富山商船高等専門学校電子制御工学科

卒業論文

サンフォトメーターの軌道計算を用いた追従性の向上

電子制御工学科 d03237

吉原　洸太

2007 年 1 月

目次

1　序章

1.1　背景 4

1.2　エアロゾル濃度の測定原理 4

1.3　サンフォトメーター 4

1.4　目的 5

2　軌道計算を用いた太陽追従

2.1　システム概要 5

2.2　軌道計算に用いる座標系について 6

2.3　太陽の黄経・黄緯の算出方法 8

2.4　地球の日心黄経から求める方法 8

2.4.1　ケプラーの方程式を用いたの求め方 8

2.4.2　日心直交座標系で表した天体の位置 10

2.4.2.1　日心直交座標系 10

2.4.2.2　軌道要素 10

2.4.2.3　xy 座標から日心直交座標への変換 11

2.4.3　惑星光行差による天体位置のずれ 12

2.5　S.Newcomb の理論から求める方法 12

2.5.1　太陽位置の略算式 12

2.5.2　黄道座標系 13

2.6　黄道座標系から赤道座標系への変換 14

3　WIS-C による計算シミュレーション

3.1　C 言語でのプログラミング 17

3.2　ディジタル・クロック 17

3.3　軌道計算プログラム 18

3.4　位置制御原理 19

3.5　追従誤差 20

3.5.1　計算誤差 20

3.5.2　移動誤差 22

3.6　まとめ 23

4　追従実験
4.1　実験方法 23
4.2　実験結果 23
4.3　考察 25
5.　参考文献 25
6.　謝辞 25
付録 26

1. 序章

1.1 背景

近年、地球温暖化や大気汚染の問題がある。これら大気環境の状態を知る指標の1つとしてエアロゾル濃度の測定がある。エアロゾルとは、黄砂や工場の排煙に含まれる微粒子のことを指し、大気中の濃度によって地球の温暖化に影響を及ぼすことが知られている。そのため、エアロゾルの濃度測定を行うことは今後の環境変化を知る上で重要となる。

1.2 エアロゾル濃度の測定原理

エアロゾル濃度の測定原理の概要を図 1-1 に示す。エアロゾル濃度を測定するために太陽を光源とする。太陽から放射された光は大気分子やエアロゾルの吸収と散乱によって地上で観測される放射照度が変化する。大気分子による影響で、地上で観測される太陽放射照度の変化は季節や緯度によって多少変化するが、時間的、空間的にほぼ一定として扱うことができる。そのため、太陽の放射照度を計測し、連続的に観測することによって、エアロゾル濃度の増減を測定することが可能である。

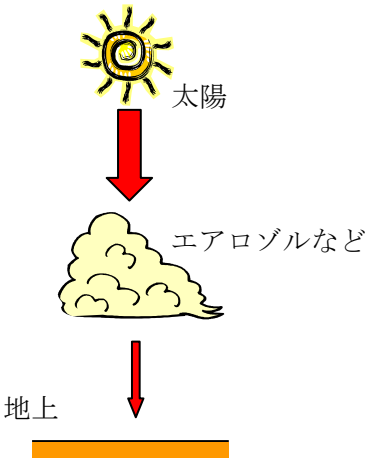

図 1-1 エアロゾル濃度の測定原理

1.3 サンフォトメーター

サンフォトメーターはエアロゾルなどで減光された太陽光を測定するための装置である。しかし、太陽は日周運動によって常に移動するためサンフォトメーターもその動きに合わせて測定しなければならない。そのため、サンフォトメーターに

は太陽を追従するサントラッカーと、太陽の光強度を測定する部分とに構成されている(図 1-2 右)。図 1-2(左)に示すサンフォトメーターはサントラッカーと測定部分が 1 つになっていて、太陽を自動追従するために 4 分割フォトダイオードを用い、駆動機としてステッピングモータを用いている。

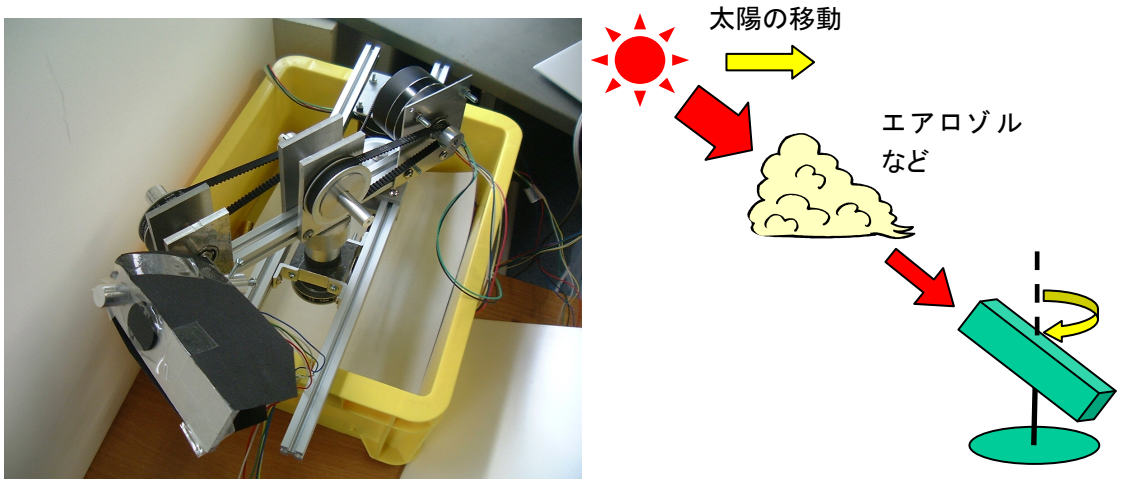

図 1-2　サンフォトメーター

1.4　目的

昨年の卒業研究ではサンフォトメーターを製作することによって、太陽を自動追従しながらエアロゾル濃度の測定が可能になった。しかし、太陽が雲に隠れると追従ができなくなり手動で位置を修正する必要があった。そこで、本研究では太陽が雲に隠れても追従できる手法とアルゴリズムの製作を行う。

2　軌道計算を用いた太陽の追従

2.1　システム概要

太陽の軌道計算を用いたシステム概要を図 2-1 に、軌道計算のフローチャートを図 2-2 に示す。太陽が雲などに隠れていないときは 4 分割フォトダイオードを用いて追従する。フォトダイオードは太陽光によって電流が発生する。この電流を変換回路で電圧に変換し、PIC でデジタル値に直す。この値で対向するフォトダイオードを比較して太陽と正対するように追従する。また、太陽が隠れている時は軌道計算を用いて太陽の位置を予測して追従する。このとき、PIC はクロックと割り込みを用で 1 秒周期を生成し、この 1 秒周期で 60 秒の時間をつくり時間、分、秒の変

数値を更新する。軌道計算ではこの値を用いて軌道計算を行い、計算から求めた値を目標位置としてサンフォトメーターに取り付けられているポテンショメータからの位置をもとに必要な信号をステッピングモータに送り太陽の位置まで移動する。

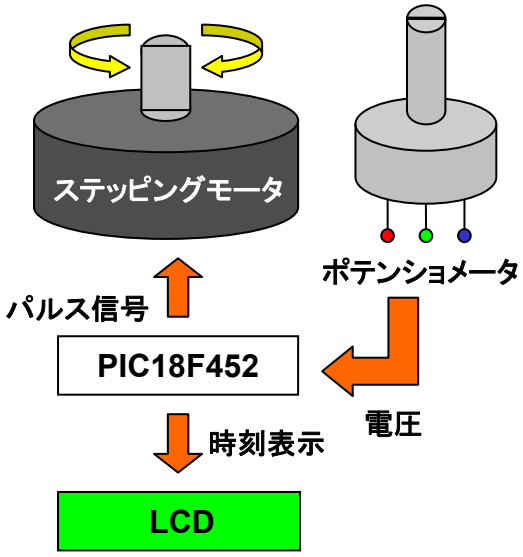

図 2-1　システム概要

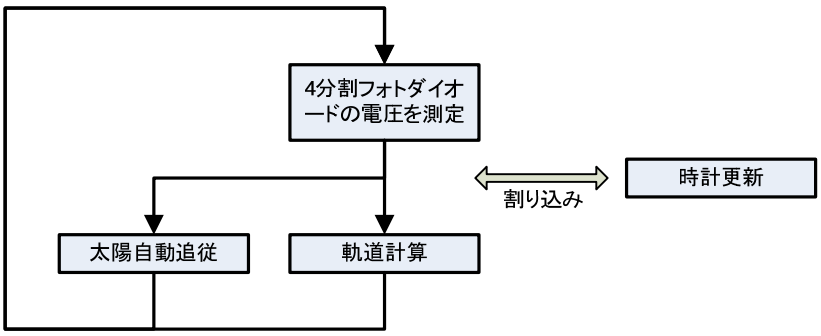

図 2-2　フローチャート

2.2　軌道計算に用いる座標系について

天体の位置を表すには、あらかじめ適当な座標系を決め、それにしたがって位置を示す。天体は空間に位置しているため、３次元直交座標系を使ってその位置を示すことができる。しかし、天体の位置は方向だけを取り扱い、距離を問題にすることは少ない。そのため、長さを扱う直交座標系は不便である。また、天体の位置をすべて球面上に投影することによって球面の内側から見たように考えることがで

きるため天体の位置がつかみやすい。そこで、天球座標系を用いて太陽の位置計算をおこなう。天球座標系では地球表面の測地系（経緯度）と同様の座標格子を用いるが、座標格子の天球への投影方法によって異なる座標系が存在する。それぞれの座標系は基準面によって決まり、基準面のとり方によって名前が付けられている。以下に座標系の基準面と極、経緯度を示す。

・ 地平座標系　地平線 － 天頂/天底 － 方位角(A)と高度角(h) （図 2-3）

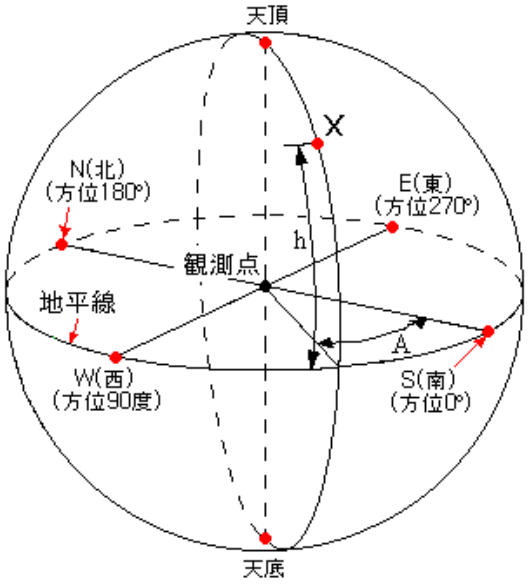

図 2-3　地平座標系

・ 赤道座標系　天の赤道 － 天の北極/天の南極 － 赤経(α)と赤緯(δ) （図 2-4）

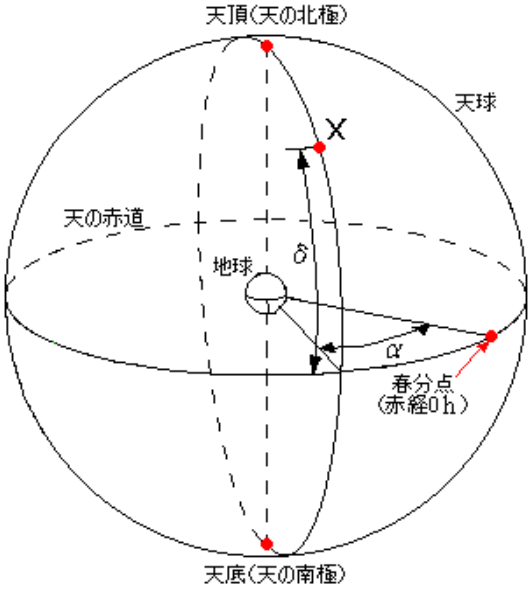

図 2-4　赤道座標系

・ 黄道座標系　黄道 － 黄道北極/黄道南極 － 黄経(λ)と黄緯(β) （図 2-5)

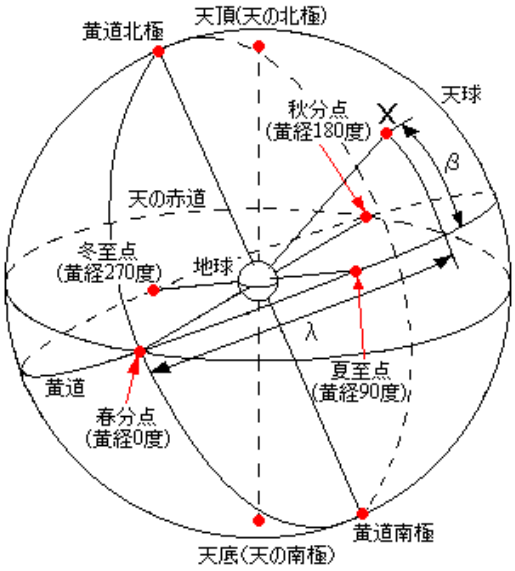

図 2-5　黄道座標系

2.3　太陽の黄経・黄緯の算出方法

太陽、月、惑星などの太陽系内の天体の位置を計算するときは黄道座標系を用いるほうが便利である。そのため、黄道座標系を用いて太陽の位置を計算する。太陽の黄経度の求め方には以下のいくつかの方法がある。

(1)地球の軌道要素を用いて、日心黄道直交座標における地球の黄経を求める方法

(2)天体暦から求める方法

(3)S.Newcomb の理論から求める方法

(1)は位置計算として一般的な方法である。しかし、観測する天体にもよるが位置計算の誤差が大きいため概略位置を計算する目的に限定される。(2)は、天体暦という太陽、月などの運行位置や日食、月食、天体の出没といった天象を推算して予報する暦を用いて求める。そして、(3)は S.Newcomb の理論を応用した略算式を用いた計算方法である。ここでは(1)と(3)について説明する。

2.4　地球の日心黄経から求める方法

2.4.1　ケプラーの方程式を用いたの求め方

2 体問題として運動している天体が、与えられた時刻に、楕円軌道のどこに位

置しているかを計算する方法について考える。ここで扱う楕円軌道は、太陽の周りを運動する惑星、小惑星、彗星などの軌道のほかに、地球の周りを回る人工衛星の軌道を計算することができる。今回は、地球からの太陽の位置を求めるために太陽のまわりの地球の運動を例として説明をする。図 2-6 に楕円軌道と xy 座標の関係図を示す。楕円軌道の長半径 a と離心率 e はあらかじめわかっているものとする。

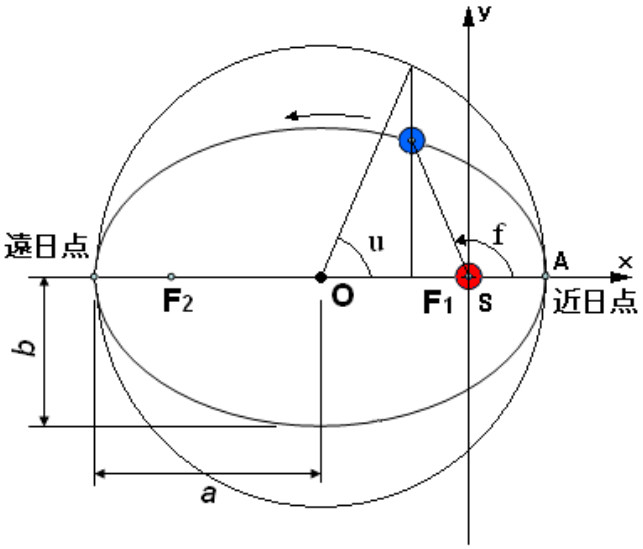

図 2-6　楕円軌道

その軌道の存在する平面を xy 直交座標面にして、太陽 S を原点にとる。そして、地球と太陽との近日点 A の方向を x 軸正の向きにとる。地球の座標(x,y)を求めるのにケプラーの方程式から離心近点角 u を導出する必要がある。ケプラーの方程式は下式で表すことが出来る。

（平均近点角：l、離心近点角：u、離心率：e）

$$u - e\sin u = l \quad (2.1)$$

ケプラーの方程式では u について解いた形に書き直すことはできない。そのため、必要な精度まで解くしかない。ケプラーの方程式から求めた値 u_n の値を必要な精度で正確な値にするには次式により求めることができる。

$$\begin{cases} \Delta u_n = \dfrac{l - u_n + e\sin u_n}{1 - e\cos u_n} \\ u_{n+1} = u_n + \Delta u_n \end{cases} \quad (2.2)$$

この計算を繰り返して、近似値を高めながら u_1、u_2…と値を求めていくうちに必

要精度の範囲で Δu_i が 0 となる。このときの u_i が解となる。求めた離心近点角 u と惑星の xy 座標の関係は次のようになる。

$$\begin{aligned} x &= a(\cos u - e) \\ y &= a\sqrt{1-e^2}\sin u \end{aligned} \quad (2.3)$$

2.4.2　日心直交座標系で表した天体の位置

天体の楕円軌道上の位置だけではその天体がどこに見えているかを知ることはできない。そのため、天体の軌道上の位置を、太陽を原点とする日心直交座標系で表す必要がある。

2.4.2.1 日心直交座標系

計算に使われる日心直交座標系には 2 種類ある。日心黄道直交座標系と、日心赤道直交座標系である。これらは黄道直交座標系、赤道直交座標系をそれぞれ平行移動して、原点を太陽の中心に移したものである。楕円軌道上で計算した天体の位置（x,y）を、日心黄道直交座標系による位置（X,Y,Z）に変換するにはこの 2 つの座標系の関係を知る必要がある。

2.4.2.2　軌道要素

図 2-7 に日心黄道直交座標系を示す。楕円軌道も日心黄道直交座標系も太陽の重心を原点としている。楕円軌道を含む平面（xy 面）は黄道面（XY 面）と一致しない限り黄道面と原点を通る直線 NSN'でまじわる。

軌道要素とは天体それぞれが持つ楕円軌道の形や日心直交座標系での位置などを表す値である。下図の∠γSN を昇交点黄経といい、Ω（オメガ）で表す。そして、軌道平面と黄道面の交わりの角を軌道傾斜角といい i で表す。この i は黄道上で黄道の増加する向きと、軌道上で運動する天体の向きにはさまれる角を示している。そのため、角度の範囲は 0≦i≦180 となる。昇交点黄経Ωと軌道傾斜角 i を求めることによって、楕円軌道面の位置が日心黄道直交座標系のどこに位置しているのかがわかるようになった。しかし、これだけでは楕円軌道の向きが決まらない。そこで、楕円軌道の x 軸を決めている近日点 A と昇交点のなす角∠NSA を用いて向きを決める。この角を近日点引数といい、ω（オメガ）で表す。こうして、Ω,i,ωの 3 つの値を決めることによって、楕円軌道を

表した xy 座標系と、日心黄道直交座標系の X,Y, Z との関係がわかる。

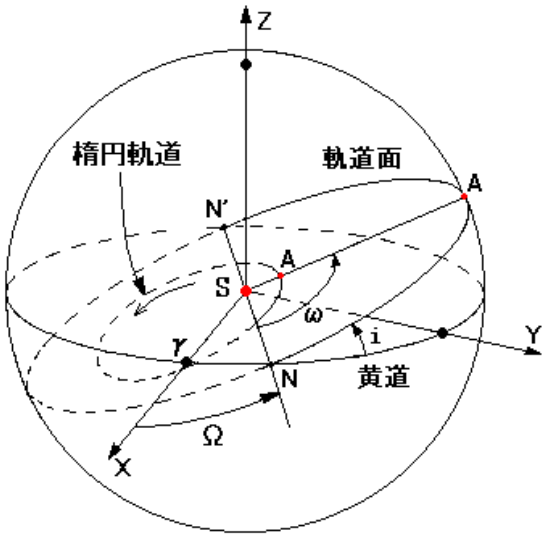

図 2-7　日心黄道直交座標系

2.4.2.3　xy 座標から日心直交座標への変換

xy 座標から日心直交座標への変換には、以下の 3 回の座標系の回転によって日心黄道直交座標系の位置（X,Y,Z）に変換できる。

（1）楕円軌道の z 軸を軸として、正の向きから見て時計回りにωの回転

（2） x 軸を軸として、正の向きから見て時計回りに i の回転。

（3） z 軸を軸として、正の向きから見て時計回りにΩの回転。

上記から得られる式は下式となる。

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{pmatrix} \begin{pmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \quad (2.4)$$

これによって、太陽から見た地球の位置座標がわかる。これを地球から見た太陽の黄経λs、黄緯βs を求める式は以下となる。

$$\lambda_S = \tan^{-1}\frac{Y}{X} \pm 180$$
$$\beta_S = -\tan^{-1}\frac{Z}{\sqrt{X^2+Y^2}} \quad (2.5)$$

また、視黄緯は 1”(=1/3600°)以下の量なので高精度の必要がなければβs=0 としてもよい。

2.4.3　惑星光行差による天体位置のずれ

実際の天体の位置はこれまで求めた値より位置がずれている。これは、惑星光行差による天体の位置のずれが生じているためである。惑星光行差とは、観測地点から天体までの距離があまりにも遠いとき、光が観測者に到達する時間のうちに天体が移動するため観測時刻に実際に天体が存在する位置と、観測地点から見てその時刻に天体が見える位置にずれが生じる現象のことである。

ある時刻 t に地球から太陽を観測したとする。地球と太陽の間の距離を L とすると、光速 c で距離 L を進んだときにかかった時間 τ は τ =L/c である。したがって、地球で時刻 t に見た太陽は、実際は t-τ の太陽の姿である。そのため、時刻 t に地球で見える太陽の位置を計算するには、時刻 t-τ に対する位置を求める必要がある。

2.5　S.Newcomb の理論から求める方法

2.5.1　太陽位置の略算式

太陽の黄経・黄緯を簡単に説明する。太陽は黄道座標系では、黄道上を移動している。そのため、黄緯は常に 0 度となる。そして、太陽の黄経を計算には略算式を用いる。略算式は Newcomb の理論を応用した推算式を海上保安庁水路部（現在は海洋情報部）が開発したものである。これにより、太陽の黄経の計算を簡単にすることができる。

黄経の計算の手順を説明する。観測時間が年、月、日、時、分、秒のとき、まず、観測日の 0 時までの経過日数 Z を計算する。そのために、以下の式に観測時間を代入する。

$$
\begin{aligned}
W &= (\text{年} - 1900)/4 \\
W &= [W] + F \\
Y &= [1461W] \\
X &= [(\text{月} + 7)/10] \\
R &= [1 - F] \\
S &= [0.44(\text{月} + 4.4)] \\
Z &= Y + 31 \times \text{月} + \text{日} + (X - 1)R - XS - 27424
\end{aligned}
\qquad (2.6)
$$

ここで、 [W] は W の整数部、F は小数部を表す。 []はガウス記号である値につ

いて、それを超えない最大の整数値を表す際に用いられる。日の端数を J で表わし

$$J = 時/24 + 分/1440 + 秒/86400 \quad (2.7)$$

とすると、世界時で表現した経過時間 t は下式となる。

$$t = (Z + J)/365.25 \quad (2.8)$$

さらに、暦表時による表現 T に換算すると

$$T = t + (0.0317t + 1.43) \times 10^{-6} \quad (2.9)$$

となり、この T を下式に代入して計算すると黄経が求まる。

$$\begin{aligned}
\lambda s' = {} & 279.0358^\circ + 360.00769^\circ T \\
& + (1.9159 - 0.00005\ T)\sin(356.531^\circ + 359.991^\circ T) \\
& + 0.0200\ \sin(353.\ 06 + 719.981T) \\
& - 0.0048\ \sin(248.64 - 19.341\ T) \\
& + 0.0020\ \sin(258.0 + 329.64T) \\
& + 0.0018\ \sin(334.2 - 4452.67\beta T) \\
& \quad \cdot \\
& \quad \cdot \\
& + 0.0004\ \sin(233.8 + 299.30\ T) \\
& - 0.0004\ \sin(198.1 + 720.02\ T) \\
& + 0.0003\ \sin(349.\ 6 + 1079.97\ T) \\
& + 0.0003\ \sin(241.2 - 44.43\ T)
\end{aligned} \quad (2.10)$$

この視黄経λs'は光行差を考慮した、その時刻に太陽重心の見える位置であり、真に太陽重心の位置する黄経λs'(幾何学的黄経)は下式となる。

$$\lambda_S = \lambda_S' + 0.0057^\circ \quad (2.11)$$

今回の実験では、精度やプログラムの組みやすさから略算式を用いて太陽の黄経、黄緯を計算した。

2.5.2　黄道座標系

太陽など太陽系内の天体の運動を表すときは、黄道座標系を用いると便利である。その理由として、黄緯の変化幅が小さいことである。黄道座標系は太陽の見かけの通り道である黄道(地球の公転面)を基準としている。そのため、太陽の黄緯は常に 0° となり惑星の緯度の変動も小さくなる。

黄道座標系を図 2-8 に示す。黄道座標では、天球上の緯度と経度にあたるものとして黄緯:β と黄経:λ を使用する。黄緯は地球の公転面の天球上への投影である黄道を 0 度、地球の公転面に垂直な方向を 90 度、−90 度として表す。黄経は春分点を 0 度として、太陽の黄道上の見かけの運動方向と同じ方向に向かって値を増やして春分点に戻る 360 度まで数える。つまり、夏至点は黄経 90 度、秋分点は黄経 180 度、冬至点は黄経 270 度となる。

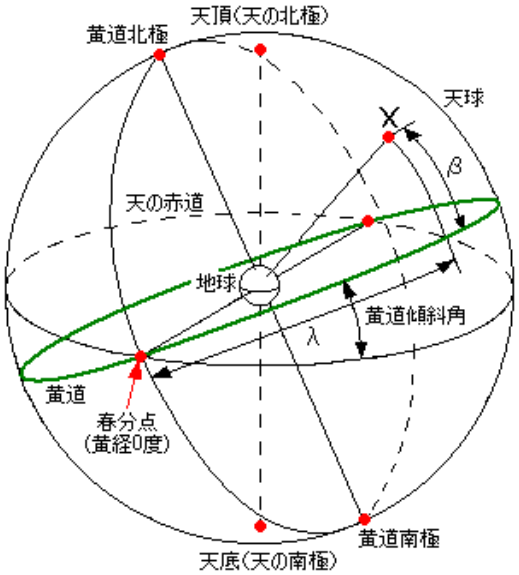

図 2-8　黄道座標系

2.6　黄道座標系から赤道座標系への変換

黄道座標系で太陽の位置を表した。しかし、黄道座標系では地球から見た位置を求めることができない。そこで、赤道座標系を用いることによって地球からの太陽の位置を求める。

黄道座標系では黄経と黄緯、赤道座標系では赤経と赤緯とそれぞれの座標系で天体の位置の表し方が違う。そのため、これらの座標系の天体の位置を X,Y,Z で表すことによってそれぞれの座標系に置換することができる。赤道直交座標系は X 軸の正の向きを春分点春分点の向きに、Y 軸の正の向きを赤経 60h（=90 度）,赤緯 0 度の向きに、Z 軸の正の向きを天の北極の向きにとった座標のことである。このときの、赤経 α と赤緯 δ で示される天体の方向余弦（X_{Eq}, Y_{Eq}, Z_{Eq}）は（2.12）となる。また、黄道直交座標系も赤道直交座標系と同様な方法で方向余弦を求めることがで

きる。黄道座標系での方向余弦は（2.13）となる。

$$\begin{cases} X_{Eq} = \cos A = \cos\delta\cos\alpha \\ Y_{Eq} = \cos B = \cos\delta\sin\alpha \\ Z_{Eq} = \cos C = \sin\delta \end{cases} \quad (2.12)$$

$$\begin{cases} X_{Ec} = \cos\beta\cos\lambda \\ Y_{Ec} = \cos\beta\sin\lambda \\ Z_{Ec} = \sin\beta \end{cases} \quad (2.13)$$

黄道座標系で表しているある天体の位置（X_{Ec}, Y_{Ec}, Z_{Ec}）を赤道座標系（X_{Eq}, Y_{Eq}, Z_{Eq}）で表すと以下の式となる。

$$\begin{cases} X_{Eq} = X_{Ec} \\ Y_{Eq} = Y_{Ec}\cos\varepsilon - Z_{Ec}\sin\varepsilon \\ Z_{Eq} = Y_{Ec}\sin\varepsilon + Z_{Ec}\cos\varepsilon \end{cases} \quad (2.14)$$

このとき ε は平均春分点における黄道と天の赤道の交わる角度である。

2.7　地表座標系

天体の赤経、赤緯がわかっても、地球の重心から見た太陽の位置をその天体が空のどこに見えるかわからない。実際に天体がどこに見えるかを示すには地平座標系を用いる必要がある。地平座標とは、地平線と方位を基準として天体の位置を表すために使用する座標系のことである。地平座標では方位角と高度を使用して天体の位置を表す。高度は地平線上を 0 度として天頂を 90 度として表し、地平線下にある天体についても高度をマイナスとして表すことができる。 方位角は真南を 0 度として真西を 90 度､真北を 180 度､真東を 270 度とするように 360 度まではかる。天体の高度や方位角は同じ時刻でも観測する地点の緯度や経度に依存して変化し、また同じ地点でも天体の日周運動や年周運動,観測時刻によって変化する。 そのため、地平座標で天体の位置を表す場合には観測地点の緯度、経度、観測の時刻を指定する必要がある。地平座標系への変換には下式を用いる。

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \sin\Phi & 0 & -\cos\Phi \\ 0 & 1 & 0 \\ \cos\Phi & 0 & \sin\Phi \end{pmatrix} \begin{pmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} L \\ M \\ N \end{pmatrix} \quad (2.15)$$

ここで、Φは観測点の緯度、Θは観測点のグリニジ平均恒星時のことである。この

計算から地平座標系に関する方向余弦（l,m,n）が求まる。この値を用いて観測点から見える太陽の位置は A,h から求めることができる。

$$A = \tan^{-1}\left(-\frac{m}{l}\right) \quad (2.16)$$
$$h = \sin^{-1} n$$

3　IZ-C による計算シミュレーション

3.1　C 言語でのプログラミング

PIC はアセンブリ言語でプログラムされることが多い。今回、PIC18F452A でディジタル・クロックと軌道計算プログラムを作成する。そのため、少し複雑なプログラムを製作しなければならないので、C 言語によってプログラムを作成することによって、複雑な動作も簡単にプログラミングができるようにした。今回使用する WIZ-C は、PIC 特有のバンクの切り替えなどを意識することなく、プログラミングを行うことができる。

WIZ-C による開発手順を簡単に説明する。まず、プロジェクトを新規作成し、アプリケーションデザイナと呼ばれる画面で、WIS-C 内蔵ツールで I/O ポートや PIC モジュールと呼ばれる LCD（液晶表示器）などのパーツやピンの設定を視覚的に定義するツールの設定を行う。設定が終わりコード生成すると、xxx_Main.c と xxx_User.c という 2 つのファイルが自動生成される（xxx はプロジェクト名）。xxx_Main.c にはアプリケーションデザイナで定義した I/O やデバイスの初期化処理が自動で生成され、main 関数が生成される。ユーザーが実際にプログラムを組み込むのは、xxx_User.c の UserLoop、UserInterrupt、UserInitialize 関数である。UserInterrupt 関数は I/O や変数などの初期化処理、UserInitialize 関数は割り込み処理、UserLoop 関数には目的のプログラムを書き込む。UserLoop 関数は main のループの中から呼び出され処理される。一通り完成したら WIZ-C からコンパイル操作を行う。このとき、コンパイルエラーがある場合は、情報ウィンドウにエラーのある行番号とエラーの内容が表示されるのでそれらを見てエラーのあるところを修正する。コンパイルエラーがなくなったら、デバッガウィンドウの PIC に LCD、抵抗などを接続し、パソコン上で作成したプログラムの動作をシミュレーションすることができる。シミュレーションの結果がうまくいった場合、コンパイル時に生成されている HEX ファイルを PIC に書き込む。不具合があったときは、プログラムを修正し、目的の動作を行うプログラムを作成する。

3.2　ディジタル・クロック

ディジタル・クロックを製作する上で重要なのは、時を刻むのに必要な正確な 1 秒周期である。そこで、外部からのクロックパルスを TIMER1 と CCP1（コンペ

アモード）を組み合わせて時計用の 1 秒を生成する。図 3-1 に 40MHz のクロックから 1 秒周期を作成する仕組みを示す。PIC の動作クロックを 40MHz としたとき、TIMER1 で内部クロックを使用するように設定すると、周波数は初段で 1/4 になる。プリスケーラは最大の 8（1/8 分周）に設定する。コンペア・レジスタ CCPR1 に 49999 を設定して TIMER1 を 50000 カウントごとにクリアするようにすると、1/49999 のプリスケーラとなる。このプリスケーラを含めた最終的な周波数は以下のとおりとなる。

$$40MHz \times \frac{1}{4} \times \frac{1}{8} \times \frac{1}{50000} = 25Hz$$

この値は 1 秒間に 25 回割り込みが発生することを意味している。つまり、CCP1 割り込みを 25 回数えることで正確な 1 秒を得ることができる。

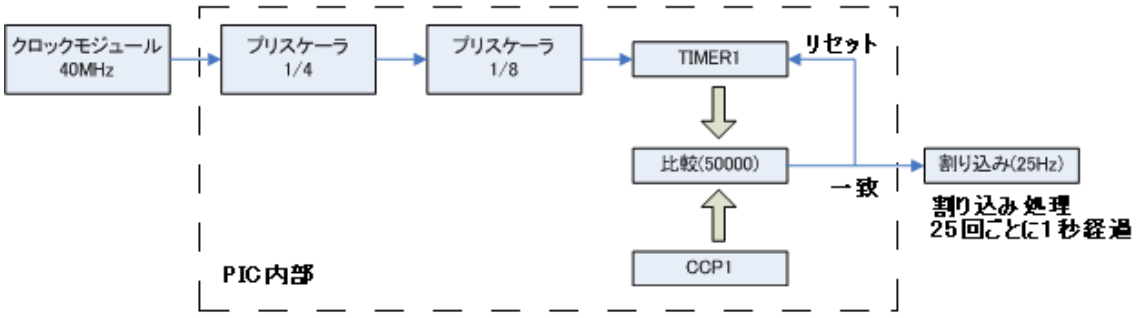

図 3-1 40MHz のクロックから 1 秒周期を作成する仕組み

3.3　軌道計算プログラム

軌道計算には三角関数や行列などの計算が多くあり、10MHz のクロックでは 30 秒以上も計算にかかってしまう。そのため、クロックを 40MHz にすることによって計算時間を 1/4 にした。クロックを 40MHz にするにはコンフィギュレーションワードの設定でクロックタイプを HS PLL 使用するに設定する。これによって、使用しているクロック数を 4 倍にすることができる。

三角関数の計算プログラムは xxx_MATHS.c に書かれている。したがって、黄経などの計算で三角関数を使用するときは、float sin（float x）;と関数を使用するファイル上でプロトタイプ宣言さえすれば簡単に計算できる。しかし、$\sin^{-1}(x)$や$\tan^{-1}(x)$関数はライブラリにない。そのため、これらの関数を作成しなければならない。WIS-C では、三角関数の計算方法はマクローリン展開を用いて近似値を求

めて三角関数の計算を求めている。そのため、$\sin^{-1}(x)$や$\tan^{-1}(x)$もマクローリン展開を用いてプログラムを作成した。$\sin^{-1}(x)$と $\tan^{-1}(x)$のマクローリン展開では$|x| \leqq 1$ であることが条件となる。計算の過程で $\sin^{-1}(x)$は x の値が範囲内に収まるが、$\tan^{-1}(x)$の場合 x が範囲外の値になるときがある。そのため、$\tan^{-1}(x)$の計算をマクローリン展開できるようにするために以下の式で計算をした。

$$\tan^{-1}(x) = \sin^{-1}\frac{x}{\sqrt{1+x^2}}$$

3.4　位置制御原理

ある時刻における観測地点からの太陽の高度角と方位角を計算から求めたとき、次はサンフォトメーターをその位置まで駆動させなければならない。しかし、サンフォトメーターの現在の位置を知らなければ目的の位置がわからない。位置を知る方法として地磁気センサーから目的の方位を見つけ出す方法と、ポテンショメータの出力電圧から位置を求める方法がある。今回はポテンショメータを用いて位置を検出する方法にした。

使用したポテンショメータは RDC50A003 で、可動回転角度は 320 度、全抵抗値は 10kΩである。ポテンショメータには回転角度を検出するロータリーポテンショメータと、直線上の位置を検出するリニアポテンショメータがある。今回使うのはロータリーポテンショメータで、回転角度に対して抵抗値の変化がリニアになっている。そのため、ポテンショメータからの出力電圧は角度に対して変化する。このポテンショメータから出力される電圧を A/D 変換して、PIC で計算した方位角、高度角の位置と比較して目的位置まで移動する。また、しきい値は 0.12 度にした。サンフォトメーターの正面を 0 度とすると、可動範囲は上下左右それぞれ 160 度となる。このとき、南を 0 度、東-90 度、西は 90 度とする。

センサーからの値が目標値と同じとき、目標位置に位置していると判定できる。センサーからの値が目標値より大きいとき、目標位置よりも右または上に位置していることがわかる。逆に小さいときは、目標値より左または下に位置していることがわかる。このような位置制御を行うプログラムのフローチャートを図 3-3 に示す。

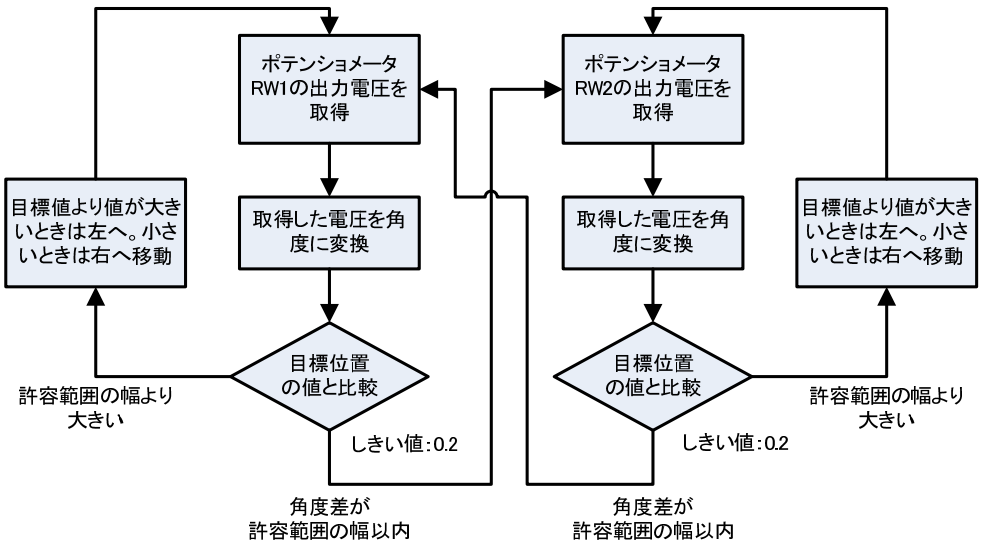

図 3-3　位置制御フローチャート

3.5　追従誤差

3.5.1 計算誤差

Excel と PIC プログラムでの方位角の計算結果を図 3-4 に示す。図での 4 回、12 回は、三角関数のマクローリン展開の項数を表している。項数を 4 回にしたときでは、Excel の結果に一致していなかった。12 回では 4 回のときより方位角の誤差がかなり抑えられている。しかし、2 時から 6 時、18 時から 20 時ぐらいの間での方位角の値がずれている。その理由としてマクローリン展開を用いた近似の項数が少ないことが挙げられる。

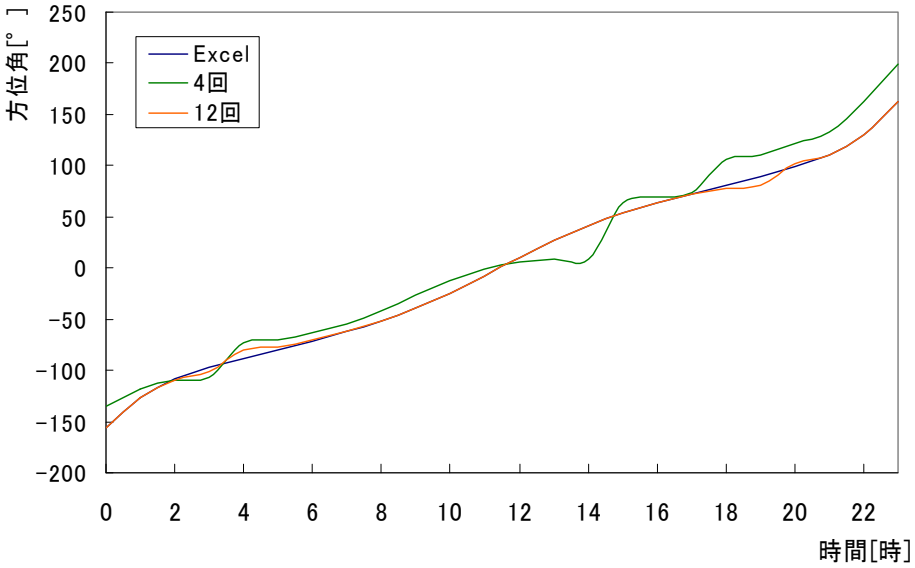

図 3-4　Excel と PIC プログラムの方位角

$\sin^{-1}(x)$のグラフの形は、横軸の値が 0 付近では傾きは緩やかであるが、±0.7 の付近では傾きが急になっている。このため、マクローリン展開の項数が少ないと±1 付近の値に誤差が生じる。このときに生じる誤差は 0.5 程度であるが、太陽の位置が 0.5 度ずれることは時間にして 2 分の誤差がでることになる。このため、$\sin^{-1}(x)$でのマクローリン展開の項数を増やした。$\sin^{-1}(x)$の計算で、項数を 3000 項程度にすれば急な曲線の部分でも誤差が小さくなり全体の誤差も少なくなる（図 3-5）。

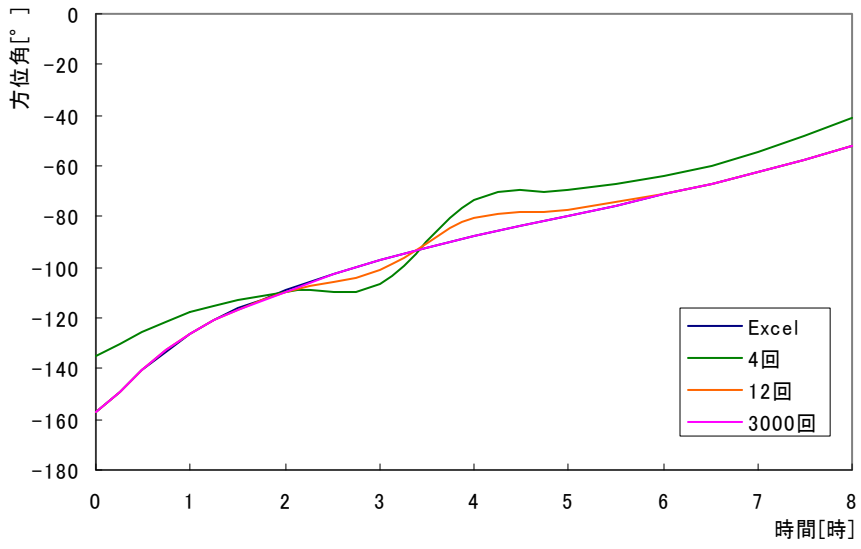

図 3-5　Excel と PIC プログラムの方位角の比較

しかし、計算に使用している変数は float 型（最大値　10^{38}、最小値　-10^{38}）のため最大値、最小値を超える値を変数に入れるとオーバーフローとなり予期しない値が代入されることになってしまう。その対策として変数がオーバーフローする前にループ関数から抜け出すようなプログラムを作成する必要がある。

3.5.2 移動誤差

ステッピングモータの駆動角度は 0.2 度である。そのため、この角度以下の精度を出すことができない。また、ポテンショメータは理想では最小抵抗値は 0、最大抵抗値は 10kΩだが、実際は端子や抵抗の接触点の抵抗などでそのような値をとらない。このような要因から位置制御で誤差が生じることがある。

ポテンショメータの抵抗値が均一に分布しているとしてステッピングモータが 0.2 度動いたときに変化する抵抗値は 6.25Ω であり、電圧にすると 3.125mV となる。また、A/D 変換の最小分解能は 4.88mV である。このため、0.2 度動いたときの変化電圧は A/D 変換の最小分解能より値が小さいため A/D 値に変化はない。そのため、ステッピングモータは動いていないと判断されステッピングモータに信号を出し駆動する。0.2 度動き、つぎは 0.4 度となるこのときの電圧は 6.25mV で A/D 値も 1 増える。ここで PIC ステッピングモータは 0.2 度駆動したと判断する。つまり、この時点で 0.2 度の誤差が発生する。理想ポテンショメータを使用した時の目標値に対するサンフォトメーターの移動角を図 3-6 に示す。サンフォトメーターの移動角は 0.2 度ずつ変化するに対して、電圧の変化が A/D の分解能より小さいため、角度が変化してもその変化を検知できないときがあり階段状に変化する。また、目標位置が正符号方向より負符号方向のほうが誤差は少ない。誤差は目標位置によって異なるが 0.0 度から 0.4 度の範囲で、実際はポテンショメータの抵抗値変化によって基準位置や電圧も変わるので誤差の大きさも変わると思われる。

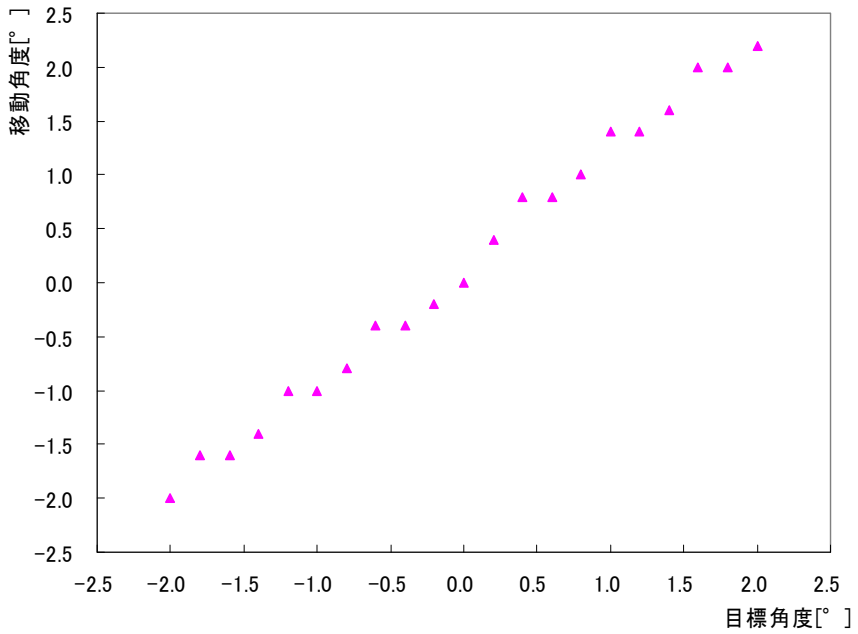

図 3-6　初期位置 0 度のときの A/D 変換による移動角度

3.6　まとめ

ディジタル・クロックと太陽の軌道を計算する関数をプログラムし、任意の時間での太陽の位置を計算しその位置に移動することを確認した。

4　太陽の追従実験

4.1　実験方法

太陽の追従実験は、電子機器実験室の窓際で行った。サンフォトメーターの正面を南に向くように設置し、太陽の方向に向けて追従の様子を記録した。また、サンフォトメーターの方位角と高度角は PIC から出力された AD 値から角度を算出し、その値を角度に変換して記録した。軌道計算による追従は 1 分おきに行うように設定した。

4.2　実験結果

2008 年 1 月 24 日の太陽の位置をサンフォトメーターで追従したときの方位角と高度角の結果と理論値を図 4-1 に示す。この実験ではしきい値を 0.3° に設定した。サンフォトメーターは 30 分ごとに軌道計算を行い、太陽を追従していることがわかる。方位角の平均誤差は 0.192° 、高度角の平均誤差は 0.172° であった。この

誤差を時間で表すと方位角は 46 秒で高度角は 41 秒のずれがあった。また、この測定した各時間帯の誤差を図 4-1 に示す。この図から三角関数の近似計算による影響はほとんどなく、設定したしきい値内に誤差がおさまっているのがわかる。

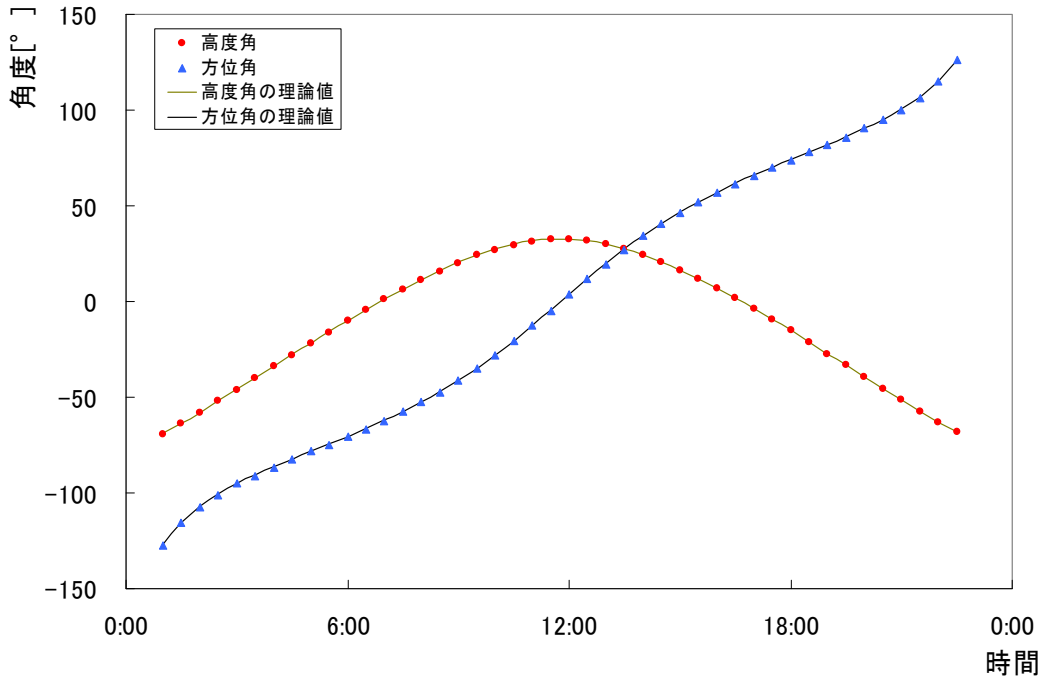

図 4-1　理論値と測定値

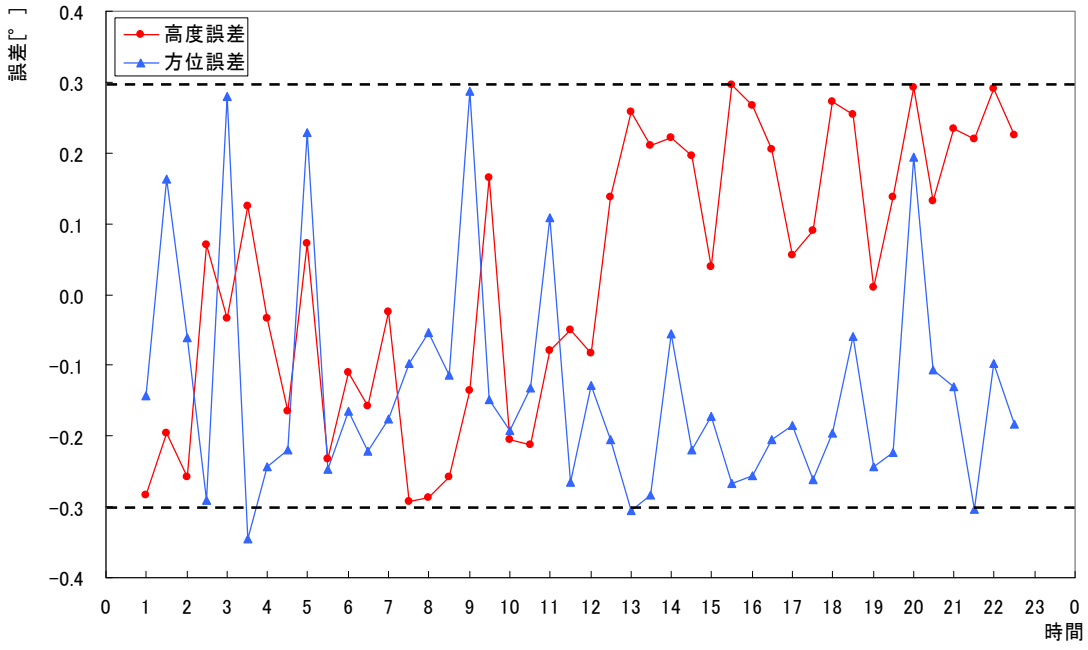

図4-2 各しきい値での誤差

4.3　考察

今回の実験では、サンフォトメーターが太陽の軌道計算と位置制御をして太陽追従ができることがわかった。測定誤差も時間にして 1 分以内におさまりよい結果となった。しかし、4 分割フォトダイオードを用いた太陽追従では、太陽が移動することによって観測地点と太陽の間の大気の厚さが変動し、太陽光強度が変化する。これにより、双方のフォトダイオードの電圧差も変化し、しきい値を下回る可能性が出るなど追従性能に支障をきたす。そのため今後の課題として、正確な位置制御ができる手法とその制御、しきい値調整の自動化アルゴリズムの作成が挙げられる。

5　参考文献

1)中尾　司：C 言語で作る PIC 電子工作，CQ 出版

2)長沢　工：天体の位置計算，地人書館

6　謝辞

本研究を行うにあたり、富山商船高等専門学校の由井四海教官をはじめ。多くの皆様にご指導および協力をいただきました。心よりお礼も押し上げます。

付録

ディジタルクロックと軌道計算、モータの駆動プログラムを以下に示す。正確な１秒周期を生成して時間変数を更新し、設定した間隔で軌道計算と位置制御を行うようにプログラムを作成した。

■ ディジタルクロック、自動追従および位置制御プログラム

```
#include " xxx_Auto.h のある場所が書いてある"

#include "Def.h"
#include "Global.h"
#include "ad_convert.h"
#include "m1drive.h"

#include <strings.h>
#include <maths.c>
#include <displays.h>

#define TH 1.0
#define Vmin 0.2
#define trigger 5
#define deg 0.35

#__config _CONFIG1H,0x24
#__config _CONFIG2L,0x0A
#__config _CONFIG2H,0x00
#__config _CONFIG3H,0x01
#__config _CONFIG4L,0x81
#__config _CONFIG5L,0x0F
#__config _CONFIG5H,0xC0

void TskClockUpdate(void);
void sun(float Year,float Month,float Day,float Hour,float Min, float Sec);
void tuijuu(void);
int Compare(WORD val1,WORD val2);
float asin(float x);

BYTE Sec;
BYTE Min;
BYTE Hour;
BYTE Day;
BYTE Mon;
WORD Year;
```

```
WORD Count;
BYTE Motor;
BYTE Secc;
BOOL TF1Sec;

void UserInterrupt()
{
	// Insert your code here

	if(TST_CCP1IF) {
		CLR_CCP1IF;
		Secc++;

		if(Secc == 25) {
			Secc = 0;
			TF1Sec = true;
			Count++;
			TskClockUpdate();
		}
	}
 #asmline goto UserIntReturn	; PIC Assembler - go back to interrupt routine
}

// ユーザ初期化処理
void UserInitialise()
{
	Secc = 0;
	TF1Sec = false;
	LCDClear();
	LCDString("Hello");
	InitDatas();

	RB_PULLUP_ON;

//	Wait(500);
	if(EEDATA != 0x12) {
		// EEPROM のデータが無効の時
		WriteEEData(0, 0x12);
		LCDClear();
		LCDString("EEPROM CLR");
//		Wait(500);
		LCDClear();
		TrgClearAll();
	}
```

```
    Year = 2008;
    Mon = 1;
    Day = 13;
    Hour = 21;
    Min = 0;
    Sec = 0;

    Count = 0;
    Motor = 1;

    EI_PERIP;
    EI_CCP1;
}

void UserLoop()
{
 int ad1,ad2,ad3,ad4;
 static float Vave1,Vave2,a=0.004883;
 static float h;

 TskClearAll();
 TskSetProg();
 TskSelClr();

 if(Count==trigger)
 {
  Count=0;
  sun( Year, Mon, Day, Hour, Min, Sec);
 }
}

void tuijuu(void)
{
 int adval1,adval2,adval3,adval4,com,com2;

 adval1=ad_convert(0);
 adval2=ad_convert(1);

 com=Compare(adval1,adval2);

 if(com==2)
 {
  m1drive(5,1,1);
 }
```

```
  else if(com==1)
  {
    m1drive(5,1,0);
  }

  adval3=ad_convert(2);
  adval4=ad_convert(3);

  com2=Compare(adval3,adval4);
  if(com2==2)
  {
    m2drive(10,1,1);
  }
  else if(com2==1)
  {
    m2drive(10,1,0);
  }
}

int Compare(WORD val1,WORD val2)
{
  float a=0.004883,b1,b2;

  b1=a*val1;
  b2=a*val2;

  if(Secc<24 || Secc>0)
  {
    if(b1+TH<b2)
    {
      return(1);
    }
    else if(b1>b2+TH)
    {
      return(2);
    }
  }
  else
  {
  }
  return(0);
}

void KeyInput()
{
```

```
//      MenuProc();
}

void EEDone() {
      EEWDone = true;
}

void TskClockUpdate(void)
{
 char str_buf[16];
 int t;

      switch(Mon) {
       case 4;
       case 6;
       case 9;
       case 11;
         t=30;
         break;

       case 2;
         if(Year%4==0) {
               t=29;
         }
         else {
               t=28;
         }
         break;

       default;
         t=31;
      }

      DI_CCP1;
      if(TF1Sec) {

            TF1Sec = false;
            EI_CCP1;
            if((CurrMenuState > 24) || (CurrMenuState < 10)) {
                  Sec+=12;
                  if(Sec >= 60) {
                        Sec = 0;
                        Min+=30;
                        if(Min >= 60) {
                              Min = 0;
```

```
                    Hour++;
                    if(Hour >= 24) {
                        Hour = 0;
                        Day++;
                        if(Day >= t) {
                            Day = 1;
                            Mon++;
                            if(Mon >= 13) {
                                Mon = 1;
                                Year++;
                            }
                        }
                    }
                }
            }
        }

        if(CurrMenuState == 0) {
            // 通常の時計表示中のとき
            sprintf(str_buf, "%04d_%02d_%02d", Year, Mon, Day);
            LCDPrintAt(0, 0);
            DI_GLOBAL;
            LCDString(str_buf);
            sprintf(str_buf, "%02d:%02d:%02d", Hour, Min, Sec);
            LCDPrintAt(0, 1);
            LCDString(str_buf);
            EI_GLOBAL;
        }
    }
    EI_CCP1;
}

void Compare2(float C,float C2,int Pinn,int S)
{
 static float data,data2;
 static unsigned int D,F;
 char str_buf[16];

start:
 data=ad_convert(Pinn);
 data2=(data-512)*C2;
 D=fabs(C-data2)*100.0;

    F=data;
    DI_GLOBAL;
```

```
    sprintf(str_buf, "%04d_%02d_%02d_%4d", Year, Mon, Day, F);
    LCDPrintAt(0, 0);
    LCDString(str_buf);
    sprintf(str_buf, "%02d:%02d:%02d_%4d", Hour, Min, Sec,D);
    LCDPrintAt(0, 1);
    LCDString(str_buf);
    EI_GLOBAL;

if(Count>=trigger-1)
{
 return();
}
else
{
 if(Secc<24)
 {
  if(fabs(C-data2)<deg)
  {
   return();
  }
  else if(data2<C)
  {
   if(S==0)
   {
    m2drive(8,1,1);
    Wait(5);
   }
   else
   {
    m1drive(8,1,0);
    Wait(5);
   }
  }
  else if(data2>C)
  {
   if(S==0)
   {
    m2drive(8,1,0);
    Wait(5);
   }
   else
   {
    m1drive(8,1,1);
    Wait(5);
   }
```

```
    }
    goto start;
   }
   else
   {
    goto start;
   }
  }
}
```

□ AD 変換プログラム

```
#define ADC_START              ADCON0|=(1<<GO)
#define TST_ADC_NOT_DONE (ADCON0&(1<<GO))

void SelADCChannel(int ch);
WORD ReadADCValue(void);

WORD ad_convert(int port){
      WORD val;
      SelADCChannel(port);
      ADC_START;
      while(TST_ADC_NOT_DONE) {}
      val = ReadADCValue();

      return(val);
}

void SelADCChannel(int ch) {
      BYTE chn;
      BYTE reg;
      chn = (ch << 3) & 0x38;
      reg = ADCON0 & ~0x38;
      reg |= chn;
      ADCON0 = reg;
}

WORD ReadADCValue(void) {
      WORD adval;

      adval = (WORD)ADRESH << 8;
      adval = adval | ADRESL;
      return adval;
}
```

■ モータ駆動プログラム

```
void m1drive(int wtime,int step,int rot);
void m2drive(int wtime,int step,int rot);

void m1drive(int wtime,int step,int rot)
{
 static int position,state2;

 while(step>0)
 {
  switch(position)
  {
   case 12:
    if(rot==1)
    {
     state2 = 144;
     position = 9;
    }
    else
    {
     state2 = 96;
     position = 6;
    }
    PB = state2;
    Wait(wtime);
    step--;
    break;

   case 9:
    if(rot==1)
    {
     state2 = 48;
     position = 3;
    }
    else
    {
     state2 = 192;
     position = 12;
    }
    PB = state2;
    Wait(wtime);
    step--;
    break;
```

```
    case 3:
     if(rot==1)
     {
      state2 = 96;
      position = 6;
     }
     else
     {
      state2 = 144;
      position = 9;
     }
     PB = state2;
     Wait(wtime);
     step--;
     break;

    case 6:       if(rot==1)
     {
      state2 = 192;
      position = 12;
     }
     else
     {
      state2 = 48;
      position = 3;
     }
     PB = state2;
     Wait(wtime);
     step--;
     break;

    default:
     state2 = 144;
     PB = state2;
     Wait(wtime);
     step--;
     position = 9;
     break;
   }
   if(step == 0)return;
  }
}

void m2drive(int wtime,int step,int rot)
{
```

```
static int position,state2;

while(step>0)
{
 switch(position)
 {
  case 12:
   if(rot==1)
   {
    state2 = 144;
    position = 9;
   }
   else
   {
    state2 = 96;
    position = 6;
   }
   PC = state2;
   Wait(wtime);
   step--;
   break;

  case 9:
   if(rot==1)
   {
    state2 = 48;
    position = 3;
   }
   else
   {
    state2 = 192;
    position = 12;
   }
   PC = state2;
   Wait(wtime);
   step--;
   break;

  case 3:
   if(rot==1)
   {
    state2 = 96;
    position = 6;
   }
   else
```

```
      {
       state2 = 144;
       position = 9;
      }
      PC = state2;
      Wait(wtime);
      step--;
      break;

     case 6:
      if(rot==1)
      {
       state2 = 192;
       position = 12;
      }
      else
      {
       state2 = 48;
       position = 3;
      }
      PC = state2;
      Wait(wtime);
      step--;
      break;

     default:
      state2 = 144;
      PC = state2;
      Wait(wtime);
      step--;
      position = 9;
      break;
    }
    if(step == 0)return;
  }
}
```

■ 軌道計算プログラム

```
#include <maths.h>

#define deg 0.01745329251994330
#define Keido0 33487.5733333333
#define Ido0 35.7888890
```

```
float orbital_of_sun(float Year,float Mon,float Day,float Hour,float Min,float
Sec);

float keisuu(float Year,float Mon,float Day);

float tihei(float Year,float Mon,float Day,float Hour,float Min,float Sec,float
L,float M,float N);
float gurinizi(float Year,float Mon,float Day,float Hour,float Min,float Sec);
float keido_etc(float G,float Hour,float Min,float Sec);

void Compare2(float C,float C2,int Pinn,int S);
//void Compare2(float A,float h);
float asin(float x);
float atan(float x);

void sun(float Year,float Month,float Day,float Hour,float Min, float Sec)
{
 static float koukei,E,L,M,N,kakudo,a,b,c,d,l,m,n,x,h,A;
 static int guardian;

 koukei=orbital_of_sun(Year,Month,Day,Hour,Min,Sec);
 E=keisuu(Year,Month,Day);

 if(koukei>270)
 {
  koukei=koukei-180.0;
  L=-1.0*cos(koukei*deg);
  M=-1.0*sin(koukei*deg)*cos(E*deg);
  N=-1.0*sin(koukei*deg)*sin(E*deg);
 }
 else
 {
  L=cos(koukei*deg);
  M=sin(koukei*deg)*cos(E*deg);
  N=sin(koukei*deg)*sin(E*deg);
 }
 kakudo=tihei(Year,Month,Day,Hour,Min,Sec,L,M,N);
 if(kakudo>270)
 {
  kakudo=kakudo-180.0;
  a=-cos(kakudo*deg);
  b=-sin(kakudo*deg);
 }
 else
 {
```

```
    a=cos(kakudo*deg);
    b=sin(kakudo*deg);
  }
  c=cos(Ido0*deg);
  d=sin(Ido0*deg);

  l=L*d*a+M*d*b-N*c;
  m=-L*b+M*a;
  n=L*a*c+M*c*b+N*d;
  h=asin(n)/deg;
  A=atan(-m/l)/deg;

  if(l<0)
  {
    A+=180.0;
  }

  if(-85<h && h<85 && -140<A && A<140)
  {
    Compare2(A,0.3125,4,0);
    Compare2(h,0.3125,5,1);
  }
  return();
}

float orbital_of_sun(float Year,float Mon,float Day,float Hour,float Min,float Sec)
{
  static float W,F,J,t,T,Koukei,U[2][19];
  static int W2,X,R,S,guardian;
  static unsigned int Y,Z;

  W=(Year-1900.0)/4.0;
  W2=(Year-1900.0)/4.0;
  F=(Year-1900.0)/4.0-W2;

  Y=1461*W;
  X=(Mon+7.0)/10.0;
  R=1.0-F;
  S=(Mon+4.4)*0.44;
  Z=Y+31.0*Mon+Day+(X-1.0)*R-X*S-27424.0;

  J=Hour/24.0+Min/1440.0+Sec/86400.0;
  t=(J+Z)/365.25;
  T=t+((0.0317*t)+1.43)*0.000001;
```

```
U[0][0]=356.531+359.991*T;
U[0][1]=353.06+719.981*T;
U[0][2]=248.64-19.341*T;
U[0][3]=285.0+329.64*T;
U[0][4]=334.2-4452.67*T;
U[0][5]=6293.7-0.20*T;
U[0][6]=242.4+450.37*T;
U[0][7]=211.1+225.18*T;
U[0][8]=208.0+659.29*T;
U[0][9]=53.5+90.38*T;
U[0][10]=12.1-30.35*T;
U[0][11]=239.1+337.18*T;
U[0][12]=10.1-1.50*T;
U[0][13]=99.1-22.81*T;
U[0][14]=264.8+315.56*T;
U[0][15]=233.8+299.3*T;
U[0][16]=198.1+720.02*T;
U[0][17]=349.6+1079.97*T;
U[0][18]=241.2-44.43*T;

U[1][0]=1.9159-0.00005*T;
U[1][1]=0.0200;
U[1][2]=-0.0048;
U[1][3]=0.0020;
U[1][4]=0.0018;
U[1][5]=0.0018;
U[1][6]=0.0015;
U[1][7]=0.0013;
U[1][8]=0.0008;
U[1][9]=0.0007;
U[1][10]=0.0007;
U[1][11]=0.0006;
U[1][12]=0.0005;
U[1][13]=0.0005;
U[1][14]=0.0004;
U[1][15]=0.0004;
U[1][16]=-0.0004;
U[1][17]=0.0003;
U[1][18]=0.0003;

Koukei=279.0358+360.00769*T;
for(X=0;X<=18;X++)
{
 if(U[0][X]<0)
 {
```

```
    guardian=U[0][X]/360;
    U[0][X]=U[0][X]/360.0;
    U[0][X]=(U[0][X]-guardian)*360+360;
   }
   if(U[0][X]>360)
   {
    guardian=U[0][X]/360;
    U[0][X]=U[0][X]/360.0;
    U[0][X]=(U[0][X]-guardian)*360;
   }
   if(U[0][X]>270)
   {
    U[0][X]=U[0][X]-180.0;
    Koukei-=U[1][X]*sin(U[0][X]*deg);
   }
   else
   {
    Koukei+=U[1][X]*sin(U[0][X]*deg);
   }
  }

  guardian=Koukei/360;
  Koukei=Koukei/360.0;
  Koukei=(Koukei-guardian)*360.0;
  return Koukei;
}

float keisuu(float Year,float Mon,float Day)
{
  static unsigned int a,b,c,i;
  static float j,total,T,E;

  a=(Year-1900)*365;
  b=(Year-1900)/4;
  c=0;
  for(i=1;i<Mon;i++)
  {
   switch(i)
   {
    case 4:
    case 6:
    case 9:
    case 11:
     c+=30;
     break;
```

```
    case 2:
     j=Year-((int)Year/4.0)*4.0;
     if(j==0)
     {
      c+=29;
     }
     else
     {
      c+=28;
     }
     break;
    default:
     c+=31;
   }
  }
  c+=Day;
  total=a+b+c-0.5;
  T=total/36525.0;
  E=23.452294-0.0130125*T-0.0000164*T*T+0.000000503*T*T*T;
  return(E);
}

float tihei(float Year,float Mon,float Day,float Hour,float Min,float Sec,float
L,float M,float N)
{
  static float G,H;
  G=gurinizi(Year,Mon,Day,Hour,Min,Sec);
  H=keido_etc(G,Hour,Min,Sec);
  return(H);
}

float gurinizi(float Year,float Mon,float Day,float Hour,float Min,float Sec)
{
  static unsigned int a,b,i,G2;
  static float c,j,total,Tu,G;

  a=(Year-1900)*365;
  b=(Year-1900)/4;
  c=0;
  for(i=1;i<Mon;i++)
  {
   switch(i)   {
    case 4:
    case 6:
    case 9:
```

```
    case 11:
      c+=30;
      break;
    case 2:
      j=Year-((int)Year/4.0)*4.0;
      if(j==0)
      {
        c+=29;
      }
      else
      {
        c+=28;
      }
      break;
    default:
      c+=31;
   }
  }
  c+=(Day-1.0)+(Hour/24.0)+(Min/1440.0)+(Sec/86400.0);
  total=a+b+c+0.5;
  Tu=total/36525.0;

  G2=(23925.836+8640184.542*Tu+0.0929*Tu*Tu)/86400;
  G=(23925.836+8640184.542*Tu+0.0929*Tu*Tu)/86400.0;
  G=(G-G2)*86400.0;

  return G;
}

float keido_etc(float G,float Hour,float Min,float Sec)
{
  static int Keido[2],World[2],Zikan[2],Hosei[2],Kouseiji[2],guardian;
  static float Z,zikan,keido,world,kouseiji,hosei,hosei2,siita;

  Zikan[0]=G/3600;
  Zikan[1]=(G/60)-(Zikan[0]*60);
  zikan=G-((float)Zikan[0]*3600)-((float)Zikan[1]*60);

  Keido[0]=Keido0/3600;
  Keido[1]=(Keido0/60)-(Keido[0]*60);
  keido=Keido0-((float)Keido[0]*3600)-((float)Keido[1]*60);

  Z=(Hour-9)*3600+Min*60+Sec;

  World[0]=Z/3600;
```

```
 World[1]=(Z/60)-(World[0]*60);
 world=Z-((float)World[0]*3600)-((float)World[1]*60);

 hosei=(((float)World[0]*3600)+((float)World[1]*60)+world)*0.00273791;

 Hosei[0]=hosei/3600;
 Hosei[1]=(hosei/60)-(Hosei[0]*60);
 hosei2=hosei-((float)Hosei[0]*3600)-((float)Hosei[1]*60);

 Kouseiji[0]=Zikan[0]+Keido[0]+World[0]+Hosei[0];
 Kouseiji[1]=Zikan[1]+Keido[1]+World[1]+Hosei[1];
 kouseiji=zikan+keido+world+hosei2;

 if(kouseiji<0)
 {
  kouseiji+=60;
  Kouseiji[1]--;
  if(Kouseiji[1]<0)
  {
    Kouseiji[1]+=60;
    Kouseiji[0]--;
  }
 }
 if(kouseiji>=60)
 {
  Kouseiji[1]+=kouseiji/60;

  guardian=kouseiji/60;
  kouseiji=kouseiji-((float)guardian*60.0);
 }
 if(Kouseiji[1]>=60) {
  Kouseiji[0]++;

  guardian=Kouseiji[1]/60;
  Kouseiji[1]=Kouseiji[1]-((float)guardian*60);
 }
 if(Kouseiji[0]>=24)
 {
  guardian=Kouseiji[0]/60;
  Kouseiji[0]=Kouseiji[0]-((float)guardian*24);
 }

siita=((float)Kouseiji[0]*15.0)+((float)(Kouseiji[1]*15.0)/60.0)+(kouseiji*15.0)/36
00.0;
```

```
  if(siita<0)
  {
    guardian=siita/360;
    siita=siita/360.0;
    siita=(siita-guardian)*360+360;
  }
  if(siita>360)
  {
    guardian=siita/360;
    siita=siita/360.0;
    siita=(siita-guardian)*360;
  }
  return(siita);
}
```

□ asin、atan 計算関数

```
#include <maths.h>

#define N 3000

float asin(float x)
{
  static float A,a,X,X1;
  static unsigned int i,t;

  a=1;
  X1=x;
  X=x;
  A=X;

  for(i=1;i<N;i++)
  {
    if(fabs(X)<1.0e-32)
    {
      break;
    }
    a*=((float)(2*i-1)/(2*i));
    X*=(X1*X1);
    A+=(X*a)/(2*i+1);
  }
  return A;
}

float atan(float x)
```

```
{
  static float i;

  i=x;
  i=x/sqrt(1+x*x);
  return asin(i);
}
```

■ LCD に表示するメニュー、キー操作関係の処理

```
#include "Clock2_Auto.h"
#include "Def.h"
#include "Global.h"

#include <strings.h>
#include <displays.h>

BYTE CurrMenuState;

BYTE PrgHour, PrgMin;
BYTE PrgNum;
BYTE PrgPort;
BYTE PrgOnOff;
BYTE PrgRep;

BOOL ProgModify = false;

void InitDatas(void) {
	CurrMenuState = 0;
	PrgHour = 0;
	PrgMin = 0;
	PrgNum = 0;
	PrgPort = 0;
	PrgOnOff = 0;
}

void MenuProc(void) {
	switch(KP4Value) {
		case 0:
			ProcSELKey();
			break;
		case 2:
			ProcUPKey();
			break;
		case 3:
```

```
                ProcDELKey();
                break;
            case 4:
                ProcESCKey();
                break;
            case 6:
                ProcDOWNKey();
                break;
            case 7:
                ProcENTKey();
                break;
        }
}

void ProcSELKey(void) {
        switch(CurrMenuState) {
            case 0:             //
            case 1:
            case 2:
                CurrMenuState++;
                break;
            case 3:
                CurrMenuState = 0;
                break;
            case 20:
            case 25:
                CurrMenuState = 21;
                break;
            case 21:
            case 22:
                CurrMenuState++;
                break;
            case 23:
                CurrMenuState = 20;
                break;
        }
        DisplayMenu(CurrMenuState);
}

void ProcUPKey(void) {
        char str_buf[16];
        switch(CurrMenuState) {
            case 2:
                PrgNum++;
                if(PrgNum > 9) {
```

```
            PrgNum = 0;
        }
        break;
case 3:
        PrgNum++;
        if(PrgNum > 9) {
            PrgNum = 0;
        }
        break;
case 10:
        Hour++;
        if(Hour > 23) {
            Hour = 0;
        }
        break;
case 11:
        Min++;
        if(Min > 59) {
            Min = 0;
        }
        break;
case 12:
        Sec++;
        if(Sec > 59) {
            Sec = 0;
        }
        break;
case 21:
        PrgPort++;
        if(PrgPort > 3) {
            PrgPort = 0;
        }
        ProgModify = true;
        break;
case 22:
        PrgOnOff++;
        PrgOnOff &= 1;
        ProgModify = true;
        break;
case 23:
        PrgRep++;
        PrgRep &= 1;
        ProgModify = true;
        break;
case 20:
```

```
                PrgHour++;
                if(PrgHour > 23) {
                    PrgHour = 0;
                }
                ProgModify = true;
                break;
            case 25:
                PrgMin++;
                if(PrgMin > 59) {
                    PrgMin = 0;
                }
                ProgModify = true;
                break;
            default:
                return;
        }
        DisplayMenu(CurrMenuState);
}

void ProcESCKey(void) {
        WORD prg00;
        union {
            WORD wd;
            BYTE bd[2];
        } prg2;

        switch(CurrMenuState) {
            case 20:
            case 21:
            case 22:
            case 23:
            case 25:
                if(ProgModify) {
                    ProgModify = false;
                    prg2.bd[0] = 0;
                    prg2.bd[1] = (PrgOnOff << 3) | (PrgPort << 4) | (PrgRep << 6)
| 0x80;
                    prg00 = ((WORD)PrgMin | ((WORD)PrgHour << 6));
                    prg2.wd |= prg00;

                    TrgSetProg(PrgNum, 0, prg2.wd);
                }
                CurrMenuState = 2;
                break;
            case 29:
```

```
            CurrMenuState = 2;
            break;
        case 3:
            CurrMenuState = 0;
            break;
        // 時計設定モード解除
        case 10:
        case 11:
        case 12:
            DI_CCP1;
            CLR_CCP1IF;
            Secc = 0;
            TF1Sec = 0;
            TMR1L = 0;
            TMR1H = 0;
            EI_CCP1;
            CurrMenuState = 0;
            break;

        default:
            CurrMenuState = 0;
            break;
    }
    DisplayMenu(CurrMenuState);
}

void ProcDOWNKey(void) {
    char str_buf[16];

    switch(CurrMenuState) {
        case 2:
            if(PrgNum == 0) {
                PrgNum = 9;
            } else {
                PrgNum--;
            }
            break;
        case 3:
            if(PrgNum == 0) {
                PrgNum = 9;
            } else {
                PrgNum--;
            }
            break;
        case 10:
```

```
        if(Hour == 0) {
            Hour = 23;
        } else {
            Hour--;
        }
        break;
    case 11:
        if(Min == 0) {
            Min = 59;
        } else {
            Min--;
        }
        break;
    case 12:
        if(Sec == 0) {
            Sec = 59;
        } else {
            Sec--;
        }
        break;
    case 21:
        if(PrgPort == 0) {
            PrgPort = 3;
        } else {
            PrgPort--;
        }
        ProgModify = true;
        break;
    case 22:
        PrgOnOff++;
        PrgOnOff &= 1;
        ProgModify = true;
        break;
    case 23:
        PrgRep++;
        PrgRep &= 1;
        ProgModify = true;
        break;
    case 20:
        if(PrgHour == 0) {
            PrgHour = 23;
        } else {
            PrgHour--;
        }
        ProgModify = true;
```

```
                break;
            case 25:
                if(PrgMin == 0) {
                    PrgMin = 59;
                } else {
                    PrgMin--;
                }
                ProgModify = true;
                break;
            default:
                return;
        }
        DisplayMenu(CurrMenuState);
}

void ProcENTKey(void) {
        switch(CurrMenuState) {
            case 1:
                CurrMenuState = 10;
                break;
            case 2:
                CurrMenuState = 20;
                break;
            case 3:
                TrgClearAll();
                CurrMenuState = 0;
                break;
            case 10:
                CurrMenuState++;
                // ToDo:
                break;
            case 11:
                CurrMenuState++;
                // ToDo:
                break;
            case 12:
                CurrMenuState = 10;
                // ToDo:
                break;
            case 20:
                CurrMenuState = 25;
                break;
            case 25:
                CurrMenuState = 20;
                break;
```

```
            case 29:
                ClearProg(PrgNum);
                CurrMenuState = 2;
                break;
            default:
                return;
        }
        DisplayMenu(CurrMenuState);
}

void ProcDELKey(void) {
        if(CurrMenuState == 2) {
            CurrMenuState = 29;
            DisplayMenu(CurrMenuState);
        }
}

void DisplayMenu(BYTE currsts) {
        WORD prg01, prg02;
        char str_buf[16];

        DI_GLOBAL;
        switch(currsts) {
            case 0:
                LCDClear();
                sprintf(str_buf,    "%02d:%02d:%02d",    Hour,    Min,    Sec);
LCDString(str_buf);
                break;
            case 1:
                LCDClear();
                LCDString("Clock set?(ENT)");
                break;
            case 2:
                LCDClear();
                sprintf(str_buf, "Prg No.%02d", PrgNum + 1);
                LCDString(str_buf);
                LCDPrintAt(0, 1);
                GetProgram(PrgNum, &prg01, &prg02);
                if(prg02 & 0x8000) {
                    PrgMin = prg02 & 0x3F;
                    prg02 = prg02 >> 6;
                    PrgHour   = prg02 & 0x1F;
                    prg02 = prg02 >> 5;
                    PrgOnOff = prg02 & 0x01;
                    prg02 = prg02 >> 1;
```

```
                    PrgPort = prg02 & 0x03;
                    prg02 = prg02 >> 2;
                    PrgRep = prg02 & 0x01;
                    sprintf(str_buf, " %02d:%02d P%d ", PrgHour, PrgMin,
PrgPort + 1);
                    LCDString(str_buf);
                    if(PrgOnOff) {
                        LCDString("ON   ");
                    } else {
                        LCDString("OFF ");
                    }
                    if(PrgRep) {
                        LCDc('R');
                    }
                } else {
                    LCDString(" --:-- -- ---");
                    if((prg01 == 0xFFFF) || (prg02 == 0xFFFF)) {
                        PrgMin = 0;
                        PrgHour = 0;
                        PrgOnOff = 1;
                        PrgPort = 0;
                        PrgRep = 1;
                    }
                }
                ProgModify = false;
                break;
            case 3:
                LCDClear();
                LCDString("All Clear?(ENT)");
                break;
            case 10:
            case 11:
            case 12:
                LCDClear();
                sprintf(str_buf, "TIME %02d:%02d:%02d", Hour, Min, Sec);
                LCDString(str_buf);
                DispCursor(currsts - 10);
                break;
            case 20:
            case 25:
                LCDClear();
                sprintf(str_buf, "No.%02d", PrgNum + 1);
                LCDString(str_buf);
                LCDPrintAt(0, 1);
                sprintf(str_buf, "Time? %02d:%02d", PrgHour, PrgMin);
```

```
            LCDString(str_buf);
            DispCursor(currsts - 17);
            break;
        case 21:
            LCDClear();
            sprintf(str_buf, "No.%02d", PrgNum + 1);
            LCDString(str_buf);
            LCDPrintAt(0, 1);
            sprintf(str_buf, "Port? %02d", PrgPort + 1);
            LCDString(str_buf);
            break;
        case 22:
            LCDClear();
            sprintf(str_buf, "No.%02d", PrgNum + 1);
            LCDString(str_buf);
            LCDPrintAt(0, 1);
            if(PrgOnOff) {
                // ON
                LCDString("On/Off? ON");
            } else {
                // OFF
                LCDString("On/Off? OFF");
            }
            break;
        case 23:
            LCDClear();
            sprintf(str_buf, "No.%02d", PrgNum + 1);
            LCDString(str_buf);
            LCDPrintAt(0, 1);
            if(PrgRep) {
                // ON
                LCDString("Repeat? ON");
            } else {
                // OFF
                LCDString("Repeat? OFF");
            }
            break;
        case 29:
            LCDClear();
            LCDString("Delete?(ENT)");
            break;
    }
    EI_GLOBAL;
}
```

```
void DispCursor(int ix) {
    int x, y;
    switch(ix) {
        case 0:
            x = 6;
            y = 0;
            break;
        case 1:
            x = 9;
            y = 0;
            break;
        case 2:
            x = 12;
            y = 0;
            break;
        case 3:
            x = 7;
            y = 1;
            break;
        default:
            x = 10;
            y = 1;
            break;
    }
    LCDPrintAt(x, y);
    LCDOnOff(1, 0, 1);

}
```

■ 予約・プログラム関係の処理

```
#include "Def.h"
#include "Global.h"

BOOL EEWDone;

union CVDATA cnv;

BYTE EEPPtr;
BYTE ProgCnt;
BOOL ProgPtr;
BYTE EEPRBuf[4];
BYTE ProgIx;

BOOL TFClearAll = false;
```

```
BOOL TFSetPrg     = false;
BOOL TFSelClr     = false;

WORD PrgClearMask;
int SelClrPtr;

void ClearProg(BYTE ix) {
      EEPPtr = EEPR_PRG_TOP + ix * 4 + 3;
      ProgIx = ix;

      EEWDone = false;
      WriteEEData(EEPPtr, 0);
}

BOOL GetProgram(BYTE ix, WORD *prg1, WORD *prg2) {
      WORD dat;
      BYTE adrs;
      BYTE ddata;

      if(ix < PROG_MAX) {
            adrs = ix * 4 + EEPR_PRG_TOP;
            dat = ((WORD)ReadEEData(adrs + 1) << 8) | ReadEEData(adrs);
            *prg1 = dat;
            dat = ((WORD)ReadEEData(adrs + 3) << 8) | ReadEEData(adrs + 2);
            *prg2 = dat;
            return true;
      } else {
            return false;
      }
}

void CompareTime(void) {
      int ix;
      WORD prg1, prg2;
      WORD hour, min;
      BYTE sw;
      WORD prg_bit = 0x0001;

      PrgClearMask = 0;

      for(ix = 0; ix < PROG_MAX; ix++) {
            GetProgram(ix, &prg1, &prg2);
            if((prg2 & 0x8000) != 0) {
                  hour = (prg2 >> 6) & 0x001F;
                  if(hour == Hour) {
```

```
                    min = prg2 & 0x003F;
                    if(min == Min) {
                        sw = (BYTE)(prg2 >> 11);
                        if(!PortCtrl(sw)) {
                            PrgClearMask |= prg_bit;
                        }
                    }
                }
            }
            prg_bit <<= 1;
        }
        if(PrgClearMask != 0) {
            TrgSelClr();
        }
}

BOOL PortCtrl(BYTE sw) {
        BYTE on_off, ch, repeat;

        on_off = sw & 0x01;
        ch = (sw >> 1) & 0x03;
        repeat = (sw >> 3) & 0x01;

        switch(ch) {
            case 0:
                PC.B0 = on_off;
                break;
            case 1:
                PC.B1 = on_off;
                break;
            case 2:
                PD.B0 = on_off;
                break;
            default:
                PE.B2 = on_off;
                break;
        }
        return (BOOL)repeat;
}

void TrgClearAll(void) {
        ProgCnt = 0;
        EEPPtr = EEPR_PRG_TOP + 3;
        EEWDone = false;
        WriteEEData(EEPPtr, 0);
```

```
        EEPPtr += 4;

        TFClearAll = true;
}

void TskClearAll(void) {
        if(TFClearAll && EEWDone) {

                EEWDone = false;
                if(ProgCnt < PROG_MAX) {
                        WriteEEData(EEPPtr, 0);
                        EEPPtr += 4;
                        ProgCnt++;
                } else {
                        TFClearAll = false;
                }
        }
}

void TrgSetProg(BYTE ix, WORD dat1, WORD dat2) {
        ProgIx = ix;

        EEPRBuf[0] = (BYTE)(dat1);
        EEPRBuf[1] = (BYTE)(dat1 >> 8);
        EEPRBuf[2] = (BYTE)(dat2);
        EEPRBuf[3] = (BYTE)(dat2 >> 8);
        EEPPtr = EEPR_PRG_TOP + ix * 4;

        EEWDone = false;
        WriteEEData(EEPPtr, EEPRBuf[0]);
        ProgPtr = 1;
        TFSetPrg = true;
}

void TskSetProg(void) {
        if(TFSetPrg && EEWDone) {

                EEWDone = false;
                if(ProgPtr < 4) {
                        WriteEEData(EEPPtr + ProgPtr, EEPRBuf[ProgPtr]);
                        ProgPtr++;
                } else {
                        TFSetPrg = false;
                        DisplayMenu(CurrMenuState);
                }
```

```
    }
}

void TrgSelClr(void) {
    SelClrPtr = 0;
    EEWDone = true;
    TFSelClr = true;
}

void TskSelClr(void) {
    if(TFSelClr && EEWDone) {

        EEWDone = false;
        if(SelClrPtr < PROG_MAX) {

            if(PrgClearMask & 0x0001) {
                ClearProg(SelClrPtr);
            } else {
                EEWDone = true;
            }
            PrgClearMask >>= 1;
            SelClrPtr++;
        } else {
            PrgClearMask = 0;
            TFSelClr = false;
        }
    }
}
```

■ 定数、レジスタ操作マクロの設定

```
#ifndef DEF_H
#define DEF_H

typedef signed char BOOL;
typedef unsigned long DWORD;

#define true 1
#define false 0

#define TST_INT0IF (INTCON&(1<<INT0IF))
#define TST_INT1IF (INTCON3&(1<<INT1IF))
#define TST_INT2IF (INTCON3&(1<<INT2IF))
#define TST_TMR0IF (INTCON&(1<<TMR0IF))
#define TST_RBIF    (INTCON&(1<<RBIF))
```

```
#define TST_PSPIF    (PIR1&(1<<PSPIF))
#define TST_ADIF     (PIR1&(1<<ADIF))
#define TST_RCIF     (PIR1&(1<<RCIF))
#define TST_TXIF     (PIR1&(1<<TXIF))
#define TST_SSPIF    (PIR1&(1<<SSPIF))
#define TST_TMR2IF (PIR1&(1<<TMR2IF))
#define TST_TMR1IF (PIR1&(1<<TMR1IF))
#define TST_EEIF     (PIR2&(1<<EEIF))
#define TST_BCLIF    (PIR2&(1<<BCLIF))
#define TST_LVDIF    (PIR2&(1<<LVDIF))
#define TST_TMR3IF (PIR2&(1<<TMR3IF))
#define TST_CCP1IF (PIR1&(1<<CCP1IF))
#define TST_CCP2IF (PIR2&(1<<CCP2IF))

#define CLR_INT0IF INTCON&=~(1<<INT0IF)
#define CLR_INT1IF INTCON3&=~(1<<INT1IF)
#define CLR_INT2IF INTCON3&=~(1<<INT2IF)
#define CLR_TMR0IF INTCON&=~(1<<TMR0IF)
#define CLR_RBIF     INTCON&=~(1<<RBIF)
#define CLR_PSPIF    PIR1&=~(1<<PSPIF)
#define CLR_ADIF     PIR1&=~(1<<ADIF)
#define CLR_RCIF     PIR1&=~(1<<RCIF)
#define CLR_TXIF     PIR1&=~(1<<TXIF)
#define CLR_SSPIF    PIR1&=~(1<<SSPIF)
#define CLR_TMR2IF PIR1&=~(1<<TMR2IF)
#define CLR_TMR1IF PIR1&=~(1<<TMR1IF)
#define CLR_EEIF     PIR2&=~(1<<EEIF)
#define CLR_BCLIF    PIR2&=~(1<<BCLIF)
#define CLR_LVDIF    PIR2&=~(1<<LVDIF)
#define CLR_TMR3IF PIR2&=~(1<<TMR3IF)
#define CLR_CCP1IF PIR1&=~(1<<CCP1IF)
#define CLR_CCP2IF PIR2&=~(1<<CCP2IF)

#define EI_GLOBAL INTCON|=1<<GIE
#define EI_PERIP   INTCON|=1<<PEIE
#define EI_INT0    INTCON|=1<<INT0IE
#define EI_INT1    INTCON3|=1<<INT1IE // INT1
#define EI_INT2    INTCON3|=1<<INT2IE // INT2
#define EI_TMR0    INTCON|=1<<TMR0IE
#define EI_RB      INTCON|=1<<RBIE
#define EI_PSP     PIE1|=1<<PSPIE
#define EI_AD      PIE1|=1<<ADIE
#define EI_RC      PIE1|=1<<RCIE
#define EI_TX      PIE1|=1<<TXIE
#define EI_SSP     PIE1|=1<<SSPIE
```

```
#define EI_TMR2     PIE1|=1<<TMR2IE
#define EI_TMR1     PIE1|=1<<TMR1IE
#define EI_EE       PIE2|=1<<EEIE
#define EI_BCL      PIE2|=1<<BCLIE
#define EI_LVD      PIE2|=1<<LVDIE
#define EI_TMR3     PIE2|=1<<TMR3IE
#define EI_CCP1     PIE1|=1<<CCP1IE
#define EI_CCP2     PIE2|=1<<CCP2IE

#define DI_GLOBAL INTCON&=~(1<<GIE)
#define DI_PERIP    INTCON&=~(1<<PEIE)
#define DI_INT0     INTCON&=~(1<<INT0IE)
#define DI_INT1     INTCON&=~(1<<INT1IE)
#define DI_INT2     INTCON&=~(1<<INT2IE)
#define DI_TMR0     INTCON&=~(1<<TMR0IE)
#define DI_RB       INTCON&=~(1<<RBIE)
#define DI_PSP      PIE1&=~(1<<PSPIE)
#define DI_AD       PIE1&=~(1<<ADIE)
#define DI_RC       PIE1&=~(1<<RCIE)
#define DI_TX       PIE1&=~(1<<TXIE)
#define DI_SSP      PIE1&=~(1<<SSPIE)
#define DI_TMR2     PIE1&=~(1<<TMR2IE)
#define DI_TMR1     PIE1&=~(1<<TMR1IE)
#define DI_EE       PIE2&=~(1<<EEIE)
#define DI_BCL      PIE2&=~(1<<BCLIE)
#define DI_LVD      PIE2&=~(1<<LVDIE)
#define DI_TMR3     PIE2&=~(1<<TMR3IE)
#define DI_CCP1     PIE1&=~(1<<CCP1IE)
#define DI_CCP2     PIE2&=~(1<<CCP2IE)

#define RB_PULLUP_ON    INTCON2&=~(1<<RBPU)
#define RB_PULLUP_OFF INTCON2|=(1<<RBPU)
#define ADC_START ADCON0|=(1<<GO)

#endif
```

■ 関数のプロトタイプ宣言

```
#include "Clock2_Auto.h"

#ifndef GLOBAL_H
#define GLOBAL_H

#define EEPR_PRG_TOP 4
#define PROG_MAX 10
```

```
union CVDATA {
      BYTE bdata[4];
      DWORD dwdata;
};

extern BYTE Sec;
extern BYTE Min;
extern BYTE Hour;
extern BYTE PrgHour;
extern BYTE PrgMin;
extern BYTE PrgNum;
extern BYTE PrgPort;
extern BYTE PrgOnOff;
extern BYTE PrgRep;
extern BYTE CurrMenuState;

extern BYTE Secc;
extern BYTE TF1Sec;

extern BOOL EEWDone;

void MenuProc(void);
void InitDatas(void);

void DisplayMenu(BYTE CurrMainState);
void DispCursor(int ix);

void ProcSELKey(void);
void ProcUPKey(void);
void ProcESCKey(void);
void ProcDOWNKey(void);
void ProcENTKey(void);
void ProcDELKey(void);
void MenuProc(void);

BOOL GetProgram(BYTE ix, WORD *prg1, WORD *prg2);

void ClearProg(BYTE ix);

void CompareTime(void);

void TrgClearAll(void);
void TskClearAll(void);
void TrgSetProg(BYTE ix, WORD prg1, WORD prg2);
```

```
void TskSetProg(void);
void TrgSelClr(void);
void TskSelClr(void);

BOOL PortCtrl(BYTE sw);

#endif
```