値に変換することができる。これを用いて文字を数値に変換してその値を角度α、βとして扱い反射鏡を傾かせる計算式で用いた。その計算式から各ステッピングモータに励磁するパルス信号数を求め、ステッピングモータを制御し反射鏡を傾かせた。

TeraTermに値を入力し実際にステッピングモータを動かし反射鏡を傾かせた。数値としてαに１５βに１５をいれてそれをラジアンに変換して、ネジの移動式に代入した。そしてステッピングモータを動かし反射鏡を傾けた。図５－２を参考に光の移動距離を考えてみると、今回はαとβの値は同じなので光のある点がX軸とY軸同じ距離離れていればいいことになる。実際に装置を動かしてみて光の移動を測定した。図６－５に反射鏡を調節した後の光の位置を示す。実験結果をみてみると光の出発点から、移動した光の距離がX軸とY軸が同じ長さ移動しているので正確に動作し調節できたと考える。

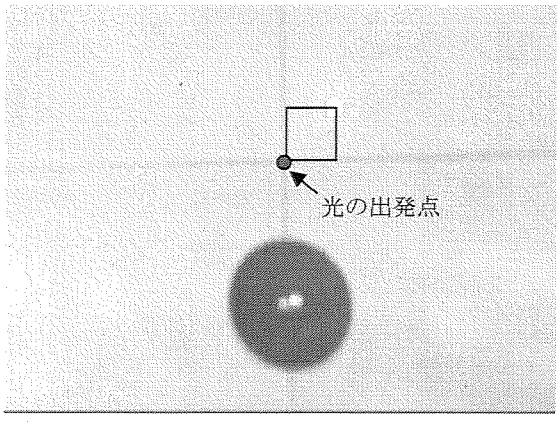

図６－５　実際の光の移動

６．２　反射光による円の軌道

反射鏡を傾けてあらゆる方向に反射光が行くか確認をするため、反射光で円軌道を描くことができれば条件を満たしていると考えた。よって反射光の移動で円の軌道を描くことを目的としたプログラムを作成した。５．１や５．２で算出した円を描く際のαとβの関係式やそれに対応したネジの移動距離を用いてステッピングモータ制御を行った。αが０～３０°間で変化する場合のフローチャートを図６－６に示す。

αが与えられそれを式に代入してβを求めた後に、ネジの移動距離を求めてそこからパルス信号数を求めている。今回使用したネジがＭ４のネジでありピッチ幅が 0.7mm であった。すなわちネジが一回転するのにネジが 0.7mm 伸び縮みするということである。ステッピングモータは 200 励磁を行うと一回転を行う。このことよりステッピングモータのコイルを一回励磁行うと 0.0035mm 伸び縮みすることができるということである。ネジの移動距離が求まったら 0.0035 で割ることによりパルス信号数を設定した。正転か逆転かの判定ではパルス信号数の値が正の値か負の値かどうかで判定し、それぞれのポート出力を行ないステッピングモータを回転させた。

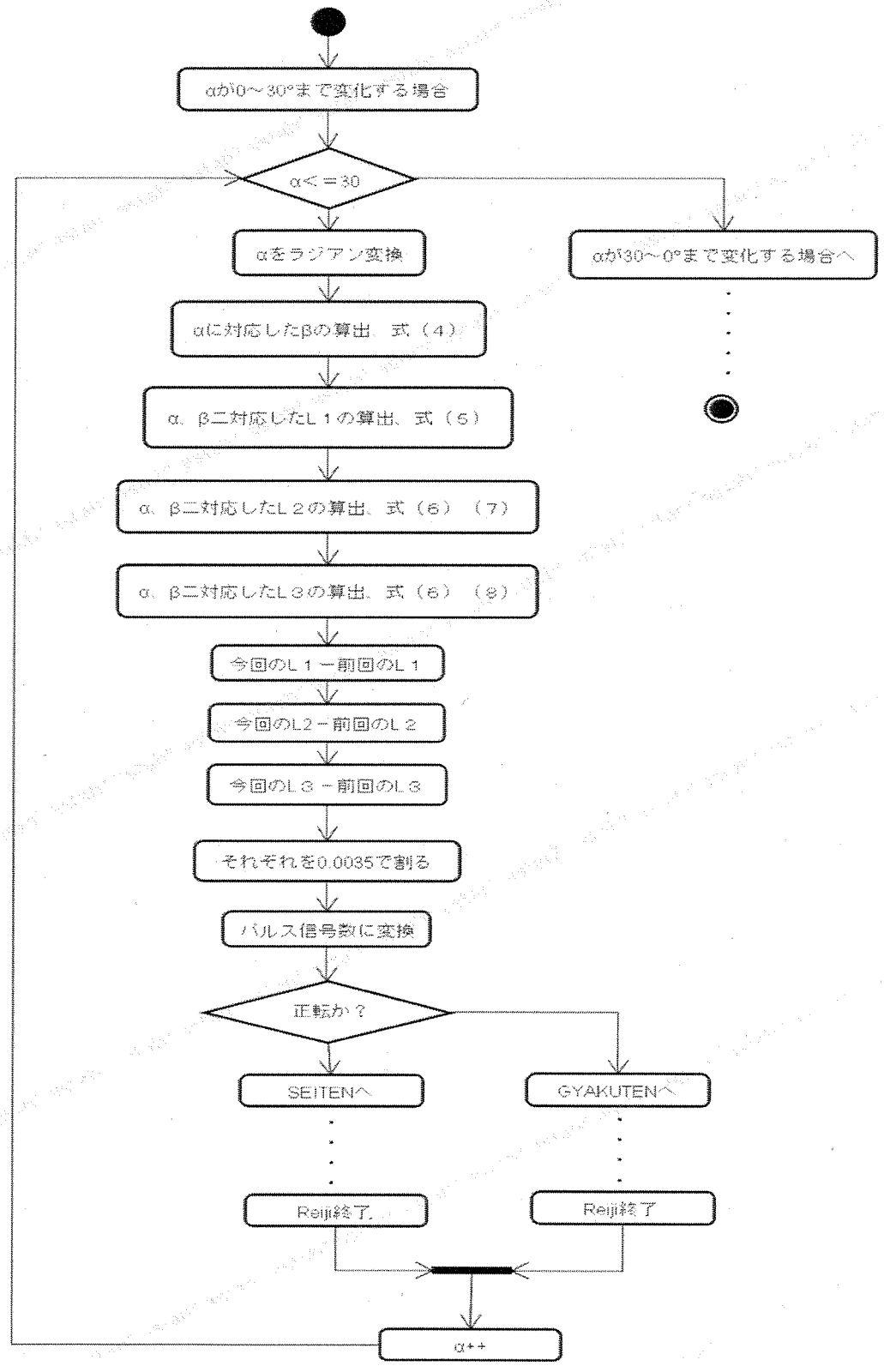

図６－６　反射光の移動で円を描くフローチャート

このフローチャートでは、αが０～３０°まで変化した場合のものである。円軌道を描く際にαが３０°に達した後に次は３０～０°というように角度が小さくなっていく。そのときにβは先ほどとは反対方向に動かなければ同じ動きをしてしまいもとの位置に戻ってしまうため円軌道は描けない。なので、描いている円の場所によってネジの伸び縮みの方向が変化してくる。αの変化に対応した反射鏡の傾く向きの関係を壁に向かう移動を行う場合のネジの伸びを＋として以下にその関係を示す。

αが０～３０°まで変化する場合

$$\beta = \tan^{-1}\sqrt{\frac{r^2}{z^2} - \tan^2\alpha}$$

$$L1 = -L\tan\alpha$$

$$L2 = \frac{L}{2}\tan\alpha + \frac{\sqrt{3}}{2}L\tan\beta$$

$$L3 = \frac{L}{2}\tan\alpha - \frac{\sqrt{3}}{2}L\tan\beta$$

αが３０～０°まで変化する場合

$$\beta = \tan^{-1}\sqrt{\frac{r^2}{z^2} - \tan^2\alpha}$$

$$L1 = -L\tan\alpha$$

$$L2 = \frac{L}{2}\tan\alpha - \frac{\sqrt{3}}{2}L\tan\beta$$

$$L3 = \frac{L}{2}\tan\alpha + \frac{\sqrt{3}}{2}L\tan\beta$$

αが０～-３０°まで変化する場合

$$\beta = -\tan^{-1}\sqrt{\frac{r^2}{z^2} - \tan^2\alpha}$$

$$L1 = -L\tan\alpha$$

$$L2 = \frac{L}{2}\tan\alpha + \frac{\sqrt{3}}{2}L\tan\beta$$

$$L3 = \frac{L}{2}\tan\alpha - \frac{\sqrt{3}}{2}L\tan\beta$$

αが-３０～０°まで変化する場合

$$\beta = -\tan^{-1}\sqrt{\frac{r^2}{z^2} - \tan^2\alpha}$$

$$L1 = -L\tan\alpha$$

$$L2 = \frac{L}{2}\tan\alpha - \frac{\sqrt{3}}{2}L\tan\beta$$

$$L3 = \frac{L}{2}\tan\alpha - \frac{\sqrt{3}}{2}L\tan\beta$$

このようにαが変化した時、それに対応した式を用いたプログラムを作成し実際にステッピングモータで動作させ円軌道を描かせた。図６－７に設定した円軌道と実際にステッピングモータ制御により描いた反射光を比較しその誤差を示した。横軸をαが変化した大きさ、縦軸を誤差とした。

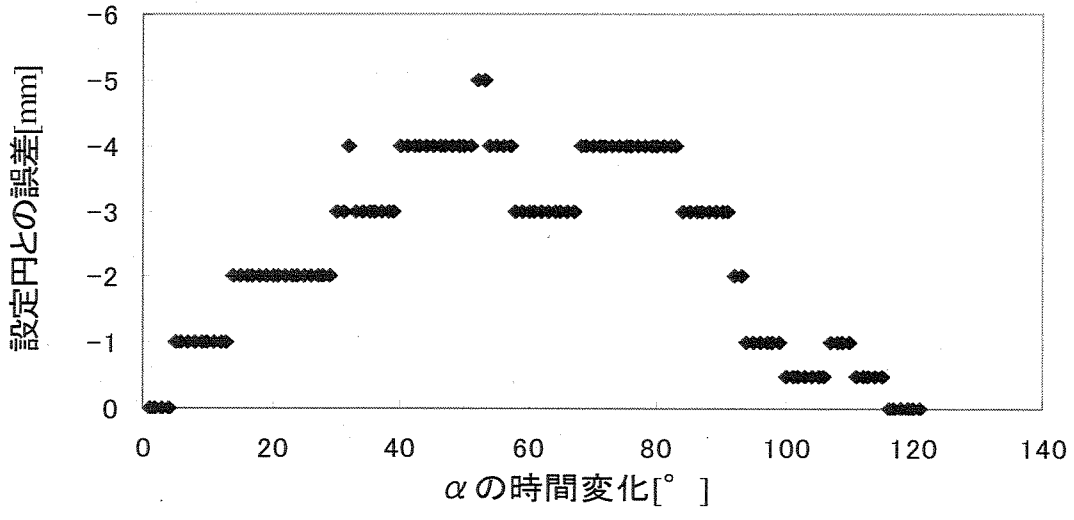

図６－７　設定円と実際に反射光が移動した軌道との誤差

7.　まとめ

反射鏡の傾きを調節する機器の製作を行った。反射鏡がネジの回転移動により正確に傾くようにバネを用いた。ステッピングモータドライブ回路を作成した。ステッピングモータを動かすためプログラムを作成しその動作確認を行った。ネジの回転移動によって反射鏡の傾きを調節するために、傾き角度とネジの回転移動による長さが対応した式を導き出した。ステッピングモータ設置箇所が三箇所あるのでそれぞれの場所の対応式を求めた。また、反射鏡を傾かせ反射光の移動により円を描くためにαとβの対応式を導き出した。実際に反射鏡調節システムを用いて反射光の移動により円を描いた。パソコンで入力したデータをマイコンで受信できるように、データの通信を行う回路を作成した。パソコンから角度αとβの値を入力しマイコンでその値を受信し数値として扱えるプログラムを作成した。値を入力すると反射鏡がその値の角度傾むき反射光が移動することを確認した。

8.　付録

8．1　プログラムリスト1

```
#include <stdio.h>
#include <p18f2550.h>
#include "delays.h"
#include <adc.h>
#include <timers.h>

#pragma config FOSC = HSPLL_HS                    //PIC18F2550 の動作設定
#pragma config WDT = OFF
#pragma config PLLDIV = 5
#pragma config CPUDIV = OSC1_PLL2
#pragma config USBDIV = 2
#pragma config PWRT = ON
#pragma config BOR = ON
#pragma config LVP =OFF
#pragma config VREGEN = ON
#pragma config MCLRE = ON
#pragma config PBADEN =OFF

static int Result,Ondo,flag;
int const LED_SEG[11] =
        {0x7E,0x0C,0xB6,0x9E,0xCC,0xDA,0xFA,0x0E,0xFE,0xCE,0x80};

void main(void)                                   //main 関数
{

    int i, parusu1 = 0,parusu2 = 0, parusu3 = 0;  //変数の宣言
        i = 1;
     TRISA=0;                                     //PORTA の入出力設定（0：出力、1：入力）
     TRISB = 0;                                   //PORTB の入出力設定
     TRISC = 0;                                   //PORTC の入出力設定
     PORTA = 0;                                   //PORTA 全ピン0
     PORTB = 0;                                   //PORTB 全ピン0
     PORTC = 0;                                   //PORTC 全ピン0
     Delay10KTCYx(250);                           //一秒待ち
```

```
    while(parusu1 <= 100)          //コイルに 100 回励磁を行う
        {                          //正転をする場合の PORTA の出力の流れ
            PORTA = 9;             //PORTA を出力
        Delay10KTCYx(240);         //一秒間待機
        parusu1 += i;              //励磁をした回数に 1 足し合わせる
    if(parusu1 == 100)             //励磁を何回目行ったか判定する
        {                          //100 回励磁をおこなっていたら while 分から抜ける
        break;
        }
        PORTA = 3;
        Delay10KTCYx(240);
        parusu1 += i;
    if(parusu1 == 100)
        {
        break;
        }
        PORTA = 6;
        Delay10KTCYx(240);
        parusu1 += i;
    if(parusu1 == 100)
        {
        break;
        }
        PORTA = 12;
        Delay10KTCYx(240);
        parusu1 += i;
      }

PORTA = 0;                         //PORTA をリセットする

while(parusu2 <= 100)              //コイルに 100 回励磁を行う
        {                          //正転の場合の PORTB の出力の流れ
        PORTB = 9;                 //PORTB を出力
        Delay10KTCYx(240);         //一秒間待機
        parusu2 += i;              //励磁をした回数に 1 足し合わせる
    if(parusu2 == 100)             //励磁を何回目行ったか判定する
```

```
        {
        break;                          //100 回励磁をおこなっていたら while 分から抜ける
        }
        PORTB = 3;
        Delay10KTCYx(240);
        parusu2 += i;
    if(parusu2 == 100)
        {
        break;
        }
        PORTB = 6;
        Delay10KTCYx(240);
        parusu2 += i;
    if(parusu2 == 100)
        {
        break;
        }
        PORTB = 12;
        Delay10KTCYx(240);
        parusu2 += i;
      }
PORTB = 0;                              //PORTB をリセット

while(parusu3 <= 100)                   //コイルに 100 回励磁を行う
        {                               //正転の場合の PORTB の出力の流れ
        PORTC = 65;                     //PORTC を出力
        Delay10KTCYx(240);              //一秒間待機
        parusu3 += i;                   //励磁をした回数に 1 足し合わせる
   if(parusu3 == 100)                   //励磁を何回目行ったか判定する
        {                               //100 回励磁をおこなっていたら while 分から抜ける
        break;
        }
        PORTC = 3;
        Delay10KTCYx(240);
        parusu3 += i;
   if(parusu3 == 100)
```

```
            {
            break;
            }
            PORTC = 6;
            Delay10KTCYx(240);
            parusu3 += i;
      if(parusu3 == 100)
            {
            break;
            }
            PORTC = 68;
            Delay10KTCYx(240);
            parusu3 += i;
           }
PORTC = 0;                          //PORTC をリセット
while(1)
{
 PORTA = 0;
 PORTB = 0;
 PORTC = 0;
}
}
```

８．２　プログラムリスト２

```
#include <stdio.h>
#include <p18f2550.h>
#include "delays.h"
#include <adc.h>
#include <timers.h>
#include <math.h>
#include <usart.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#pragma config FOSC = HSPLL_HS
#pragma config WDT = OFF
#pragma config PLLDIV = 5
#pragma config CPUDIV = OSC1_PLL2
#pragma config USBDIV = 2
#pragma config PWRT = ON
#pragma config BOR = ON
#pragma config LVP =OFF
#pragma config VREGEN = ON
#pragma config MCLRE = ON
#pragma config PBADEN =OFF
void Gattai(char r_data[]);
void getsUSART(char*buffer,unsigned char len);
//unsigned long ToDec(const char str[ ]);

static int Result,Ondo,flag;
int const LED_SEG[11] =
        {0x7E,0x0C,0xB6,0x9E,0xCC,0xDA,0xFA,0x0E,0xFE,0xCE,0x80};

char CmdMsg[] = "¥r¥nCommand";
char InpMsg[] = "  Input= ";
char ContMSG[] = "¥r¥nInput 10 Character = ";
char Buffer[15] = {0x0D,0x0A,0,0,0,0,0,0,0,0,0,0,0,0,0};
char output[15];
```

```
void main(void)
{
                                                        //変数の宣言
    char data[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    float f1,f2;
    int i, c, f, g, j, k, t, u, parusu1 ,parusu2 , parusu3, z;
    double a, d, e, b,h,y, rad1, rad2, L1, L2, L3,p,q,r;
    char cmnd;
    char*tp;
    char*tp2;
    char v[5]={0,0,0,0,0},x[5]={0,0,0,0,0};

    k = 1;
     TRISA=0;                          //PORTA の入出力設定（０：出力、１：入力）
     TRISB = 0;                        //PORTB の入出力設定
     TRISC = 0b11000000;               //PORTC の入出力設定(二進数でピン設定)
     PORTA = 0;                        //PORTA 全ピン０
     PORTB = 0;                        //PORTB 全ピン０
     PORTC = 0;                        //PORTC 全ピン０

    OpenUSART(USART_TX_INT_OFF&USART_RX_INT_OFF&      //USART 制御レジスタの設定
    USART_ASYNCH_MODE&USART_EIGHT_BIT&USART_CONT_RX&  //高速通信用に設定
    USART_BRGH_HIGH,155);                             //システムクロック周波数の設定

 while(1)                                             //無限に繰り返す
   {
       f1 = 0;                                        //変数をリセット
       f2 = 0;
       cmnd = 0;
       for(i = 0; cmnd != '-'; i++ )                  //配列の中に-が入るまで繰り返し
       {
              while(!DataRdyUSART());                 //受信待ち
              cmnd = ReadUSART();                     //一文字受信
              data[i] = cmnd;                         //受信した文字を配列に収納
```

```
}
putsUSART(data);                    //文字列折り返し送信
for( i = 0;  data[i] != ','; i++ )  //文字列を,を境に区切る
{
        v[i] = data[i];             //文字列の,より先の文字をｖに代入
}
v[5] = 0;                           //最後の文字をNULL文字とする
putsUSART(v);                       //配列Vの文字列折り返し送信
z = 0;
for( i += 1; data[i] != '-'; i++ )  //,以降の文字列を-が来るまで読み取り
{
        x[z] = data[i];             //,以降の値をXに代入
        z++;
}
x[5] = 0;                           //最後の文字をNULL文字とする
putsUSART(x);                       //配列Xの文字列折り返し送信

f1 = atof(v);                       //配列Vの文字列の数字を数値に変換
f2 = atof(x);                       //配列Xの文字列の数字を数値に変換
                                   //V,Xそれぞれの数値を角度α、βとする
rad1 = f1*3.141592/180;             //角度αをラジアン変換
rad2 = f2*3.141592/180;             //角度βをラジアン変換
a = -1.750*tan(rad1);               //L1のネジの移動距離の算出
d = 1.750/2.000*tan(rad1)+sqrt(3.000)/2.000*1.750*tan(rad2);
e = 1.750/2.000*tan(rad1)-sqrt(3.000)/2.000*1.750*tan(rad2);
                                   //L2,L3のネジの移動距離の算出
L1 = a;
L2 = d;
L3 = e;

b = 0.0035;                         //1パルス0.0035mmネジの移動
if(0 <= L1)                         //四捨五入の流れ
{
        p = 0.5;
        }
        else
```

```
        {
                p = -0.5;
        }
        if(0 <= L2)
{
                q = 0.5;
        }
        else
        {
                q = -0.5;
        }
        if(0 <= L3)
{
                r = 0.5;
        }
        else
        {
                r = -0.5;
        }
  c = (int)((L1 / b)+p);                    //L1 のパルス信号数を計算
  f = (int)((L2 / b)+q);                    //L2 のパルス信号数を計算
  g = (int)((L3 / b)+r);                    //L3 のパルス信号数を計算

  parusu1 = 0;                              //変数リセット
  parusu2 = 0;
  parusu3 = 0;

Delay10KTCYx(1000);                         //時間待機
Delay10KTCYx(1000);
Delay10KTCYx(1000);
Delay10KTCYx(1000);
Delay10KTCYx(1000);

        if(0 <= c)              //L1 の正転を行う場合の流れ
    {                           //これ以降はプログラムリスト１を見ると判る
        while(parusu1 <= c)
```

```
{
        if(parusu1 == c)
{
        break;
}
PORTA = 9;
        Delay10KTCYx(60);
        parusu1 += k;
                if(parusu1 == c)
        {
                break;
        }
        PORTA = 12;
        Delay10KTCYx(60);
        parusu1 += k;
        if(parusu1 == c)
        {
                break;
        }
        PORTA = 6;
        Delay10KTCYx(60);
        parusu1 += k;
        if(parusu1 == c)
        {
                break;
        }
        PORTA = 3;
        Delay10KTCYx(60);
        parusu1 += k;
}
        PORTA = 0;
}
else                            //L1 の逆転を行う場合の流れ
{                               //正転の PORT の出力順番を変更したのみ
        while(c <= parusu1)
{
```

```
            if(parusu1 == c)
    {
            break;
    }
                    PORTA = 9;
            Delay10KTCYx(60);
            parusu1 -= k;
            if(parusu1 == c)
            {
                    break;
            }
            PORTA = 3;
            Delay10KTCYx(60);
            parusu1 -= k;
            if(parusu1 == c)
            {
                    break;
            }
            PORTA = 6;
            Delay10KTCYx(60);
            parusu1 -= k;
            if(parusu1 == c)
            {
                    break;
            }
            PORTA = 12;
            Delay10KTCYx(60);
            parusu1 -= k;
    }
    PORTA = 0;
}

if(0 <= f)                          //L2の正転を行う場合の流れ
{
    while(parusu2 <= f)
    {
```

```
        if(parusu2 == f)
{
        break;
}
                PORTB = 9;
        Delay10KTCYx(60);
parusu2 += k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 12;
        Delay10KTCYx(60);
        parusu2 += k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 6;
        Delay10KTCYx(60);
        parusu2 += k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 3;
Delay10KTCYx(60);
        parusu2 += k;
}
        PORTB = 0;
}
else                            //L2の逆転を行う場合の流れ
{                               //正転のPORTの出力順番を変更したのみ
        while(parusu2 >= f)
{
        if(parusu2 == f)
```

```
{
        break;
}
                PORTB = 9;
        Delay10KTCYx(60);
        parusu2 -= k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 3;
        Delay10KTCYx(60);
        parusu2 -= k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 6;
        Delay10KTCYx(60);
        parusu2 -= k;
        if(parusu2 == f)
        {
                break;
        }
        PORTB = 12;
        Delay10KTCYx(60);
        parusu2 -= k;
}
        PORTB = 0;
}
if(0 <= g)                              //L3の正転を行う場合の流れ
{
        while(parusu3 <= g)
{
        if(parusu3 == g)
{
```

```
        break;
}
        PORTB = 0b10010000;
Delay10KTCYx(60);
        parusu3 += k;
                if(parusu3 == g)
        {
                break;
        }
        PORTB = 0b11000000;
Delay10KTCYx(60);
        parusu3 += k;
                if(parusu3 == g)
        {                                       break;
        }
        PORTB = 0b01100000;
        Delay10KTCYx(60);
        parusu3 += k;
                if(parusu3 == g)
        {
                break;
        }
PORTB = 0b00110000;
        Delay10KTCYx(60);
        parusu3 += k;
}
        PORTC = 0;
}
else                                    //L3の逆転を行う場合の流れ
{                                       //正転のPORTの出力順番を変更したのみ
        while(parusu3 >= g)
{
        if(parusu3 == g)
{
        break;
}
```

```
                PORTB = 0b10010000;
                        Delay10KTCYx(60);
                        parusu3 -= k;
                                if(parusu3 == g)
                        {
                                break;
                        }
                PORTB = 0b00110000;
                        Delay10KTCYx(60);
                        parusu3 -= k;
                                if(parusu3 == g)
                        {
                                break;
                        }
                        PORTB = 0b01100000;
                        Delay10KTCYx(60);
                        parusu3 -= k;
                                if(parusu3 == g)
                        {
                                break;
                        }
                        PORTB = 0b11000000;
                        Delay10KTCYx(60);
                        parusu3 -= k;
                }
                        PORTB = 0;
                }
        }
}
```

謝辞

本研究を行なうにあたり、多数の文献をお借りし、研究においては丁寧なご指導、また論文の推敲にも大変なお力添えを頂いた本校　電子制御工学科　由井四海　教官に深く感謝いたします。

参考文献

1) Tekurobo工作室 ソフトウェア編ステッピングモータ(2003)
URL:http://homepage1.nifty.com/rikiya/software/114stepping1.htm

2) 改訂版 電子工作のためのPIC18F本格活用ガイド　　加藤文明社 後閑 哲也 著

3) 入門ANSI-C 増補改訂版 実教出版株式会社 石田 晴久 著