平成 21 年度卒業論文

大型低速風洞内飛行実験装置の高性能化

福岡工業大学　工学部
知能機械工学科

06E1059 山路 健太

06E1060 山本 清貴

指導教員　河村良行　　指導院生　柴田光紀

第 1 章 緒言

近年、モータやバッテリの小型化が進み、小型飛翔体の研究が発達してきた。中でも羽ばたき飛翔体は外乱に強く、ホバリングができるという特徴がある。

本研究では風洞内で小型飛翔体を自由飛行させ、その飛行メカニズムを解明することを目的としている。

今年度は羽ばたき飛翔体の実験導入までには至らなかったが、昨年度の飛行制御アルゴリズムの見直しと飛行制御システムの遅れ補正の導入により飛行精度の向上を図る実験を行った。

また LabVIEWによる直感的でよりわかりやすいプログラムの開発を行った。

第 2 章 飛行制御システムの構成

2.1 飛行実験装置の概要

制御システムの概要を図 1 に示す。機体に上面と側面のマーカーを取り付け、２台の CCD カメラでマーカーの画像を撮影し、撮影した画像を画像処理装置内で処理を行い、マーカーの位置、角度を検出する。その検出したデータもとにパソコン内でＰＤ制御の計算を行い、D/A 変換器を通して無線制御装置とインバータへ送る。無線制御装置は D/A 変換器から送られてくる電圧を送信機に電圧をかけて、赤外線で機体に取り付けてある受信機に信号を送る。信号に応じて受信機から機体に搭載されたモータとラダーを動かして飛行体を制御している。インバータを通じて風洞送風機のモータを操作することで風速を変化させた。

2.2 画像処理装置

機体に取り付けたマーカーの位置や角度を検出させるために CCD カメラを用いた画像処理装置 CV-3000（KEYENCE 社製）を使用した。CCD カメラと本体ユニットをそれぞれ図 2、図 3 に示す。画像処理装置では指定された色の塊が一番大きいものを検出する、【ブロブ】計測という方法を用いた。上面マーカーで機体の前後左右の位置とヨー角を検出し、側面のマーカーで機体の高さと迎角を検出することができる。

2.3 通信手段

昨年度は画像処理装置とパソコン間は USB ケーブルで行っていた。本年度は LabVIEW の通信が USB ケーブルでできなかったため、Ethernet 接続を行った。Ethernet 接続は我々が普段インターネットに使うときに欠かせない LAN ケーブルを用いた接続方式である。しかし、LAN ケーブルといってもインターネット接続に用いているストレート LAN ケーブルではなく、パソコン同士を通信させる際に用いるクロス LAN ケーブルを使用した。ストレート LAN ケーブルとクロス LAN ケーブルを図 4(a)、図 4(b) に示す。

2.4 D/A 変換機

D/A 変換器は Visual Basic の場合、Measurement Computing Corporation の USB-1208FS を使用しており、LabVIEW の場合、National Instrument の USB6009 を使用している。D/A 変換器はそれぞれアナログ出力が２ポートしかなく、赤外線無線機とインバータを制御するためには４ポート必要なので D/A 変換器を２台組み合わせて使用している。USB-1208FS を図 5、USB6009 を図 6 に示す。

2.5 大型低速風洞装置

今回、大型低速風洞の精度向上のため、木製から鉄製へと変えた。変更した大型低速風洞装置を図 7 に示す。風洞の風速を制御するために送風機のモータ回転数をインバータ【3G3MV-A2007（オムロン製）】によって制御している。インバータは送風機の電源を供給するときに電圧の周波数や流す電流を制御することで送風機の制御を行ってい

る。また、アナログ電圧をインバータの FR ポート、FC ポートにかけることで送風機のモータ回転数を制御することができるようになっている。インバータを図 8 に示す。

2.6 電力増幅回路

D/A 変換器の電源は USB 端子から電力を供給されているため 0[V]から最大で＋4[V]までのアナログ電圧しか出力されないのでモータやラダーを動かすためには電圧や電流が足りない。その上、ラダーを左右に動かすためには±電圧が必要である。そこでこれらの問題を解決するために電力増幅回路を使用した。電力増幅回路図を図 9 に示す。電力増幅回路は 12[V]の電源が２つ入っており、OP アンプによる差動増幅回路とトランジスタによる電流増幅回路で構成している。OP アンプが行っていることを式で表すと、Vout=A×(2-Vin)となる。これは信号電圧 Vin と２Ｖの一定電圧を OP アンプに通じて±電圧に変換して差動増幅回路の可変抵抗の抵抗値を変えることで OP アンプの増幅率 A を調整する。OP アンプから出力された電圧 Vout をトランジスタによって電流を増幅して出力される。定格電流 1 [A]のトランジスタ（2SC495、2SA496）の電力増幅回路を図 10 に示す。

2.7 無線制御装置

無線通信は機体のコントローラーの回路を利用した。コントローラーの回路はモータの強弱やラダーを左右に操作する操作レバーを通じて可変抵抗が連動して動くようになっていた。可変抵抗の抵抗が変動している所の電圧を測定すると操作レバーに追従して電圧が 0[V]から 3.2[V]の範囲で変動していた。そこで可変抵抗から固定抵抗に変更してそこに D/A 変換器からの信号電圧をかけることでパソコンからの信号に合わせてモータやラダーを動かすことが可能になった。無線制御装置を図 11 に示す。

また回路の電源として AC100[V]から DC15[V]に変換する AC 安定化電源[Power Supply(TDK 社製)]を使用しており、電圧が高いので 15[V]から 6[V]へ落とす可変電圧回路を利用している。可変電圧回路を図 12 に示す。

2.8 制御プログラム

制御プログラムは Visual Studio 2005 と LabVIEW を使用した。画像処理装置で測定したマーカーの位置や角度、測定回数が Visual Studio 2005 の場合は USB、LabVIEW の場合クロス LAN ケーブルを通じてパソコンに送られる。制御には PD 制御を用いており、画面上の比例係数 Kp、微分係数 Kd の制御器に数値を入力することで制御電圧を求めている。制御電圧はパソコンに接続された D/A 変換器によりアナログ電圧として出力されるようプログラムされている。

2.9 実験に使用した機体

従来は手作りの小型飛翔体を制作し、飛行させたが脆く飛行精度が安定しないため、タイヨー社製のウルトラライトプレーンを使用することにした。仕様と機体を図 13 に示す。

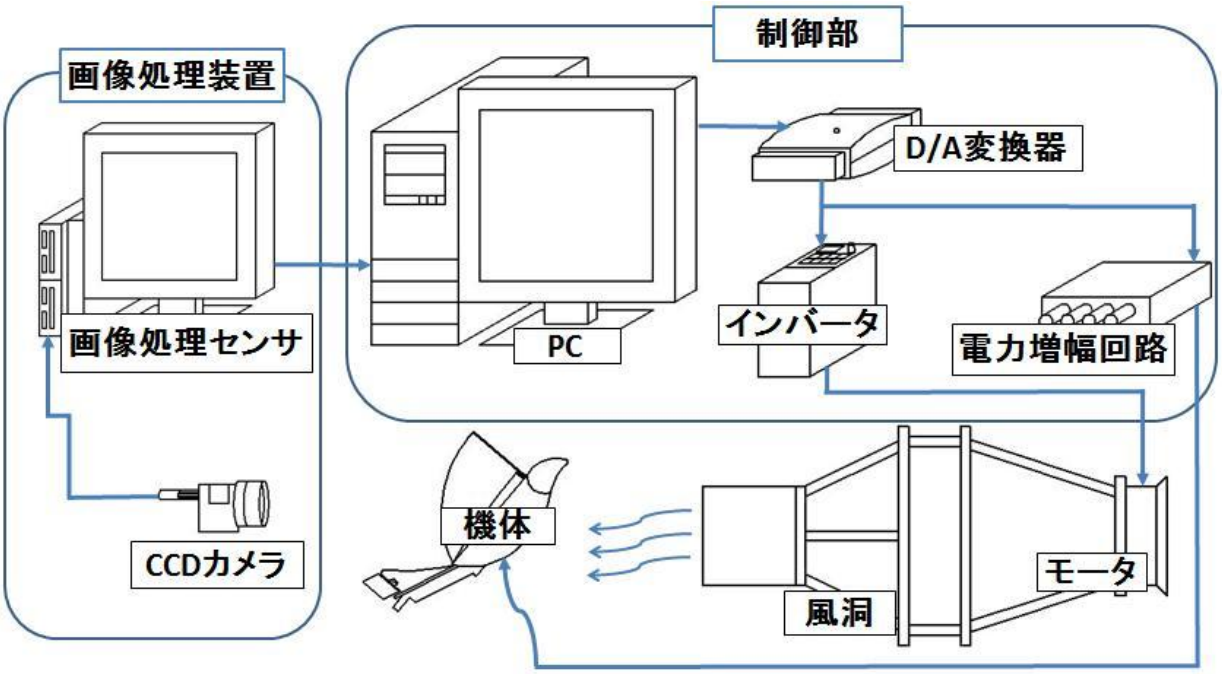
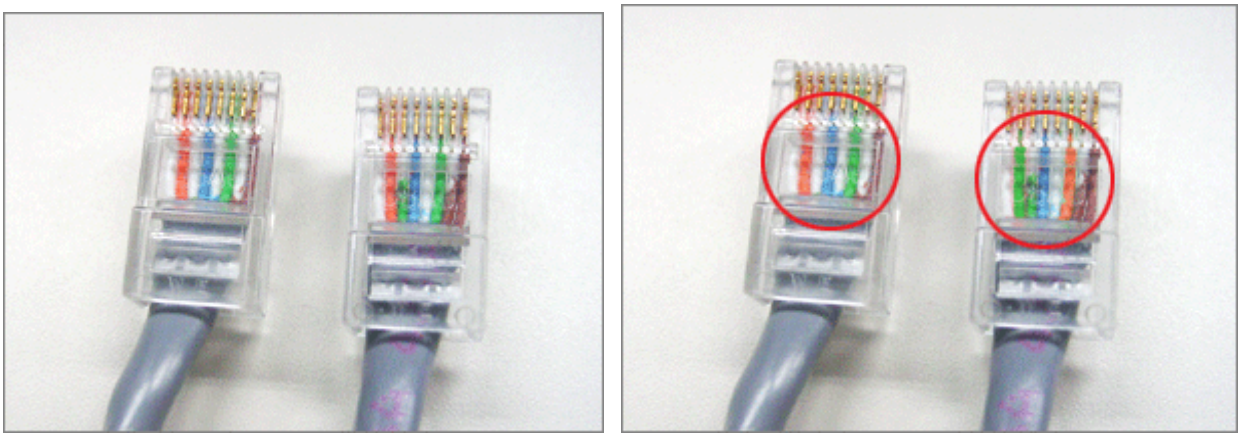


図 1　飛行制御システム

図 2　画像処理装置(CV-3000)

図 3　CCD カメラ



(a)　ストレート LAN ケーブル　　(b)　クロス LAN ケーブル

図 4　LAN ケーブルの比較

図 5　USB-1208FS

図 6　USB6009

図 7　大型低速風洞装置

図 8　インバータ

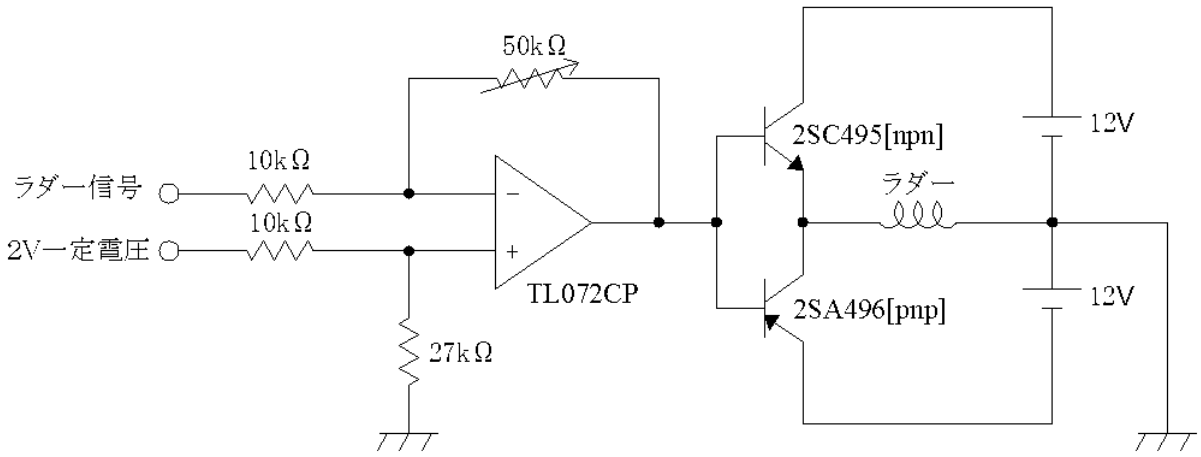


図 9　電力増幅回路図

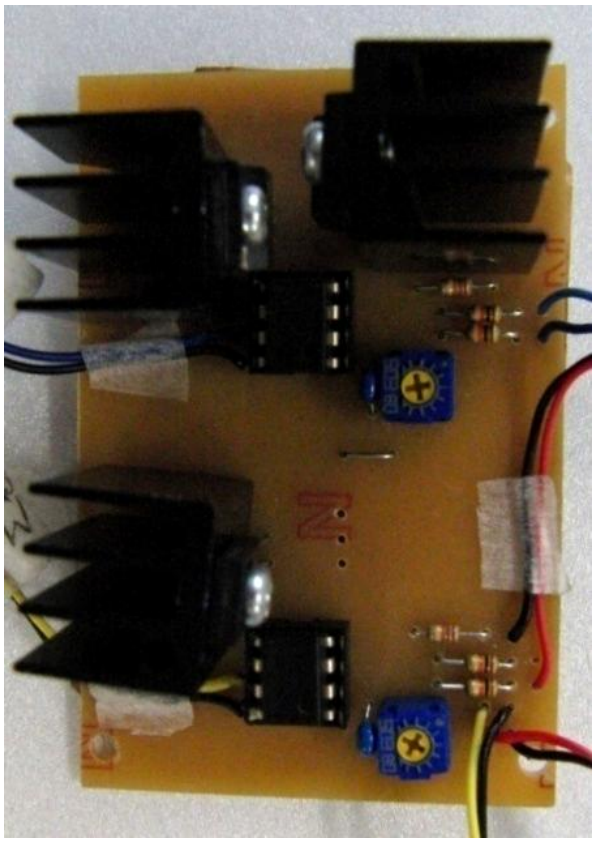


図 10　電力増幅回路

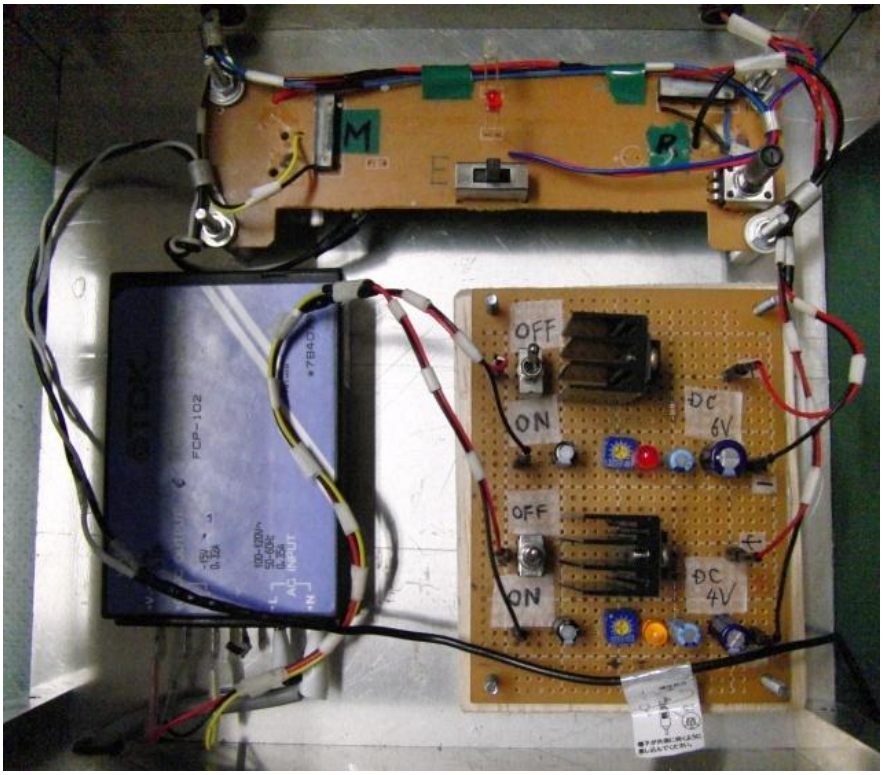


図 11　無線制御装置

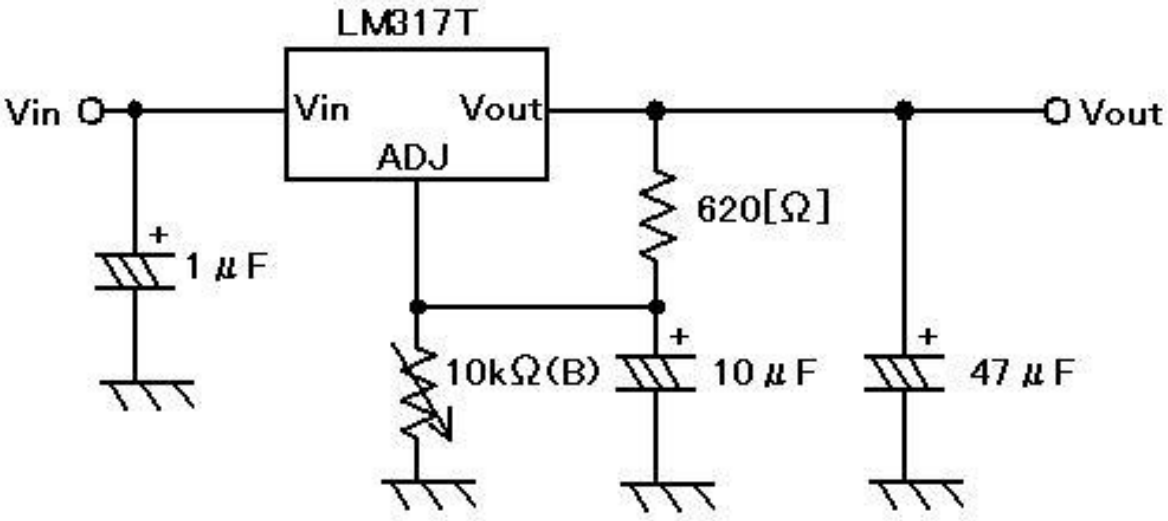


図 12　可変電圧回路図

固定翼機
ウルトラライトプレーン HF [タイヨー社製]
全翼長： 270 [mm]　　翼弦： 85[mm]
全長： 220 [mm]　　重量：11.8 [g]

図 13　固定翼機と仕様

第 3 章 LabVIEW 制御プログラム

3.1 LabVIEW について

LabVIEW とは Laboratory Virtual Instrumentation Engineering Workbench を略したものであり、National Instrument が開発した直感的でわかりやすいグラフィカルプログラム言語である。従来のプログラミング言語は文でプログラム作成するが、VI（Virtual Instrument）というアイコン同士を画面上で配線することでプログラムを作成することができ、初心者にわかりやすい。

例として図 14 に示す。だが従来のプログラミング言語を学んだ者に対しては理解が難しい。VI は演算や定数や While ループなど多種多様であり、すべてを把握するのは容易ではない。しかし、ある程度 VI を把握していると数分でプログラムを作成することができる。入力や計算式や出力が多いほど VI が増え、線が複雑になってしまい、分かりにくくなってしまう。だがフォミューラノードという言語を用いた VI で分かりやすくすることができる。例として図 15 に示す。

3.2 画像処理装置からのデータの計測

プログラムが複雑なため、VI が多く線が複雑になるので今回はサブ VI という VI を格納するプログラムを使用した。例として図 16 に示す。

計測の部分は画像処理装置（CV-3000）から送られてくる 7 つのデータ［T1.+*X*.+*Y*.+*θ*.+*X*2.+*Y*2.+*θ*2.計測回数］を取り込み、要らない部分を省き、VI の配列を用いて *X*、*Y*、*θ*‥として一つ一つデータに分ける。分けたデータは計算や計測に使われる。計測におけるプログラムを図 17 に示す。

3.3 PD 制御による計算

計算では風洞における風速と周波数、サンプリングタイム、モータ制御電圧、ラダー制御電圧の補正ありと補正なしの 6 つの計算を行っている。補正ありでは式が多いため、フォミュラーノードを用いて作成を行った。計算におけるプログラムを図 18-1、18-2、18-3、18-4 に示す。

3.4 データの出力

出力では計算されたモータ制御電圧とラダー制御電圧、風洞制御のためのインバータにアナログ電圧、ラダー制御のためのアナログ電圧を出力している。LabVIEW では DAQ アシスタントという入力、出力専用 VI を用いることで出力を行っている。

DAQ アシスタントは簡易と上級があり、簡易の場合だと簡単に設定するだけで出力ができるが 100ms の遅れが発生するおそれがある。上級の場合だと細かい設定をしないといけず、上級 DAQ アシスタントはいくつかの VI で構成されるため、VI が多い。しかし遅れが発生する心配はない。出力におけるプログラムを図 19 に示す。

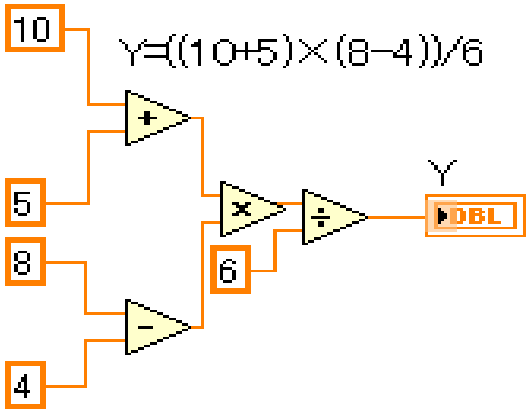


図 14　典型的な LabVIEW のプログラム

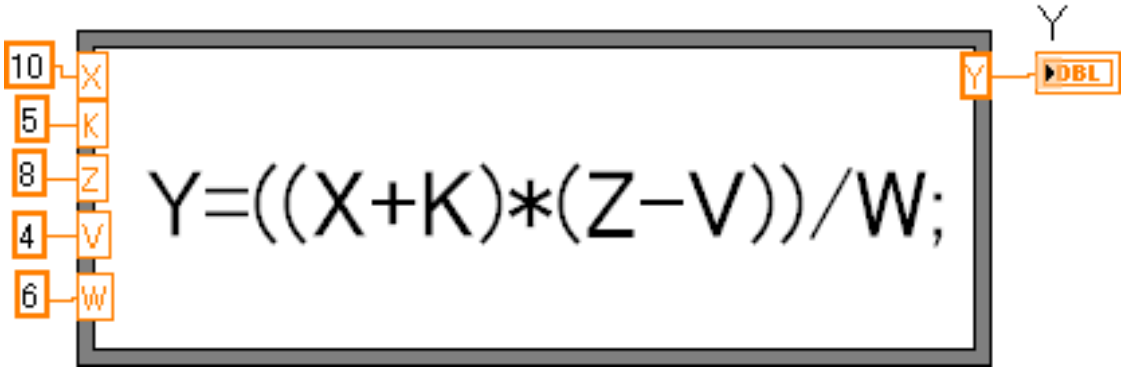


図 15　フォーミュラノードを用いた LabVIEW のプログラムの例

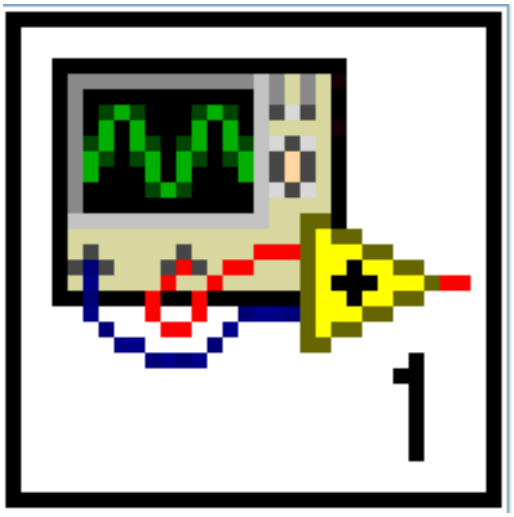


図 16　サブ VI

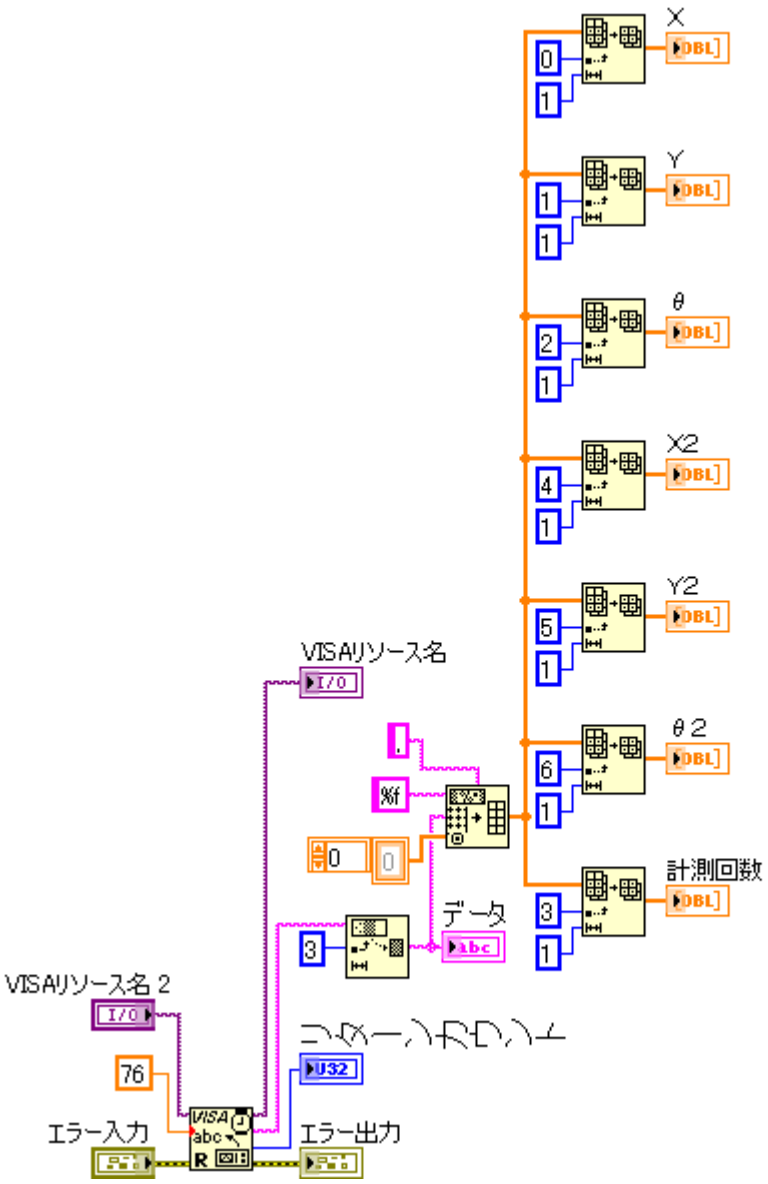


図 17　画像処理装置からのデータの計測プログラム

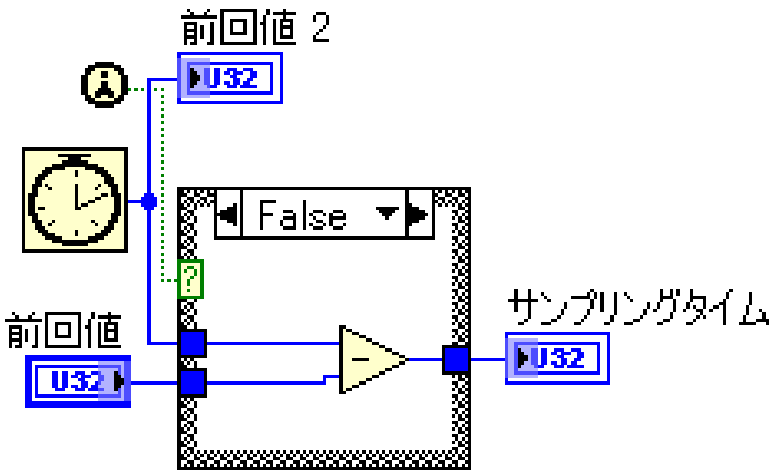


図 18-1　　サンプリングタイムの時間の計算プログラム

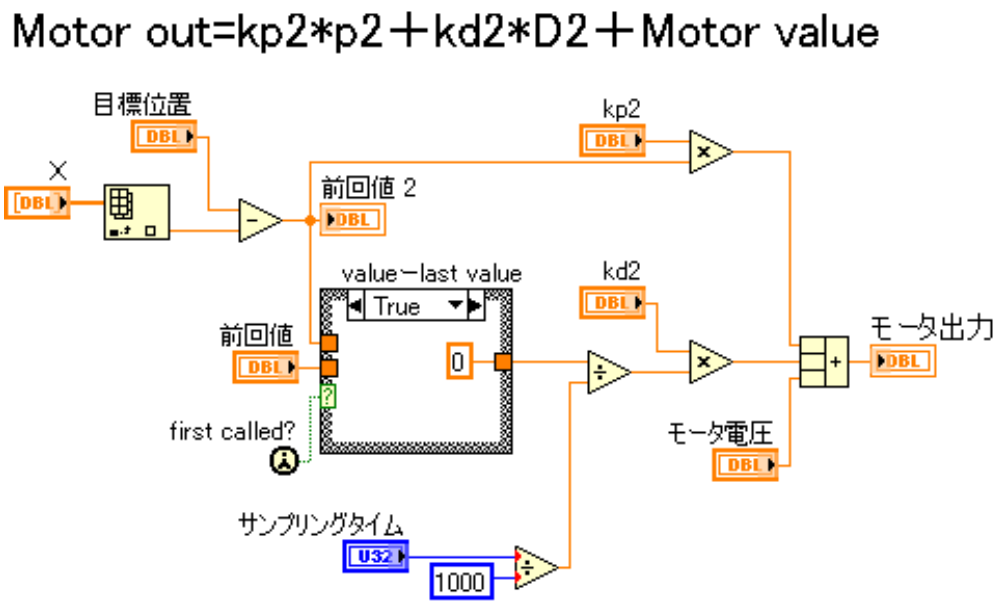


図 18-2　モータ制御電圧の計算プログラム

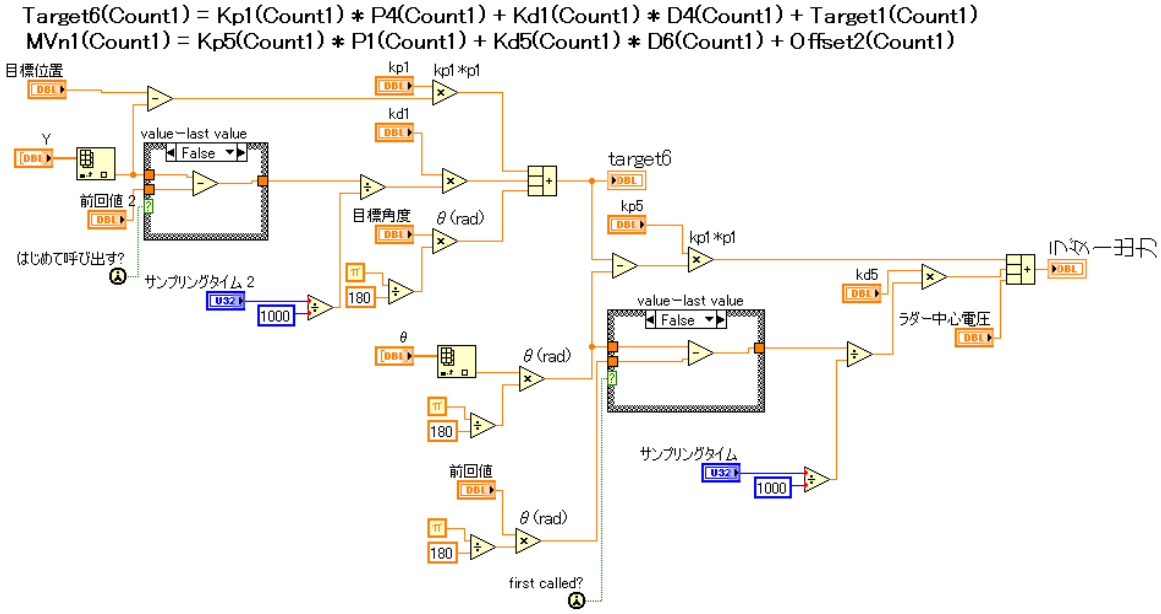


図 18-3　ラダー制御電圧の遅れ補正なしの計算プログラム

図 18-4　ラダー制御電圧の遅れ補正ありの計算プログラム

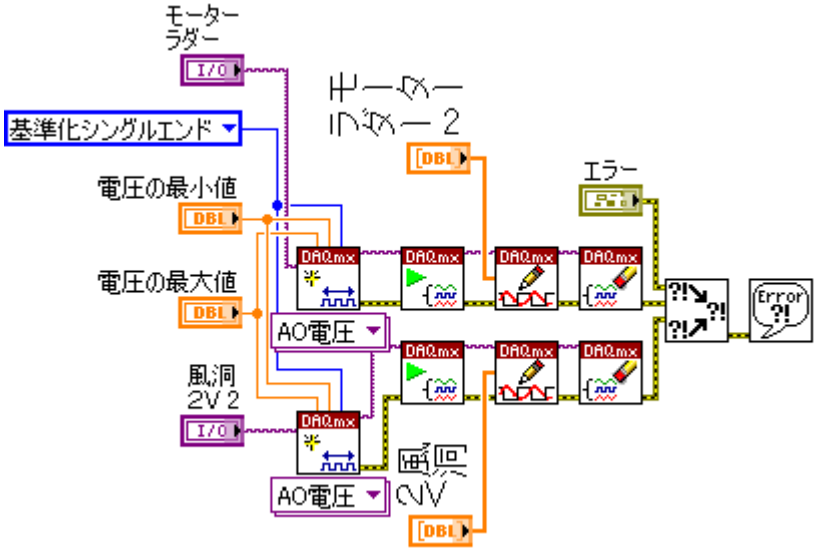


図 19　D/A 変換器へ出力するためのプログラム

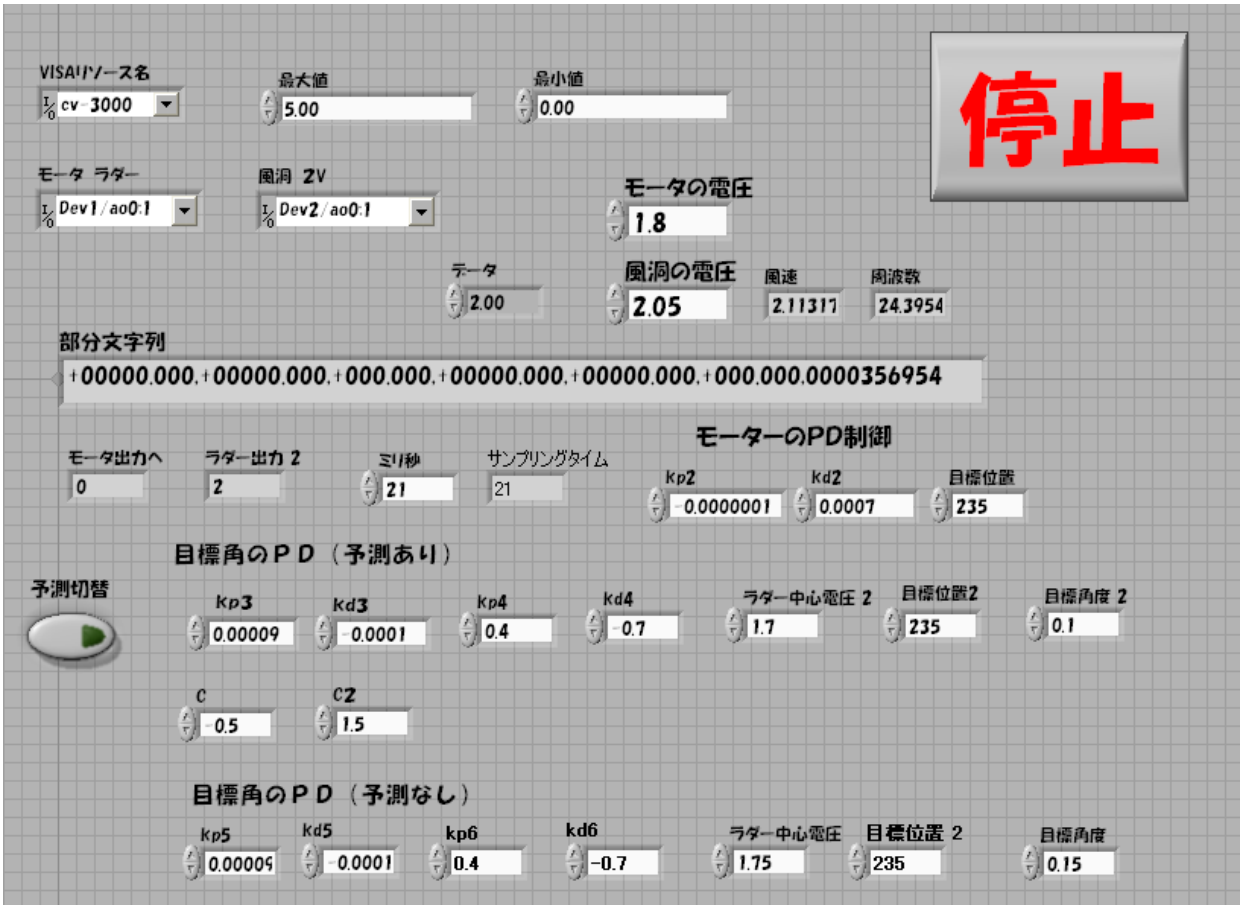


図 20　飛行制御プログラムのフロントパネル

第 4 章 飛行体位置制御式の構築

4.1 飛行体のヨー角制御

機体の上面に取り付けてあるマーカーより機体のヨー角を検出し PD 制御を用いて常に機体の角度を一定に保てるように制御を行った。制御式は

$$V = K_{\mathrm{p}\theta}(\theta_{\mathrm{ta}} - \theta_n) + K_{\mathrm{d}\theta}\left[\frac{\theta_n - \theta_{n-1}}{T_{\mathrm{s}}}\right] \tag{4.1}$$

となっており、ここでVはラダーにかける電圧、$\theta_{\mathbf{ta}}$が機体の目標角、$\theta_{\boldsymbol{n}}$が現在の機体角、θ_{n-1}が現在の機体角の 1 つ前の機体角、T_{s}がサンプリングタイムである。

4.2 位置制御の構築

ヨー角制御のみでは、機体の位置は受動的に決まってしまい制御することができない。よって制御式に Y 座標を組み込み機体の位置を制御することにした。昨年度は機体の位置を制御するためにヨー角制御に位置制御を足し合わせた制御式である

$$V = K_{\mathrm{pY}}(Y_{\mathrm{ta}} - Y_n) + K_{\mathrm{dY}}\left[\frac{Y_n - Y_{n-1}}{T_{\mathrm{s}}}\right] + K_{\mathrm{p}\theta}(\theta_{\mathrm{ta}} - \theta_n) + K_{\mathrm{d}\theta}\left[\frac{\theta_n - \theta_{n-1}}{T_{\mathrm{s}}}\right] \tag{4.2}$$

を用いていた。ここで、Y_{ta}が機体の目標位置、Y_nが現在の機体位置、Y_{n-1}が現在の機体位置の 1 つ前の機体位置である。

しかし、図 21 の実験結果を見て分かるように機体の位置を目標位置に安定飛行させるのが難しかった。

原因として、この制御式では 2 つの PD 制御が混同しているので、位置制御部分の制御ゲインを限りなく小さな値にしなければ飛行することができず、ほとんどヨー角制御と変わらない飛行になっていたことが考えられる。

そこでこの位置制御式を見直し、ヨー角制御部分の目標角に Y 座標を組み込むことで目標角を常に変更しつつ位置を制御するといった制御方法を行った。制御式は

$$V = K_{\mathrm{p}\theta}(\theta_{ta} - \theta_n) + K_{\mathrm{d}\theta}\left(\frac{\theta_n - \theta_{n-1}}{T_s}\right) \tag{4.3}$$

である。目標角 θ_{ta} の導き出す式は、

$$\theta_{\mathrm{ta}} = K_{\mathrm{pY}}(Y_{\mathrm{ta}} - Y_{\mathrm{n}}) + K_{\mathrm{dY}}\left(\frac{Y_{\mathrm{n}} - Y_{\mathrm{n}-1}}{T_s}\right) \tag{4.4}$$

となっている。この位置制御式に変更することで飛行中に機体の位置を変更することが可能となった。図 22 では飛行中に機体の目標位置を変えて機体の位置をコントロールした実験結果を示す。

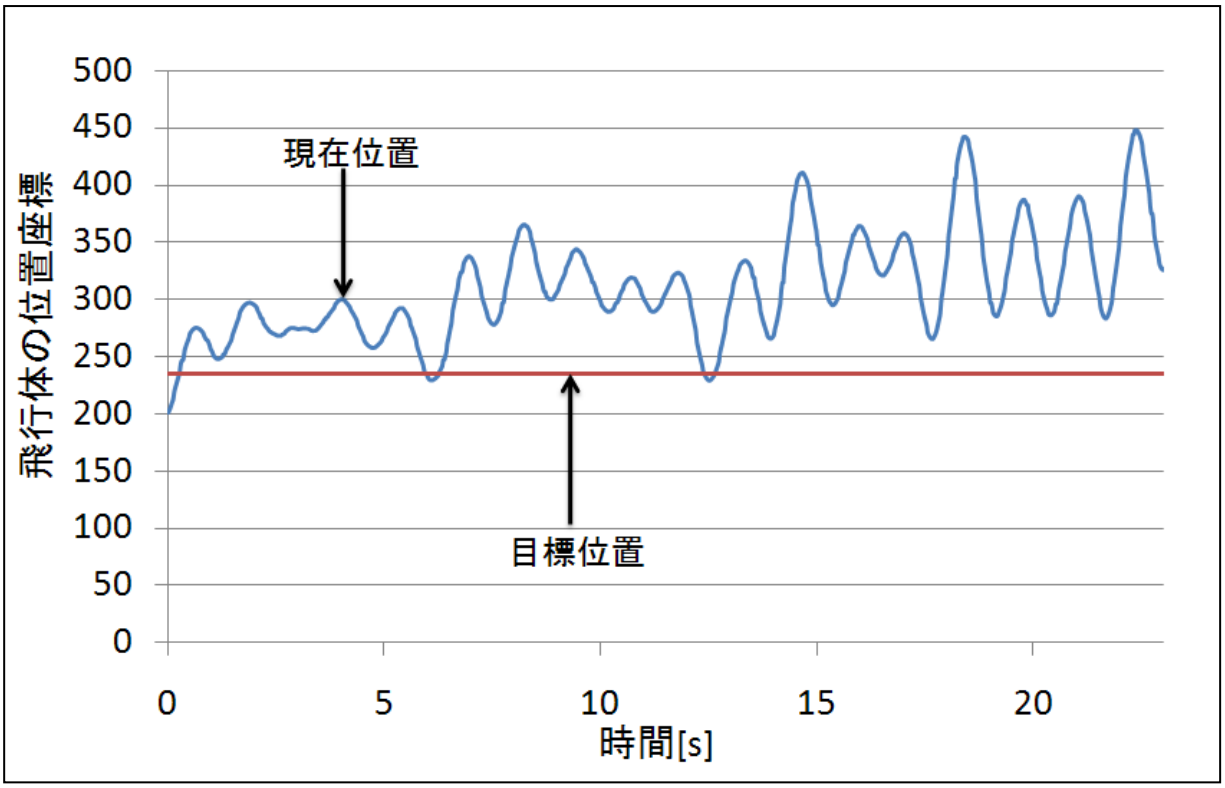


図 21　昨年度の位置制御結果

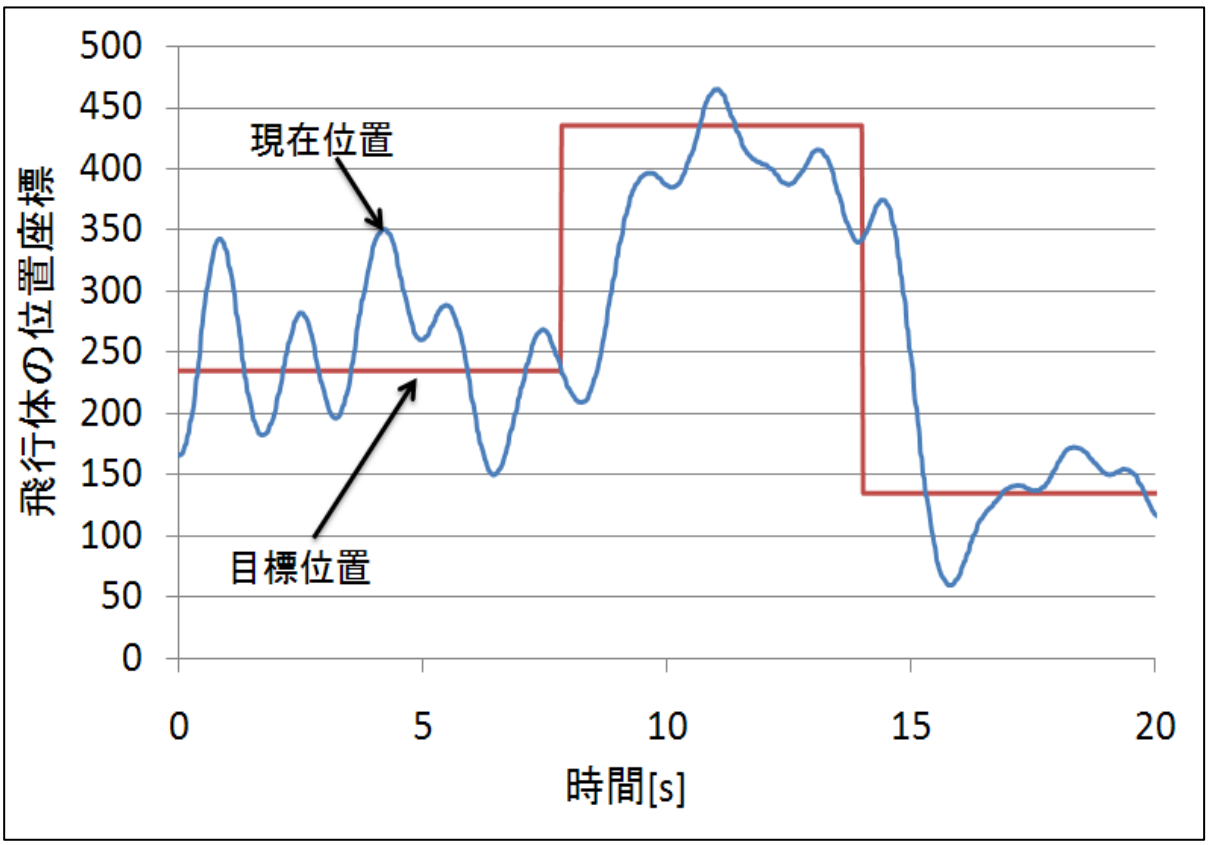


図 22　飛行中の機体のステップ応答

第 5 章 制御システムの遅れ

5.1 飛行システム全体の遅れ

現在の飛行制御システムでは画像処理装置から送られてくるヨー角と位置座標をもとにフィードバック制御を行っている。しかしこの情報に遅れが生じた場合、古い情報をもとに制御を行ってしまうので制御に悪影響を及ぼしてしまう。

そこで今回制御システムにどれほどの遅れがあるのか調査を行った。実験はまず制御システム全体の遅れを確認するため、機体を固定した状態でラダーにマーカーを付け、PC から信号を送ってラダーを動かし、ラダーが動いた情報を PC が受け取るまでにどれほどの時間がかかるのかをステップ応答により調べた。実験の結果を図 23 に示す。実験の結果 100ms の遅れが確認できた。つまり、現在の制御システムは図 24 に示す通り 100ms 過去の値を用いてフィードバック制御を行っているということである。

また、この遅れがどこで発生しているのかを確認するための予備実験を行った。

5.2 画像処理部での遅れ

CCD カメラがマーカーを読み、画像処理装置を通して PC に送られてくるまでの画像処理部での遅れがどれほどあるのかの確認を行った。

実験は固定した乾電池にマーカーを付け、乾電池からの信号を PC で D/A 変換器を通して受け取り、乾電池に衝撃を与えて弾き飛ばし、乾電池と D/A 変換器との接点が断たれ信号がとぎれるのと乾電池が動き出す間の遅れを調べた。実験の結果を図 25 に示す。実験は数十回行い、結果 75ms の遅れが確認できた。

5.3 コントロール部での遅れ

実験に使用した飛行体のラダーの切り替えにはコイルアクチュエータを使用している、この間にどれほどの遅れが発生しているのか、ファンクションジェネレータとレーザー変位計とオシロスコープを用いて計測した。

実験の結果、信号を出しラダーが反応するまでに発生している遅れは 25ms ほどであった。この遅れはコイルに電流が流れて、磁石が吸い寄せられるまでに起きているものと考えられる。結果を図 26 に示す。

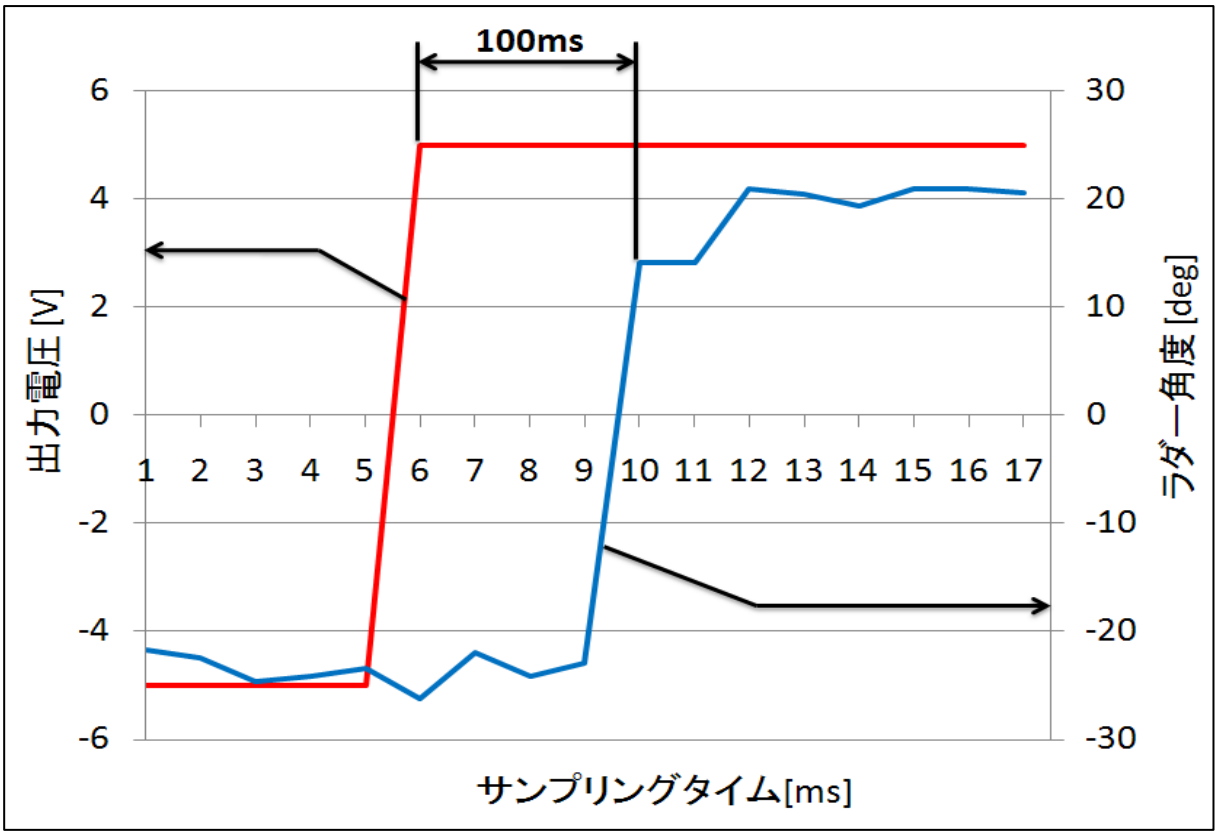


図 23　飛行システム全体の遅れ

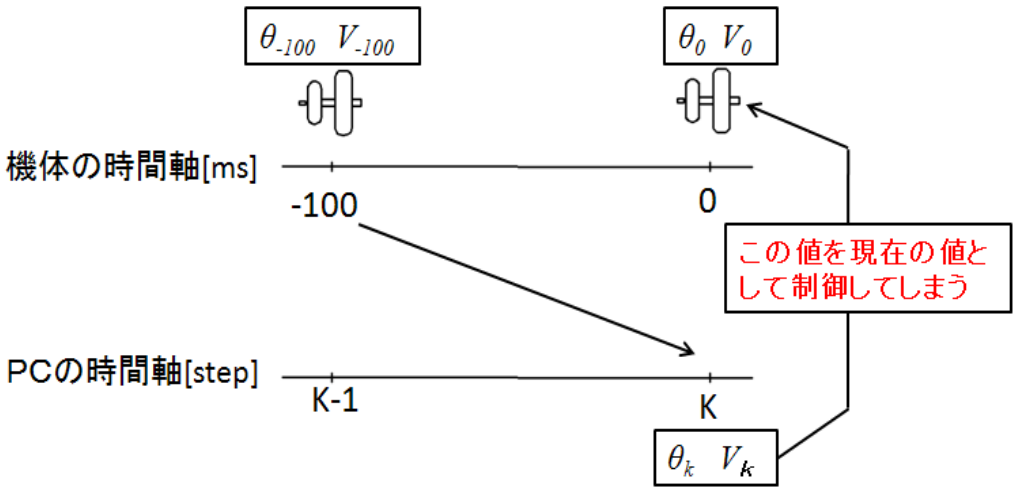


図 24　遅れ制御の様子

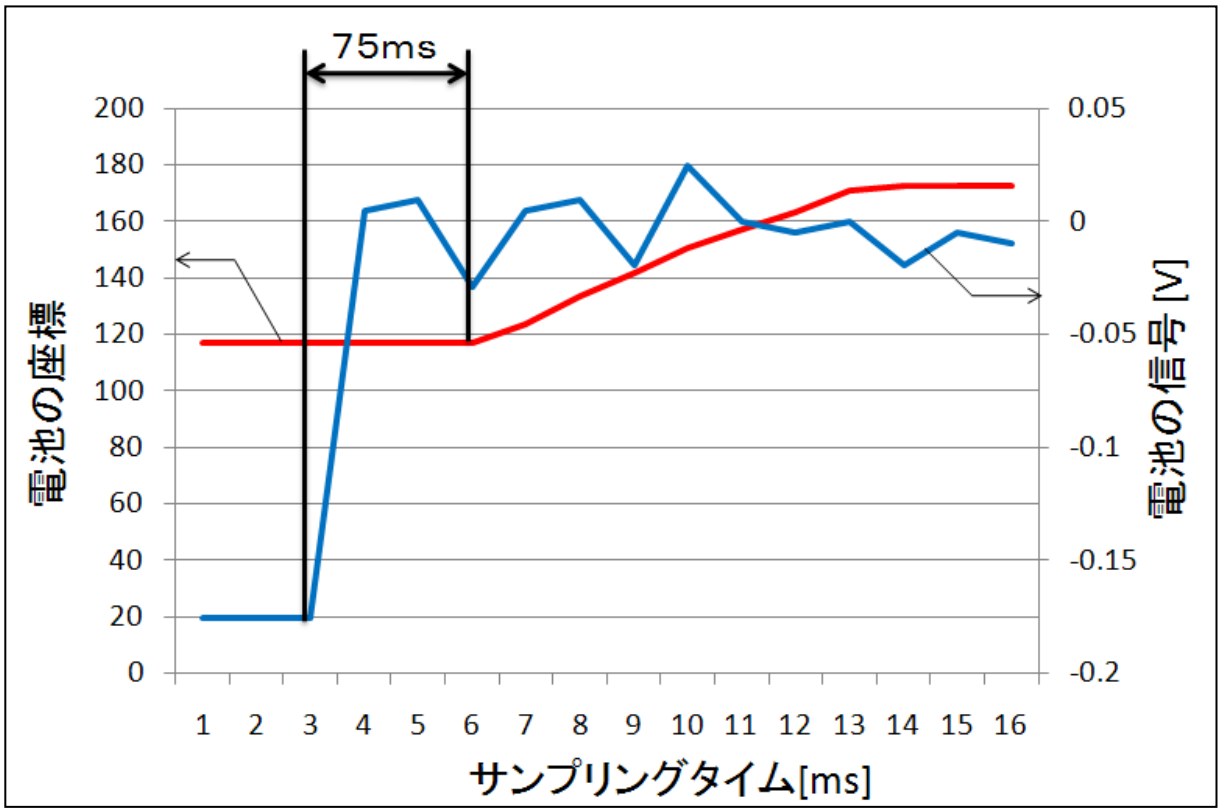


図 25　観測装置の遅れ

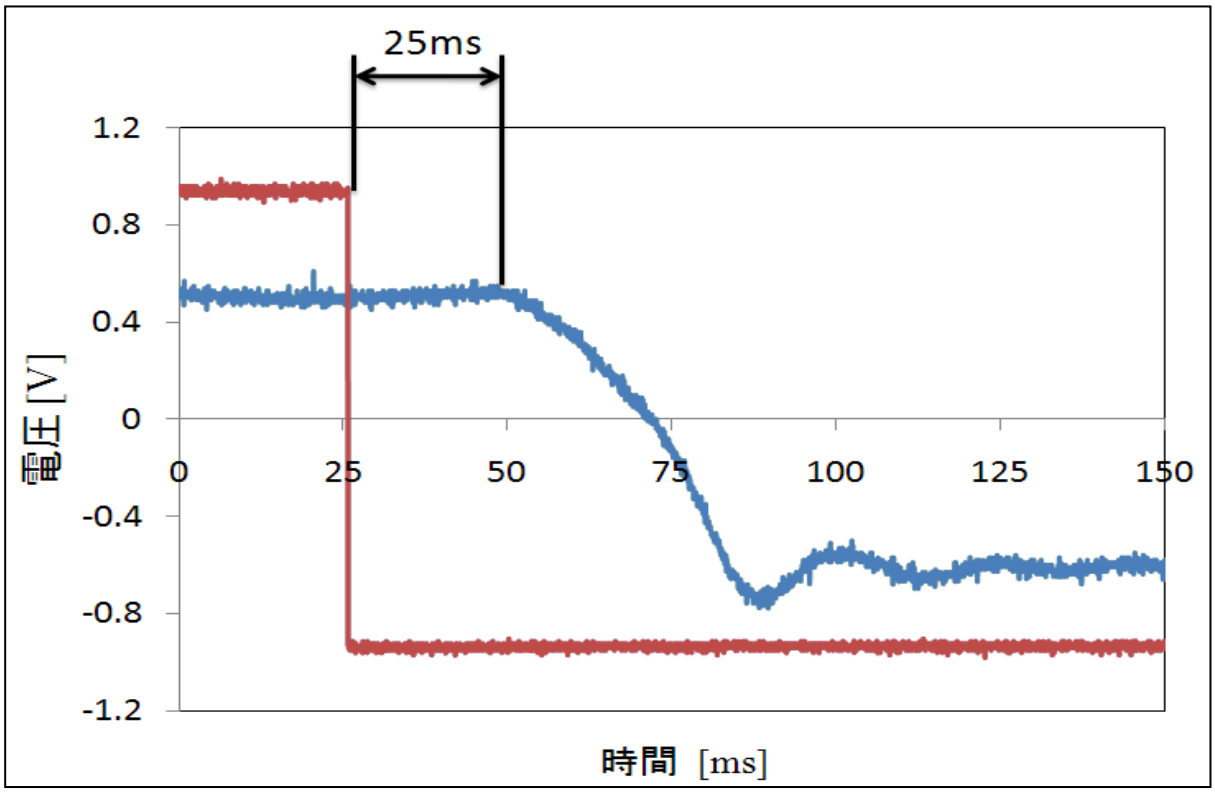


図 26　機体の反応遅れ

第 6 章 遅れ補正による機体のヨー角と位置座標の状態推定

6.1 遅れ補正式の概要

前章の実験より現在の飛行制御プログラムに 100ms の遅れがあることが分かった。よってさらなる飛行精度の向上にために、この遅れの補正を行うことにした。

現在の制御システムは離散時間系であり、サンプリングタイム間の電圧は常に一定である。よって運動方程式から過去の値をもとに現在のヨー角と位置座標の推定値を算出しその値をもとに制御を行った。

6.2 機体のヨー角の遅れ補正式の構築

ラダーによる機体のトルクが、機体の慣性モーメントと機体の加速度との比例関係にあることから

$$J\ddot{\theta} = \tau V \tag{6.1}$$

の式が成り立つ。ここで機体の角加速度が$\ddot{\theta}$、機体の慣性モーメントが J、1 ボルトあたりのトルクがτ、電圧がVである。この式を展開して

$$\ddot{\theta} = \frac{\tau}{J} V \tag{6.2}$$

とする。ここでτ/JをC_θとおき、C_θについては過去の実際に飛行したデータから同定する。

また電圧 V は過去のヨー角をもとに出力する値なので$V_{\mathrm{k}-1}$とし、そこから算出される機体の角加速度は現在の推定値であるので$\hat{\ddot{\theta}}$とする。よって

$$\hat{\ddot{\theta}} = \mathrm{C}_\theta V_{\mathrm{k}-1} \tag{6.3}$$

とし、この式を２階積分することで現在のヨー角の推定値を算出する積分過程は

$$\hat{\dot{\theta}} = \mathrm{C}_\theta \mathrm{V}_{\mathrm{k}-1} T_s + \frac{\theta_k - \theta_{k-1}}{T_s} \tag{6.4}$$

$$\hat{\theta} = \frac{1}{2} \mathrm{C}_\theta \mathrm{V}_{\mathrm{k}-1} {T_s}^2 + \frac{\theta_k - \theta_{k-1}}{T_s} T_s + \theta_k \tag{6.5}$$

となっており、ここで$\hat{\theta}$が現在のヨー角の推定値、T_sがサンプリングタイムである。

6.3 過去の飛行データによるC_θの同定

式(6.5)で用いているC_θについては過去の飛行したデータから、実際にラダーに電圧をかけて機体のヨー角がどのように変化していったかが分かるので同定を行った。サンプリングタイムごとに保存されているヨー角$\theta_0 \rightarrow \theta_1 \rightarrow \theta_2 \rightarrow \cdots \rightarrow \theta_{\mathrm{n}}$ があり、それぞれに式(6.5)を適用すると、

$$\theta_1 = \frac{1}{2}\mathrm{C}_\theta V_0 {T_s}^2 + \dot{\theta}_0 T_s + \theta_0 \quad (6.6)$$

$$\theta_2 = \frac{1}{2}\mathrm{C}_\theta V_1 {T_s}^2 + \dot{\theta}_1 T_s + \theta_1 \quad (6.7)$$

.
.
.

となる。ここで式(6.7)に

$$\dot{\theta}_1 = \mathrm{C}_\theta V_0 T_s + \dot{\theta}_0 \quad (6.8)$$

を代入して

$$\theta_2 = \frac{1}{2}\mathrm{C}_\theta V_1 {T_s}^2 + (\mathrm{C}_\theta V_0 T_s + \dot{\theta}_0) T_s + \theta_1 \quad (6.9)$$

を得る。式(9)に式(6)を展開した

$$\dot{\theta}_0 = \frac{\theta_1 - \frac{1}{2} C_\theta V_0 {T_s}^2 - \theta_0}{T_s} \quad (6.10)$$

を代入し

$$\theta_2 = \frac{1}{2} C_\theta V_1 {T_s}^2 + \left(C_\theta V_0 T_s + \left(C_\theta V_0 T_s + \frac{\theta_1 - \frac{1}{2} C_\theta V_0 {T_s}^2 - \theta_0}{T_s}\right) T_s + \theta_1 \right. \quad (6.11)$$

を得る。この式(11)を展開して

$$\left\{\frac{1}{2}(V_1 + V_0) T_s\right\} C_\theta = 2\theta_2 - 2\theta_1 + \theta_0 \quad (6.12)$$

とする。ここで、

$$\left\{\frac{1}{2}(V_1 + V_0) T_s\right\} = \alpha \quad (6.13)$$

$$2\theta_2 - 2\theta_1 + \theta_0 = \beta \quad (6.14)$$

とおき、式(12)を

$$\alpha \mathrm{C}_\theta = \beta \quad (6.15)$$

とし、C_θについて展開して

$$\mathrm{C}_\theta = \frac{\beta}{\alpha} \quad (6.16)$$

と変形する。このC_θについて

$$\mathrm{C}_{\theta 1} = \frac{\beta_1}{\alpha_1}$$

$$\mathrm{C}_{\theta 2} = \frac{\beta_2}{\alpha_2}$$

$$\mathrm{C}_{\theta 3} = \frac{\beta_3}{\alpha_3}$$

.
.
.

$$\mathrm{C}_{\theta \mathrm{n}} = \frac{\beta_n}{\alpha_n}$$

と各サンプリングタイム間の値を計算し平均を算出した。

6.4 機体の位置（Y 座標）の遅れ補正式の構築

位置座標つまり Y 軸の値の状態推定に関して、運動方程式より物体の力が質量と加速度との比例関係にあることから、ラダーによる機体の横方向の力が機体の質量と Y 軸の加速度との比例関係にあることがいえる。この関係により

$$F_R\theta = m\hat{\ddot{Y}} \tag{6.17}$$

が成り立つ。ここで、$F_R\theta$がラダーによる機体の横方向の力、$\hat{\ddot{Y}}$が機体の Y 軸の加速度の推定値、θ が機体のヨー角である。この式(6.17)を展開して

$$\hat{\ddot{Y}} = \frac{m}{F_R}\theta \tag{6.18}$$

とする。またここで$\mathrm{m}/F_R = \mathrm{C_Y}$とおき式(6.18)を

$$\hat{\ddot{Y}} = \mathrm{C_Y}\theta \tag{6.19}$$

とし、２階積分することで現在の Y 座標の推定値を算出する積分過程は

$$\dot{Y} = \mathrm{C_Y}\theta_k T_s + \frac{Y_k - Y_{k-1}}{T_s} \tag{6.20}$$

$$\hat{Y} = \frac{1}{2}\mathrm{C_Y}\theta_\mathrm{k}{T_s}^2 + \frac{Y_k - Y_{k-1}}{T_s}T_s + Y_k \tag{6.21}$$

となっており、ここで$\hat{Y}$が現在の Y 座標の推定値、T_sがサンプリングタイムである。

6.5 過去の飛行データによる$\mathrm{C_Y}$の同定

式(6.21)で用いている$\mathrm{C_Y}$についても 6.3 同様に過去の飛行したデータから、実際にラダーに電圧をかけて機体の Y 座標がどのように変化していったかが分かるので同定を行った。サンプリングタイムごとに保存されている Y 座標 $Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \cdots \rightarrow Y_n$ があり、それぞれに式(6.21)を適用すると

$$Y_1 = \frac{1}{2}\mathrm{C_Y}\theta_0{T_s}^2 + \dot{Y}_0 T_s + Y_0 \tag{6.22}$$

$$Y_2 = \frac{1}{2}\mathrm{C_Y}\theta_1{T_s}^2 + \dot{Y}_1 T_s + Y_1 \tag{6.23}$$

・
・

となる。ここで式(6.23)に

$$\dot{Y}_1 = \mathrm{C_Y}\theta_0 T_s + \dot{Y}_0 \tag{6.24}$$

を代入して

$$Y_2 = \frac{1}{2}\mathrm{C_Y}\theta_1 {T_s}^2 + (\mathrm{C_Y}\theta_0 T_s + \dot{Y}_0)T_s + Y_1 \tag{6.25}$$

を得る。式(6.25)に式(6.22)を展開した

$$\dot{Y}_0 = \frac{Y_1 - \frac{1}{2}\mathrm{C_Y}\theta_0 {T_s}^2 - Y_0}{T_s} \tag{6.26}$$

を代入し

$$\mathrm{Y_2} = \frac{1}{2}\mathrm{C_Y}\theta_1 {T_s}^2 + \left(\mathrm{C_Y}\theta_0 T_s + \left(\mathrm{C_Y}\theta_0 T_s + \frac{Y_1 - \frac{1}{2}\mathrm{C_Y}\theta_1 {T_s}^2 - Y_0}{T_s}\right)T_s + Y_1\right. \tag{6.27}$$

を得る。この式(6.27)を展開して

$$\left\{\frac{1}{2}(\theta_1 + \theta_0)T_s\right\}\mathrm{C_Y} = 2\mathrm{Y_2} - 2Y_1 + Y_0 \tag{6.28}$$

とする。ここで、

$$\left\{\frac{1}{2}(\theta_1 + \theta_0)T_s\right\} = \alpha \tag{6.29}$$

$$2\mathrm{Y_2} - 2Y_1 + Y_0 = \beta \tag{6.30}$$

とおき、式(6.28)を

$$\alpha\mathrm{C_Y} = \beta \tag{6.31}$$

とし、$\mathrm{C_Y}$について展開して

$$\mathrm{C_Y} = \frac{\beta}{\alpha} \tag{6.32}$$

と変形する。この$\mathrm{C_Y}$について

$$\mathrm{C_{Y1}} = \frac{\beta_1}{\alpha_1}$$

$$\mathrm{C_{Y2}} = \frac{\beta_2}{\alpha_2}$$

$$\mathrm{C_{Y3}} = \frac{\beta_3}{\alpha_3}$$

・
・
・

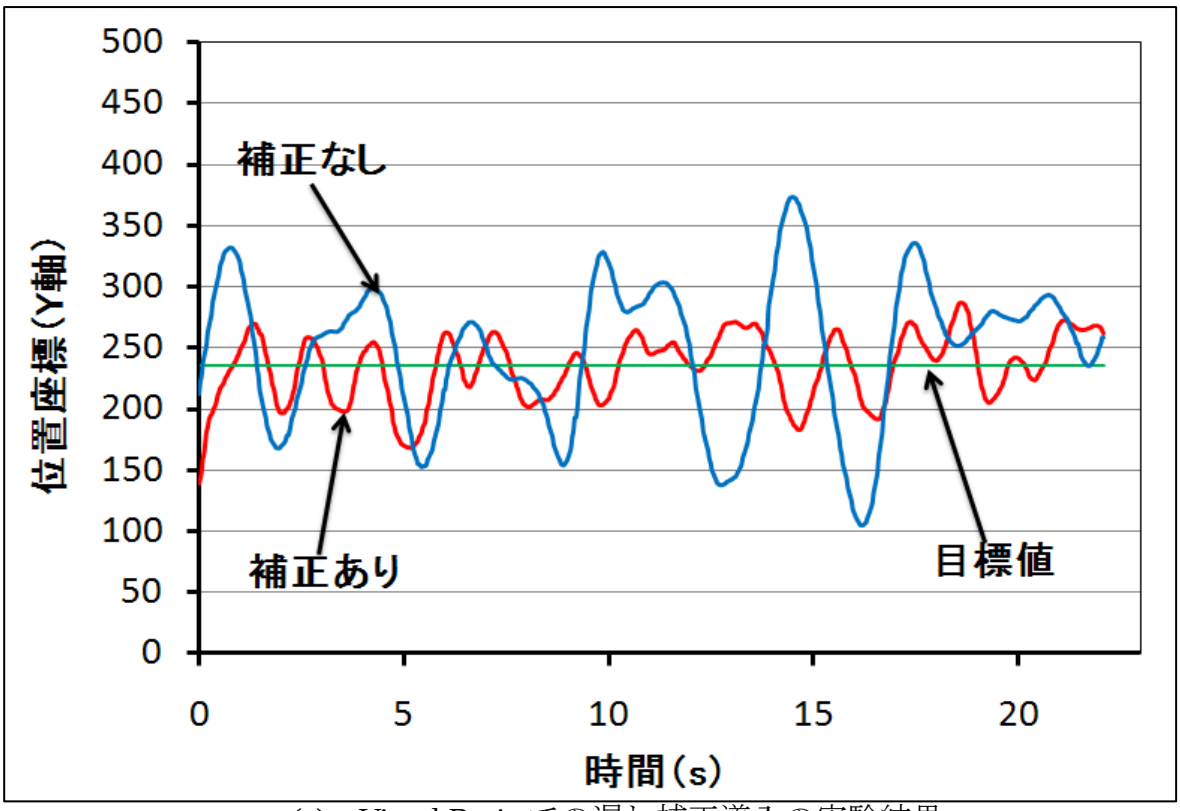
$$\mathrm{C_{Yn}} = \frac{\beta_n}{\alpha_n}$$

と各サンプリングタイム間の値を計算し平均を算出した。

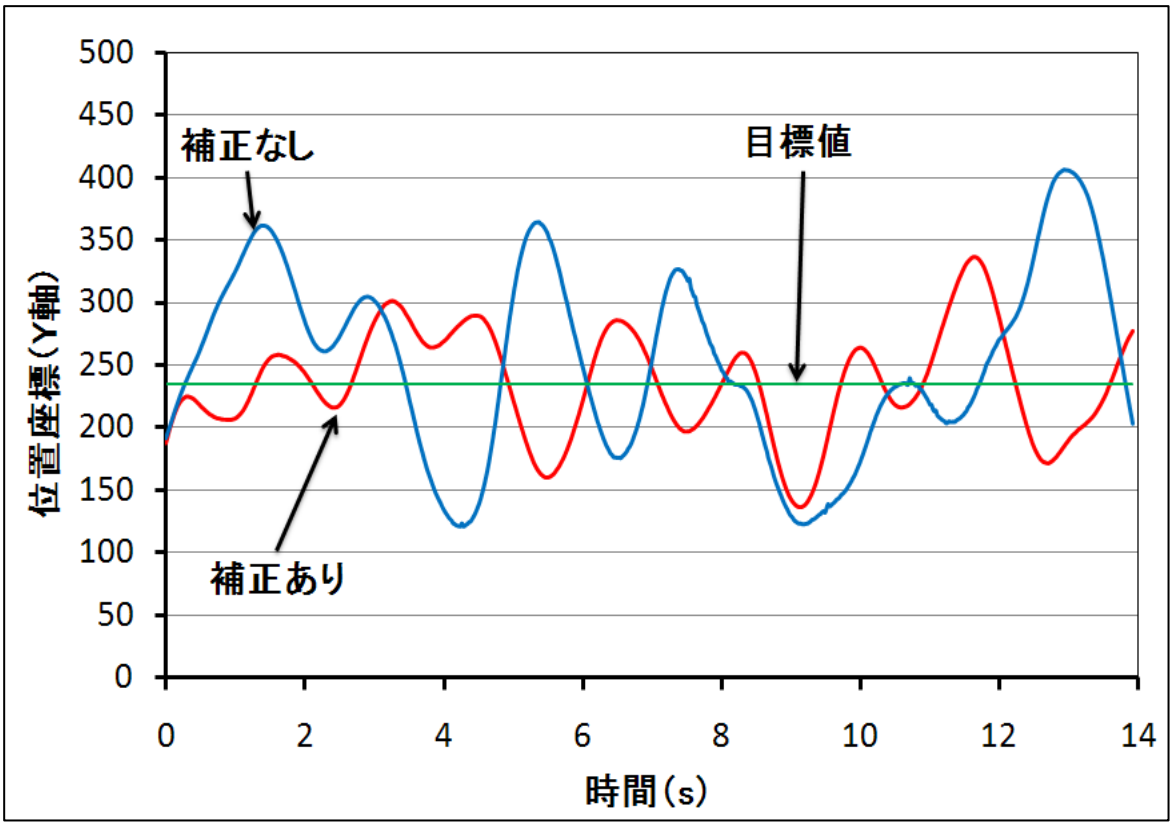
6.6 ヨー角と Y 座標の遅れ補正導入による飛行実験結果

実験は同じ機体で補正なしとありの場合で 10 回行いそれぞれの標準偏差の平均を計算した。その結果 Visual Basic を用いた場合、補正なしの場合 44.2 で、補正ありの場合 32.6 となった。

LabVIEW に関しても遅れ補正式を導入することで、補正なしが 51.8 、補正ありが 46.8 の標準偏差となり遅れ補正を導入することで目標位置に対しての安定度が向上した。実験の結果を図 27 に示す。



(a)　Visual Basic での遅れ補正導入の実験結果



(b)　LabVIEW での遅れ補正導入の実験結果

図 27　遅れ補正導入の実験結果

第 7 章 画像処理部の変更による遅れの改善

7.1 デジタル I/O による通信速度向上の検証

LabVIEW にて、Ethernet 接続を行っていたが遅れが大幅に発生したため、デジタル I/O 接続を行った。デジタル I/O 接続は KEYENCE 社専用パラレル接続ケーブル（OP-51657）を使い通信させるケーブルであり、特徴としては 10 進数で送られてくるデータを 2 進数にし、上位バイトと下位バイトに分け、送るものである。

デジタル I/O を図 28 に示す。画像処理装置内で 2 進数から 10 進数に変換する手間が省かれるため、USB より速い。デジタル I/O 接続は 40 ピンの接続ができるが今回はデータの取り込みと接続するのが D/A 変換器のため、接続が限られる。よって 24~40 ピンだけを使うことにする。2 台の D/A 変換器を使い、8 つずつ分け接続を行った。送られてきた上位バイトと下位バイトの 2 進数を結合させ、10 進数に変換することで正規のデータとなる。だが X、Y だけで通信させた結果、不規則に同じ値を出すため、計算に用いることはできない。この問題を表 1 に示す。改善を試みたが問題解決までは至らなかった。

7.2 CV-5000 導入による実験結果

遅れの確認の予備実験により画像処理部に 75ms の遅れがあることが分かった。現在の制御システムでは画像処理装置に KEYENCE 製の CV-3000 を用いている。

今回この遅れを改善するために CV-3000 の後継機である CV-5000 (図 29) を用いて遅れの確認の予備実験を行った。実験を数十回行った結果、遅れが 50ms となり 1 ステップ速くなっていることが分かった。これは、CV-3000 のトリガ間隔が 21ms なのに対して CV-5000 のトリガ間隔が 10ms と向上しているからではないかと考えられる。

また機体を固定したうえでマーカーの読み取り精度の誤差を調べたところ標準偏差が CV-3000 は 0.241 で CV-5000 が 0.079 と向上していることが分かった。

7.3 シーケンサ導入による通信速度向上の実験

現在の制御システムは画像処理装置からの計測値を USB から Windows 上の LabVIEW または Visual Basic のプログラムで取り込み再び USB を通して D/A 変換器に送っている。

ここでの処理の高速化を図るため今回 KEYENCE 社よりシーケンサを借用して処理速度がどれほど向上するのかの実験を行った。借用したシーケンサである KV-5000 (図 31) は D/A ユニット、A/D ユニットが装着可能であり、この KV-5000 ひとつで今まで Windows 上で行っていた処理がすべて可能となった。

実験の結果、従来の制御システムでは最大サンプリング速度が 25ms であったのに対して KV-5000 導入により最大サンプリング速度が 0.3ms と大幅に向上することが分かった。

5.3 節同様の観測の遅れの確認実験を行ったところ、従来のシステムで 75ms あった遅れがシーケンサの導入で CV-3000 とのリンクの場合 43ms で CV-5000 とのリンクの場合 30ms になることが分かった。実験の結果を図 32 に示す。

図 28　デジタル I/O ケーブル

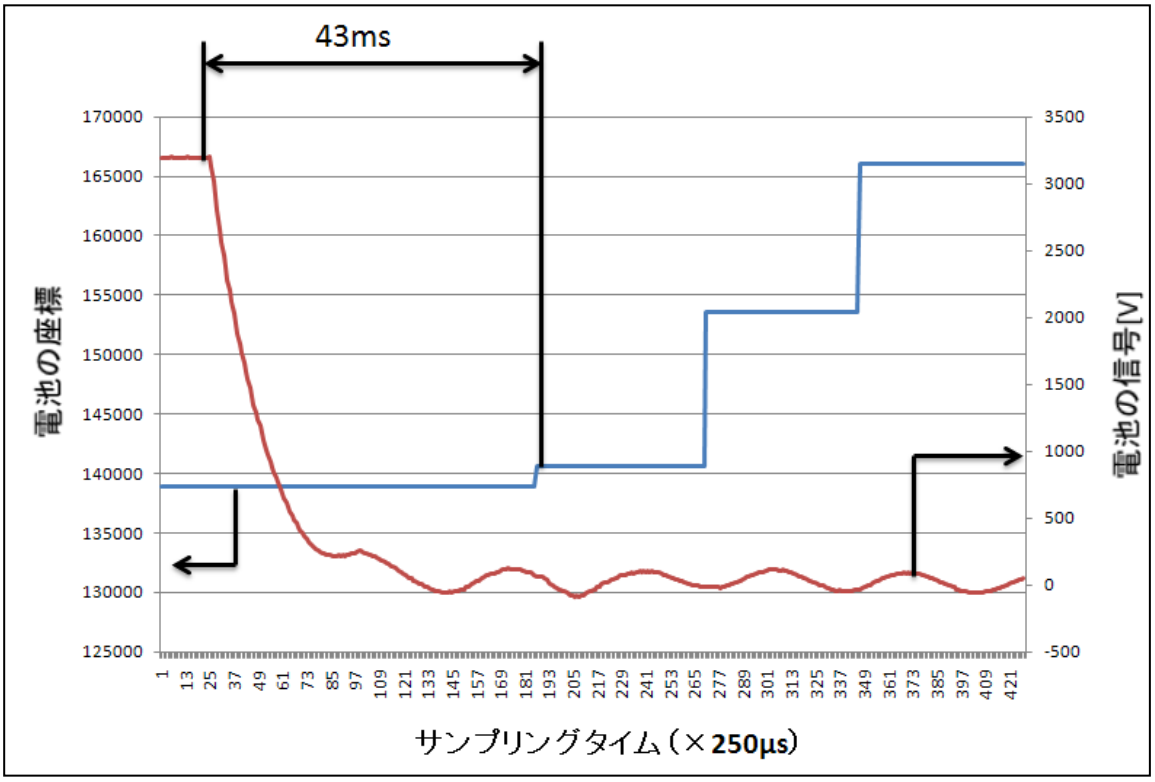
表１　デジタル I/O の問題点

座標データ	下位１６ビット	上位１６ビット
177502	46430	2
177389	46317	2
210620	14012	3
210512	13904	3

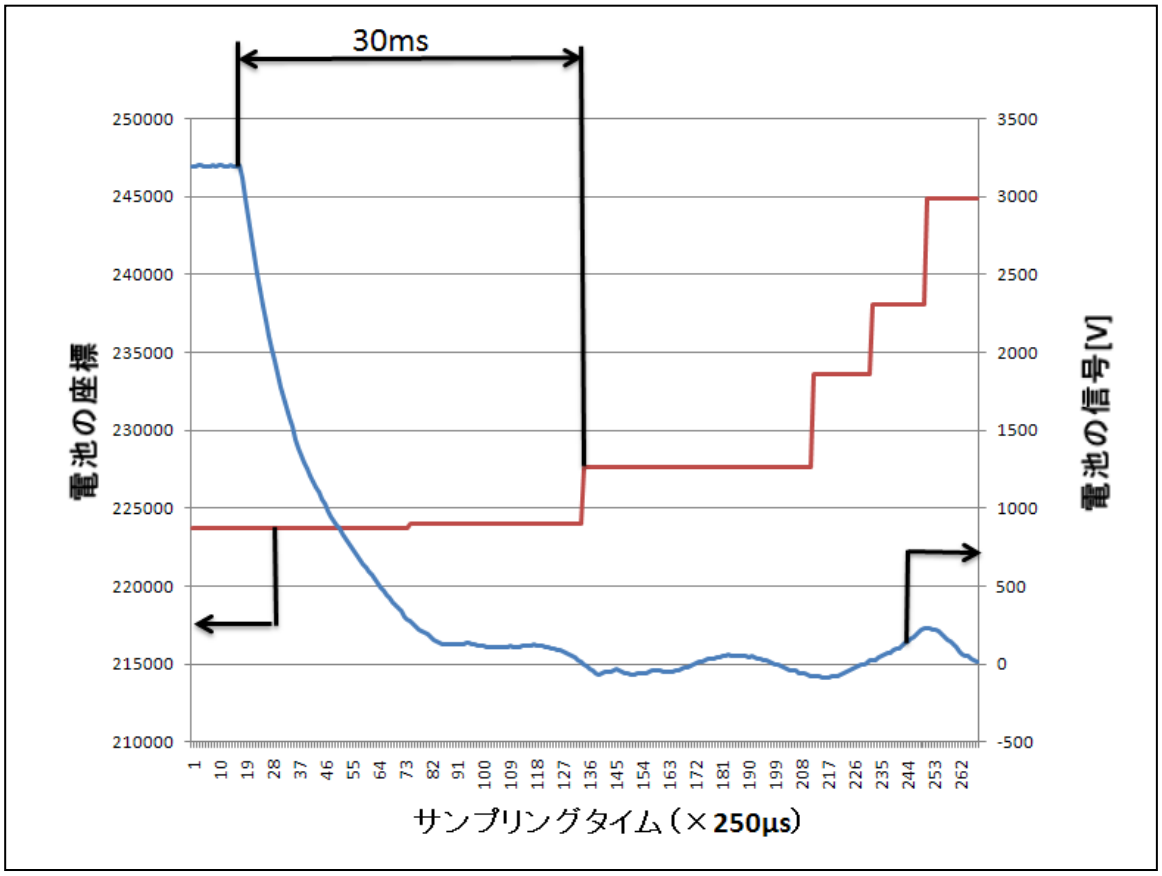
図 29　画像処理装置（CV-5000）

図 30　CCD カメラ

図 31　画像処理装置（CV-5000）



(a)　シーケンサと CV-3000 のリンク時の遅れ



(b)　CV-5000 のリンク時の遅れ

図 32　シーケンサと画像処理装置のリンク時の遅れ

第 8 章 結言

今回、位置制御式の見直しと遅れ補正の導入により飛行精度の向上に成功した。また、LabVIEW を用いた位置制御プログラムを完成させ、遅れ補正を行い飛行させる事ができた。

今後の課題として二枚翼羽ばたき飛行機を制作し、実験に導入することで飛行時における重心の変化や翼のトルクを計測し羽ばたき飛翔体の飛行メカニズムを解明すること。シーケンサによる高速処理ができる飛行制御プログラムを完成させ、さらなる飛行精度向上を目指したいと検討している。

LabVIEW に関してもシステム内の遅れをプログラムの見直しとデジタル **I/O** の導入により改善することを考えている。

謝辞

本研究を進めていくうえで、多大なご指導ご指摘をして頂いた。河村良行教授に深く感謝の意を表します。

情報システム工学科の辻輝生教授に飛行体の位置制御式の見直し、遅れ補正による状態推定のアルゴリズムについて多大なご指導をして頂いたことを、深く感謝いたします。

プログラミング、飛行実験、機体の改良を通してお世話になった指導院生の柴田光紀先輩に深く感謝いたします。

最後に同じ研究室で 1 年間研究をともにし、協力し合った河村研究室卒研生全員に深く感謝の意を表します。

付録

付録 1 飛行制御プログラム（Visual Basic）

付録 2 制御パネル（Visual Basic）

付録 3 飛行制御プログラム（LabVIEW）

付録 4 データ保存プログラム（LabVIEW）

付録 4 飛行制御プログラム
（KV-5000 ラダープログラム）

付録 1 飛行制御プログラム（Visual Basic）

```
Public Class 飛行機制御

    Private stopWatch As Diagnostics.Stopwatch                    '時間計測の為の宣言文
    Dim Buffer As Object                                          '受信バッファの宣言
    Dim Time1(100000), Time2(100000), Time5(100000) As Single     '時間定義の為の変数
    Dim X_1(100000), Y_1(100000), Angle1(100000), X_2(100000), Y_2(100000), Angle2(100000) As Single
    'CV-3000のデータ格納用変数
    Dim Target1(100000), Target2(100000), Target3(100000), Target4(100000), Target5(100000), Target6(100000) As Single
    '制御目標値の変数宣言
    Dim Value1(100000), Value2(100000), Value3(100000), Value1h(100000), Value1hh(100000), Value1hhh(100000), Value1hhhh(100000), Value1hhhhh(100000), Value3h(100000), Value3hh(100000), Value3hhh(100000), Value3hhhh(100000), Value3hhhhh(100000) As Single
    '現在値の変数宣言
    Dim Count1 As String                                          'プログラムの実行回数カウント

    '各電圧オフセット設定、入力電圧、モータ電流・電圧、風洞風速変数宣言
    Dim Offset1(100000), Offset2(100000), Vin1(100000), Vin2(100000) As Single
    Dim Ma(100000), Mv(100000), WS(100000), Count(100000) As Single

    'PID制御値の変数宣言
    Dim Kp1(100000), Kd1(100000), Ki1(100000), C1(100000), C2(100000), Ts1(100000) As Single
'制御値の変数宣言
    Dim Kp2(100000), Kd2(100000), Ki2(100000) As Single           '制御値の変数宣言
    Dim Kp3(100000), Kd3(100000), Ki3(100000) As Single           '制御値の変数宣言
    Dim Kp4(100000), Kd4(100000) As Single                        '制御値の変数宣言
    Dim Kp5(100000), Kd5(100000) As Single                        '制御値の変数宣言
    Dim P1(100000), D1(100000), I1(100000), D4h(100000), D4hh(100000), D4hhh(100000), D4hhhh(100000), D4hhhhh(100000) As Single          '制御値の変数宣言
    Dim P2(100000), D2(100000), I2(100000) As Single              '制御値の変数宣言
    Dim P3(100000), D3(100000) As Single                          '制御値の変数宣言
```

```
    Dim P4(100000), D4(100000) As Single                                    '制御
値の変数宣言
    Dim P5(100000), D5(100000), D6(100000), D6h(100000), D6hh(100000), D6hhh(100000),
D6hhhh(100000), D6hhhhh(100000) As Single                                   '制御
値の変数宣言
    Dim dt, dv1, dv2, dv3, dv4, dv5 As Single                               '制御
値の変数宣言
    Dim MVn1(100000), MVn2(100000), MVn3(100000), MVn4(100000), MVn5(100000), MVn1p(100000),
MVn1pp(100000), MVn1ppp(100000), MVn1pppp(100000) As Single  '制御値の変数宣言
    Dim yaw_angle(100000) As Single                                         '制御
値の変数宣言
    Dim deg1(100000), deg2(100000) As Single                                '制御
値の変数宣言
    Dim v As Single                                                         '制御
値の変数宣言

    '=========================================D/Aボード宣言
========================================
    'ボード宣言
    Dim ULStat As MccDaq.ErrorInfo
    Private DaqBoard0 As MccDaq.MccBoard = New MccDaq.MccBoard(0)           'AD/DA変換機台
目の宣言
    Private DaqBoard1 As MccDaq.MccBoard = New MccDaq.MccBoard(1)           'AD/DA変換機台
目の宣言

    Dim DataValuesIn(16) As System.UInt16                                   'データの構造宣
言
    Dim DataValuesOut(4) As System.UInt16                                   'データの構造宣
言
    Dim EngUnitsin(16) As Single                                            '検出電圧の変数
宣言
    Dim Engunitsout(4) As Single                                            '検出電圧の変数
宣言

    'ポート数設定
    Dim ChanIn1 As Integer = 0      '入力ポート
    Dim ChanIn2 As Integer = 1

    Const Chan1 As Integer = 0      '出力ポート
    Const Chan2 As Integer = 1

    'レンジ設定
    Const Range1 As MccDaq.Range = MccDaq.Range.Bip10Volts                  '入力ポートの
```

```
電圧宣言(±V)
    Const Range As MccDaq.Range = MccDaq.Range.Uni4Volts                    '出力電圧Vにレ
ンジ設定

    '====================================CV-3000コントロール
==========================================

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        'MSCommコントロールの初期設定()
        msSerial.CommPort = 3                                               '通信ポートNo3
指定
        msSerial.Settings = "9600,n,8,1"                                    '通信条件設定
        msSerial.Handshaking = MSCommLib.HandshakeConstants.comNone         'フロー制御無し
        msSerial.RTSEnable = False                                          'RTS制御無し
        msSerial.RThreshold = 1                                             '1バイト受信毎
にOnCommイベント発生
        msSerial.SThreshold = 1                                             '送信バッファ空
でOnCommイベント発生

        stopWatch = New Diagnostics.Stopwatch()                             '時間計測関数の
呼び出し
        stopWatch.Start()

    End Sub

    Private Sub msSerial_OnComm(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles msSerial.OnComm

        'CommEventプロパティに対する処理
        Select Case msSerial.CommEvent                                      'イベントの区分
            Case MSCommLib.OnCommConstants.comEvCD                          '何もしない
            Case MSCommLib.OnCommConstants.comEvCTS
            Case MSCommLib.OnCommConstants.comEvDSR
            Case MSCommLib.OnCommConstants.comEvRing
            Case MSCommLib.OnCommConstants.comEvReceive                     '受信データを表
示
                Buffer = msSerial.Input
                CV3000.Text = Buffer
            Case MSCommLib.OnCommConstants.comEvSend
            Case MSCommLib.OnCommConstants.comEvEOF
        End Select
    End Sub
```

```
#Region "Universal Library Initialization - Expand this region to change error handling, etc."

    Private Sub InitUL()
        Dim ULStat As MccDaq.ErrorInfo

        ' declare revision level of Universal Library


        ULStat = MccDaq.MccService.DeclareRevision(MccDaq.MccService.CurrentRevNum)

        ' Initiate error handling
        '  activating error handling will trap errors like
        '  bad channel numbers and non-configured conditions.
        '  Parameters:
        '    MccDaq.ErrorReporting.PrintAll :all warnings and errors encountered will be printed
        '    MccDaq.ErrorHandling.StopAll   :if any error is encountered, the program will stop

        ULStat = MccDaq.MccService.ErrHandling(MccDaq.ErrorReporting.PrintAll,
MccDaq.ErrorHandling.StopAll)
        If ULStat.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then
            Stop
        End If


    End Sub
#End Region


    Private Sub TextControl()

        '================================================================
        '=====================       テキスト表示          ================
        '================================================================

        '座標
        X1_txt.Text = X_1(Count1)            'カメラのX座標
        Y1_txt.Text = Y_1(Count1)            'カメラのY座標
        X2_txt.Text = X_2(Count1)            'カメラのX座標
        Y2_txt.Text = Y_2(Count1)            'カメラのY座標
        Value4_txt.Text = Y_2(Count1)        'カメラのY座標
```

```
        '角度「deg」
        deg01_txt.Text = Angle1(Count1) * 180 / 3.1415      'カメラの主軸角
        deg02_txt.Text = Angle2(Count1) * 180 / 3.1415      'カメラの主軸角
        yaw_angle(Count1) = deg01_txt.Text                  '現在値*********
        Value1_txt.Text = Value1(Count1) * 180 / 3.1415     'カメラの主軸角
        Value2_txt.Text = Value2(Count1)                    'カメラのX座標
        Value3_txt.Text = Value3(Count1)                    'カメラのY座標

        deg1(Count1) = Angle1(Count1) * 180 / 3.1415 '角度をradからdeg表示し、Excelデータ出力に(制御計算に関係なし)
        deg2(Count1) = Angle2(Count1) * 180 / 3.1415 '角度をradからdeg表示し、Excelデータ出力に(制御計算に関係なし)

        '--------------------電圧出力----------------------
        RudderVoltage_txt.Text = MVn1(Count1)    'ラダー出力
        MotorVoltage_txt.Text = MVn2(Count1)     'モータ出力
        Position_txt.Text = MVn1(Count1)         '位置制御

        '---------------------その他-----------------------
        frequency_txt.Text = WS(Count1)                             '風洞のインバータの周波数
        Time_txt.Text = stopWatch.ElapsedMilliseconds / 1000        'プログラム経過時間
        count_txt.Text = Count1                                     'プログラム実行回数

        '---------------------時間指定-----------------------




    End Sub

    Private Sub InputGain()

        '=================================================================
        '=====================       ゲイン入力             ================
        '=================================================================

        '------------------ラダーゲイン入力-----------------------
        Kp1(Count1) = Val(Kp1_txt.Text) 'PID係数
        Ki1(Count1) = Val(Ki1_txt.Text)
        Kd1(Count1) = Val(Kd1_txt.Text)

        '------------------モーターゲイン入力---------------------
        Kp2(Count1) = Val(Kp2_txt.Text) '比例
```

```
    Ki2(Count1) = Val(Ki2_txt.Text) '積分
    Kd2(Count1) = Val(Kd2_txt.Text) '微分

    '------------------位置制御ゲイン入力----------------------
    Kp3(Count1) = Val(Kp3_txt.Text) '比例
    Kd3(Count1) = Val(Kd3_txt.Text) '微分

    '-----------------位置制御時のモータ入力--------------------
    Kp4(Count1) = Val(Kp4_txt.Text) '比例
    Kd4(Count1) = Val(Kd4_txt.Text) '微分

    '----------------風洞制御ゲイン入力--------------------
    Kp5(Count1) = Val(Kp5_txt.Text) '比例
    Kd5(Count1) = Val(Kd5_txt.Text) '微分

    '----------------------現在値-----------------------------
    Value1(Count1) = Angle1(Count1)
    Value2(Count1) = X_1(Count1) '進行方向
    Value3(Count1) = Y_1(Count1) 'ヨー方向

    Offset1(Count1) = Val(Offset1_txt.Text)             'モータのオフセット電圧
    Offset2(Count1) = Val(Angle_adjustment_txt.Text)    'ラダー角の角度調整[V]*

    '------------------制御目標座標----------------------------
    Target1(Count1) = Val(kakudoTarget_txt.Text)        '角度
    Target2(Count1) = Val(Target2_txt.Text)             '進行方向
    Target3(Count1) = Val(Target3_txt.Text)             'ヨー方向
    Target4(Count1) = Val(Target4_txt.Text)             '上下方向
    Target5(Count1) = Val(kakudoTarget_txt.Text)        'Y方向目標値

    C1(Count1) = Val(c1_txt.Text)
    C2(Count1) = Val(c2_txt.Text)
    Ts1(Count1) = Val(Ts1_txt.Text)

 End Sub

Private Sub InputValue()

    '----------------------CV-3000データ取り込み-----------------
    X_1(Count1) = Buffer.Substring(3, 10)
    Y_1(Count1) = Buffer.Substring(14, 10)
    Angle1(Count1) = Buffer.substring(25, 8) * 3.1415 / 180
    'X_2(Count1) = Buffer.Substring(34, 10)
```

```
        'Y_2(Count1) = Buffer.Substring(45, 10)
        'Angle2(Count1) = Buffer.substring(56, 8) * 3.1415 / 180
        'Count(Count1) = Buffer.substring(65, 11)

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click

        '通信開始ボタンのClickイベント
        msSerial.PortOpen = Not msSerial.PortOpen  '交互の制御
        If msSerial.PortOpen = True Then
            '通信開始命令
            Button1.Text = "通信停止"               '表示の切り替え
            Count1 = 2                              '回路基準用V電圧出力
            Engunitsout(3) = Single.Parse(2)
            ULStat = DaqBoard1.FromEngUnits(Range, Engunitsout(3), DataValuesOut(3))
            ULStat = DaqBoard1.AOut(Chan1, Range, DataValuesOut(3))

            Timer1.Enabled = True
        Else
            '通信停止命令
            Button1.Text = "通信開始"
            Timer1.Enabled = False
            Timer2.Enabled = False
            Timer5.Enabled = False
        End If
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        Count1 = 5
        Timer1.Enabled = False

        Timer2.Enabled = True
        Timer5.Enabled = False
    End Sub

    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button7.Click
        Count1 = 2
        Timer1.Enabled = False
        Timer2.Enabled = False
```

```
        Timer5.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick

        InputValue()        'CV-3000データ取り込み
        InputGain()         'フォーム入力の値を変換



        '================================================================================
        '=================PID計算(ヨー角制御のみ)=======================================
        '================================================================================

lab1:   Time1(Count1) = stopWatch.ElapsedMilliseconds / 1000 - Time2(Count1 - 1)  '現時間から
前回ループ時の時間を引き、プログラムの間隔時間を計算
        If Time1(Count1) < 0.1 Then GoTo lab1
        Time2(Count1) = stopWatch.ElapsedMilliseconds / 1000 '次ループの為に現時間を記録


        '----------------------P項の係数----------------------
        P1(Count1) = (Target1(Count1) - Value1(Count1)) '角度
        P2(Count1) = (Target2(Count1) - Value2(Count1)) 'モータ出力

        '----------------------D項の係数----------------------
        dt = Time1(Count1)                          '角度
        dv1 = Value1(Count1) - Value1(Count1 - 1)
        D1(Count1) = dv1 / dt

        dv2 = Value2(Count1) - Value2(Count1 - 1)   'モータ出力
        D2(Count1) = dv2 / dt


        '----------------------I項の係数----------------------
        I1(Count1) = I1(Count1 - 1) + P1(Count1) '角度
        I2(Count1) = I2(Count1 - 1) + P2(Count1) 'モータ出力


        '----------------------印加電圧の計算----------------------
        MVn1(Count1) = Kp1(Count1) * P1(Count1) + Ki1(Count1) * I1(Count1) + Kd1(Count1) *
D1(Count1) + Offset2(Count1) '角度********
        MVn2(Count1) = Kp2(Count1) * P2(Count1) + Ki2(Count1) * I2(Count1) + Kd2(Count1) *
```

```
D2(Count1) + Offset1(Count1) 'モータ出力

        WS(Count1) = 11.89885076 * Val(wind_tunnel_txt.Text) + 0.002739645   '風洞の周波数
        MVn4(Count1) = Val(wind_tunnel_txt.Text)
        huusoku_txt.Text = -0.0187 * MVn4(Count1) ^ 5 + 0.2198 * MVn4(Count1) ^ 4 - 0.9185 *
MVn4(Count1) ^ 3 + 1.5821 * MVn4(Count1) ^ 2 + 0.0406 * MVn4(Count1) + 0.0893 '風洞の風速[Excel
で出た近似曲線式(次数)]

        '==================================電圧出力=======================================

        If Value2(Count1) = 0 Then MVn2(Count1) = 0 '範囲外になればモータ出力を停止に
        If Value1(Count1) = 0 Then MVn1(Count1) = 2 '範囲外になればラダー出力を中立に

        '----------------------ラダー出力----------------------
        Engunitsout(1) = Single.Parse(MVn1(Count1))
        ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(1), DataValuesOut(1))
        ULStat = DaqBoard0.AOut(Chan1, Range, DataValuesOut(1))

        '----------------------モータ出力----------------------
        Engunitsout(2) = Single.Parse(MVn2(Count1))
        ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(2), DataValuesOut(2))
        ULStat = DaqBoard0.AOut(Chan2, Range, DataValuesOut(2))

        '----------------------風洞制御----------------------
        Engunitsout(4) = Single.Parse(MVn4(Count1))
        ULStat = DaqBoard1.FromEngUnits(Range, Engunitsout(4), DataValuesOut(4))
        ULStat = DaqBoard1.AOut(Chan2, Range, DataValuesOut(4))

        '----------------------モータ電圧測定----------------------
        ULStat = DaqBoard0.AIn(ChanIn2, Range1, DataValuesIn(2))
        ULStat = DaqBoard0.ToEngUnits(Range1, DataValuesIn(2), EngUnitsin(2))
        Vin2(Count1) = EngUnitsin(2).ToString("0.0###")

        Mv(Count1) = Vin2(Count1)  '電圧表示
        TextControl()              'テキスト表示
        Count1 = Count1 + 1        'カウント
    End Sub


    Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer2.Tick
        InputValue()        'CV-3000データ取り込み
        InputGain()         'フォーム入力の値を変換
```

```
        '================================================================
        '====================目標角のＰＤ制御============================
        '================================================================

lab1:   Time1(Count1) = stopWatch.ElapsedMilliseconds / 1000 - Time2(Count1 - 1)  '現時間から前回ループ時の時間を引き、プログラムの間隔時間を計算
        If Time1(Count1) < Ts1(Count1) Then GoTo lab1
        Time2(Count1) = stopWatch.ElapsedMilliseconds / 1000 '次ループの為に現時間を記録


        '---------------------------------------ヨー角(Value1)の予測
-----------------------------------
        '----------------------1ステップ予測----------------------------
        Value1h(Count1) = 0.5 * Ts1(Count1) ^ 2 * C1(Count1) * MVn1(Count1 - 4) + Ts1(Count1) * ((Value1(Count1) - Value1(Count1 - 1)) / Ts1(Count1)) + Value1(Count1)
        D4h(Count1) = Ts1(Count1) * C1(Count1) * MVn1(Count1 - 4) + ((Value1(Count1) - Value1(Count1 - 1)) / Ts1(Count1))
        '----------------------2ステップ予測----------------------------
        Value1hh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C1(Count1) * MVn1(Count1 - 3) + Ts1(Count1) * ((Value1h(Count1) - Value1(Count1)) / Ts1(Count1)) + Value1h(Count1)
        D4hh(Count1) = Ts1(Count1) * C1(Count1) * MVn1(Count1 - 3) + ((Value1h(Count1) - Value1(Count1)) / Ts1(Count1))
        '----------------------3ステップ予測----------------------------
        Value1hhh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C1(Count1) * MVn1(Count1 - 2) + Ts1(Count1) * ((Value1hh(Count1) - Value1h(Count1)) / Ts1(Count1)) + Value1hh(Count1)
        D4hhh(Count1) = Ts1(Count1) * C1(Count1) * MVn1(Count1 - 2) + ((Value1hh(Count1) - Value1h(Count1)) / Ts1(Count1))
        '----------------------4ステップ予測----------------------------
        Value1hhhh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C1(Count1) * MVn1(Count1 - 1) + Ts1(Count1) * ((Value1hhh(Count1) - Value1hh(Count1)) / Ts1(Count1)) + Value1hhh(Count1)
        D4hhhh(Count1) = Ts1(Count1) * C1(Count1) * MVn1(Count1 - 1) + ((Value1hhh(Count1) - Value1hh(Count1)) / Ts1(Count1))


        '---------------------------------------位置(Value3)の予測
-----------------------------------
        '----------------------1ステップ予測----------------------------
        Value3h(Count1) = 0.5 * Ts1(Count1) ^ 2 * C2(Count1) * Value1(Count1) + Ts1(Count1) * ((Value3(Count1) - Value3(Count1 - 1)) / Ts1(Count1)) + Value3(Count1)
        D6h(Count1) = Ts1(Count1) * C2(Count1) * Value1(Count1) + ((Value3(Count1) - Value3(Count1
```

```
- 1)) / Ts1(Count1))
        '----------------------2ステップ予測----------------------------
        Value3hh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C2(Count1) * Value1h(Count1) + Ts1(Count1) *
((Value3h(Count1) - Value3(Count1)) / Ts1(Count1)) + Value3h(Count1)
        D6hh(Count1) = Ts1(Count1) * C2(Count1) * Value1h(Count1) + ((Value3h(Count1) -
Value3(Count1)) / Ts1(Count1))
        '----------------------3ステップ予測----------------------------
        Value3hhh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C2(Count1) * Value1hh(Count1) + Ts1(Count1)
* ((Value3hh(Count1) - Value3h(Count1)) / Ts1(Count1)) + Value3hh(Count1)
        D6hhh(Count1) = Ts1(Count1) * C2(Count1) * Value1hh(Count1) + ((Value3hh(Count1) -
Value3h(Count1)) / Ts1(Count1))
        '----------------------4ステップ予測----------------------------
        Value3hhhh(Count1) = 0.5 * Ts1(Count1) ^ 2 * C2(Count1) * Value1hhh(Count1) + Ts1(Count1)
* ((Value3hhh(Count1) - Value3hh(Count1)) / Ts1(Count1)) + Value3hhh(Count1)
        D6hhhh(Count1) = Ts1(Count1) * C2(Count1) * Value1hhh(Count1) + ((Value3hhh(Count1) -
Value3hh(Count1)) / Ts1(Count1))


        '----------------------目標角のPD制御----------------------------
        P4(Count1) = (Target3(Count1) - Value3hhhh(Count1)) 'P項(比例)

        dt = Time1(Count1)                                  'D項(微分)
        dv4 = Value3(Count1) - Value3(Count1 - 1)
        D4(Count1) = dv4 / dt

        Target6(Count1) = Kp3(Count1) * P4(Count1) + Kd3(Count1) * D6hhhh(Count1) +
Target1(Count1)


        '-----------------P項の係数[モータ]-------------------------
        P1(Count1) = (Target6(Count1) - Value1hhhh(Count1)) 'P項(比例)
        P2(Count1) = (Target2(Count1) - Value2(Count1)) 'モータ出力

        '-----------------D項の係数[モータ]-------------------------
        dv2 = Value2(Count1) - Value2(Count1 - 1)       'モータ出力
        D2(Count1) = dv2 / dt

        dt = Time1(Count1)                              '角度
        dv1 = Value1(Count1) - Value1(Count1 - 1)
        D1(Count1) = dv1 / dt


        MVn1(Count1) = Kp4(Count1) * P1(Count1) + Kd4(Count1) * D4hhhh(Count1) + Offset2(Count1)
```

```
'
        MVn2(Count1) = Kp2(Count1) * P2(Count1) + Kd2(Count1) * D2(Count1) + Offset1(Count1) '
モータ出力


        '----------------------印加電圧の計算----------------------

        WS(Count1) = 11.89885076 * Val(wind_tunnel_txt.Text) + 0.002739645    '風洞の周波数
        MVn4(Count1) = Val(wind_tunnel_txt.Text)
        huusoku_txt.Text = -0.0187 * MVn4(Count1) ^ 5 + 0.2198 * MVn4(Count1) ^ 4 - 0.9185 *
MVn4(Count1) ^ 3 + 1.5821 * MVn4(Count1) ^ 2 + 0.0406 * MVn4(Count1) + 0.0893 '風洞の風速[Excel
で出た近似曲線式から(次数)]

        '================================電圧出力=======================================
        If Value2(Count1) = 0 Then MVn2(Count1) = 0 '範囲外になればモータ出力を停止に
        If Value1(Count1) = 0 Then MVn1(Count1) = 2 '範囲外になればラダー出力を中立に

        '----------------------ラダー出力----------------------------
        Engunitsout(1) = Single.Parse(MVn1(Count1))
        ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(1), DataValuesOut(1))
        ULStat = DaqBoard0.AOut(Chan1, Range, DataValuesOut(1))

        '----------------------モータ出力----------------------------
        Engunitsout(2) = Single.Parse(MVn2(Count1))
        ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(2), DataValuesOut(2))
        ULStat = DaqBoard0.AOut(Chan2, Range, DataValuesOut(2))

        '----------------------風洞制御----------------------------
        Engunitsout(4) = Single.Parse(MVn4(Count1))
        ULStat = DaqBoard1.FromEngUnits(Range, Engunitsout(4), DataValuesOut(4))
        ULStat = DaqBoard1.AOut(Chan2, Range, DataValuesOut(4))

        '----------------------モータ電圧測定-----------------------
        ULStat = DaqBoard0.AIn(ChanIn2, Range1, DataValuesIn(2))
        ULStat = DaqBoard0.ToEngUnits(Range1, DataValuesIn(2), EngUnitsin(2))
        Vin2(Count1) = EngUnitsin(2).ToString("0.0###")

        Mv(Count1) = Vin2(Count1)   '電圧表示
        TextControl()               'テキスト表示
        Count1 = Count1 + 1         'カウント
```

```
    End Sub

    Private Sub Timer5_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer5.Tick
        InputValue()          'CV-3000データ取り込み
        InputGain()           'フォーム入力の値を変換

        '================================================================================
        '=========================PID計算(位置制御+高さ制御)============================
        '================================================================================

        Time1(Count1) = stopWatch.ElapsedMilliseconds / 1000 - Time2(Count1 - 1)  '現時間から
前回ループ時の時間を引き、プログラムの間隔時間を計算
        Time2(Count1) = stopWatch.ElapsedMilliseconds / 1000 '次ループの為に現時間を記録

        '----------------------位置制御----------------------------
        P4(Count1) = (Target3(Count1) - Value3(Count1)) 'P項(比例)

        dt = Time1(Count1)                                  'D項(微分)
        dv4 = Value3(Count1) - Value3(Count1 - 1)
        D4(Count1) = dv4 / dt

        '---------------位置制御時のヨー角制御---------------------
        P3(Count1) = (Target5(Count1) - Value1(Count1)) 'P項(比例)
        dt = Time1(Count1)                                  'D項(微分)
        dv3 = Value1(Count1) - Value1(Count1 - 1)
        D3(Count1) = dv3 / dt

        '-----------------P項の係数[モータ]-------------------------
        P2(Count1) = (Target2(Count1) - Value2(Count1)) 'モータ出力

        '-----------------D項の係数[モータ]-------------------------
        dv2 = Value2(Count1) - Value2(Count1 - 1)       'モータ出力
        D2(Count1) = dv2 / dt

        '-----------------I項の係数[モータ]-------------------------
        I2(Count1) = I2(Count1 - 1) + P2(Count1)        'モータ出力

        '----------------------印加電圧の計算----------------------
        MVn1(Count1) = Kp3(Count1) * P4(Count1) + Kd3(Count1) * D4(Count1) + Kp4(Count1) *
P3(Count1) + Kd4(Count1) * D3(Count1) + Offset2(Count1) '角度+位置
        MVn2(Count1) = Kp2(Count1) * P2(Count1) + Ki2(Count1) * I2(Count1) + Kd2(Count1) *
D2(Count1) + Offset1(Count1) 'モータ出力
```

```
'==========================================================
'=======================風洞制御=============================
'==========================================================

'------------------------P項の係数[風洞制御]--------------------------------
P5(Count1) = (Target4(Count1) - Y_2(Count1)) '高さ(カメラ_Y)

'------------------------D項の係数[風洞制御]--------------------------------
dt = Time1(Count1)                          ''高さ(カメラ_Y)
dv5 = Y_2(Count1) - Y_2(Count1 - 1)
D5(Count1) = dv1 / dt

'------------------------印加電圧の計算[風洞制御]----------------------------
MVn5(Count1) = Val(wind_tunnel_txt.Text)
MVn4(Count1) = Kp5(Count1) * P5(Count1) + Kd5(Count1) * D5(Count1) + MVn5(Count1)

'--------------------風洞の風速・周波数計算-------------------------
WS(Count1) = 11.89885076 * Val(wind_tunnel_txt.Text) + 0.002739645   '風洞の周波数
huusoku_txt.Text = -0.0187 * MVn4(Count1) ^ 5 + 0.2198 * MVn4(Count1) ^ 4 - 0.9185 *
MVn4(Count1) ^ 3 + 1.5821 * MVn4(Count1) ^ 2 + 0.0406 * MVn4(Count1) + 0.0893 '風洞の風速[Excel
で出た近似曲線式から(次数)]

'==========================電圧出力=============================
If Value2(Count1) = 0 Then MVn2(Count1) = 0 '範囲外になればモータ出力を停止に
If Value1(Count1) = 0 Then MVn1(Count1) = 2 '範囲外になればラダー出力を中立に

'----------------------ラダー出力----------------------------
Engunitsout(1) = Single.Parse(MVn1(Count1))
ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(1), DataValuesOut(1))
ULStat = DaqBoard0.AOut(Chan1, Range, DataValuesOut(1))

'----------------------モータ出力----------------------------
Engunitsout(2) = Single.Parse(MVn2(Count1))
ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(2), DataValuesOut(2))
ULStat = DaqBoard0.AOut(Chan2, Range, DataValuesOut(2))

'----------------------風洞制御----------------------------
Engunitsout(4) = Single.Parse(MVn4(Count1))
ULStat = DaqBoard1.FromEngUnits(Range, Engunitsout(4), DataValuesOut(4))
ULStat = DaqBoard1.AOut(Chan2, Range, DataValuesOut(4))

'----------------------モータ電圧測定------------------------
```

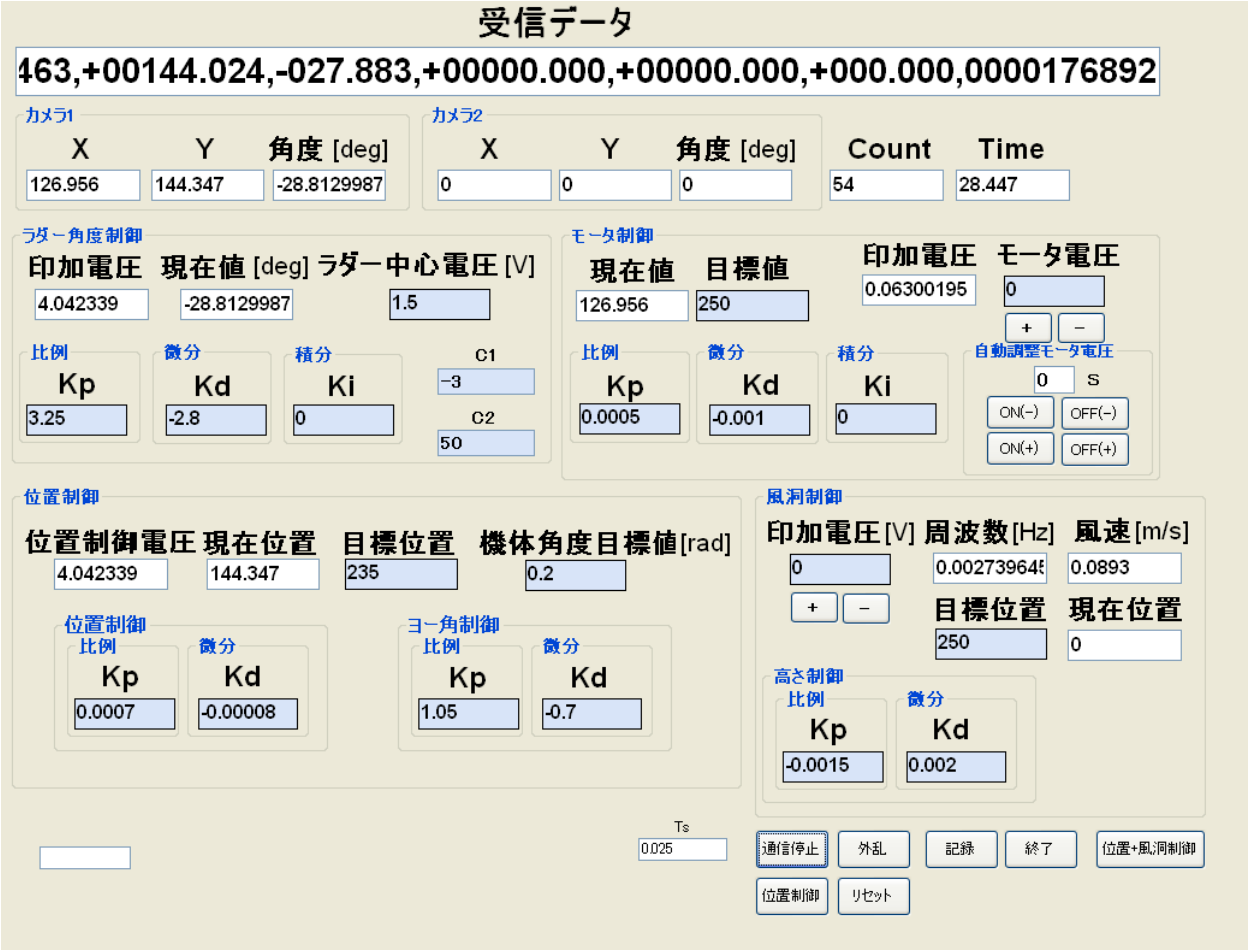
```
        ULStat = DaqBoard0.AIn(ChanIn2, Range1, DataValuesIn(2))
        ULStat = DaqBoard0.ToEngUnits(Range1, DataValuesIn(2), EngUnitsin(2))
        Vin2(Count1) = EngUnitsin(2).ToString("0.0###")

        Mv(Count1) = Vin2(Count1)  '電圧表示
        TextControl()              'テキスト表示
        Count1 = Count1 + 1        'カウント

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        Dim n As Integer                        'カウント用関数
        Dim dtNow As DateTime = DateTime.Now '現在の日時を取得
        Dim stPrompt1 As String = dtNow.ToString("yyyy年MM月dd日_(dddd)_tt_hh時mm分ss秒") ' 指
定した書式で日付を文字列に変換する
        Dim Writer1 As New IO.StreamWriter("C:¥" & stPrompt1 & "1[HF].csv") 'ファイル書き込み
開始、ファイル保存先決定
        'Dim Writer2 As New IO.StreamWriter("C:¥" & stPrompt1 & "2[HF].csv") 'ファイル書き込み
開始、ファイル保存先決定

        Writer1.WriteLine("Count" & "," & "Time" & "," & "X_1" & "," & "Y_1" & "," & "X_2" & ","
& "Y_2" & "," & "yaw(deg)" & "," & "pitch(deg)" & "," & "Kp(r)" & "," & "Kd(r)" & "," & "Kp(M)"
& "," & "Kd(M)" & "," & "Target(M)" & "," & "Target(r)" & "," & "Target(W)" & "," & "V(r)" & ","
& "V(M)" & "," & "V(W)" & "," & "Time2" & "," & "Yta" & "," & "Ota" & "," & "Kpy" & "," & "Kdy"
& "," & "Kpo" & "," & "Kdo" & "," & "MVn1(r)" & "," & "MVn2(M)") 'ファイルに書き込み

        For n = 0 To Count1 Step 1
            Writer1.WriteLine(Count(n) & "," & Time2(n) & "," & X_1(n) & "," & Y_1(n) & "," & X_2(n)
& "," & Y_2(n) & "," & deg1(n) & "," & deg2(n) & "," & Kp1(n) & "," & Kd1(n) & "," & Kp2(n) & ","
& Kd2(n) & "," & Target2(n) & "," & Target1(n) & "," & Target4(n) & "," & MVn1(n) & "," & MVn2(n)
& "," & MVn4(n) & "," & Time2(n) & "," & Target3(n) & "," & Target5(n) & "," & Kp3(n) & "," & Kd3(n)
& "," & Kp4(n) & "," & Kd4(n) & "," & MVn1(n) & "," & MVn2(n))     'ファイルに書き込み
        Next n
        Writer1.Close()

        'Writer2.WriteLine("Count" & "," & "Time2" & "," & "Yta" & "," & "Ota" & "," & "Kpy" &
"," & "Kdy" & "," & "Kpo" & "," & "Kdo" & "," & "MVn1(r)" & "," & "MVn2(M)") 'ファイルに書き込
み

        'For n = 0 To Count1 Step 1
        'Writer2.WriteLine(Count(n) & "," & Time2(n) & "," & Target3(n) & "," & Target5(n) & ","
& Kp3(n) & "," & Kd3(n) & "," & Kp4(n) & "," & Kd4(n) & "," & MVn1(n) & "," & MVn2(n)) 'ファイ
```

```
ルに書き込み
        'Next n
        'Writer2.Close() '書き込み終了
    End Sub

    Private Sub Timer3_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer3.Tick
        TimeCounter.Text = TimeCounter.Text + 0.1

        If TimeCounter.Text = 3 Then            '調整時間はここの数値で指定する[今の設定は(s)]
            v = 0.01
            Offset1_txt.Text = (Offset1_txt.Text - v).ToString("0.00") 'ここで指定した場所の数
値を減らす
            TimeCounter.Text = 0
        End If
    End Sub

    Private Sub Timer4_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer4.Tick
        TimeCounter.Text = TimeCounter.Text + 0.1

        If TimeCounter.Text = 3 Then            '調整時間はここの数値で指定する[今の設定は(s)]
            v = 0.01
            Offset1_txt.Text = (Offset1_txt.Text + v).ToString("0.00") 'ここで指定した場所の数
値を増やす
            TimeCounter.Text = 0
        End If
    End Sub

    Private Sub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button14.Click
        Timer3.Enabled = True
        Button14.Enabled = False
        Button15.Enabled = False
        Button16.Enabled = False
    End Sub

    Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button13.Click
        Timer3.Enabled = False
        Button14.Enabled = True
        Button15.Enabled = True
        Button16.Enabled = True
```

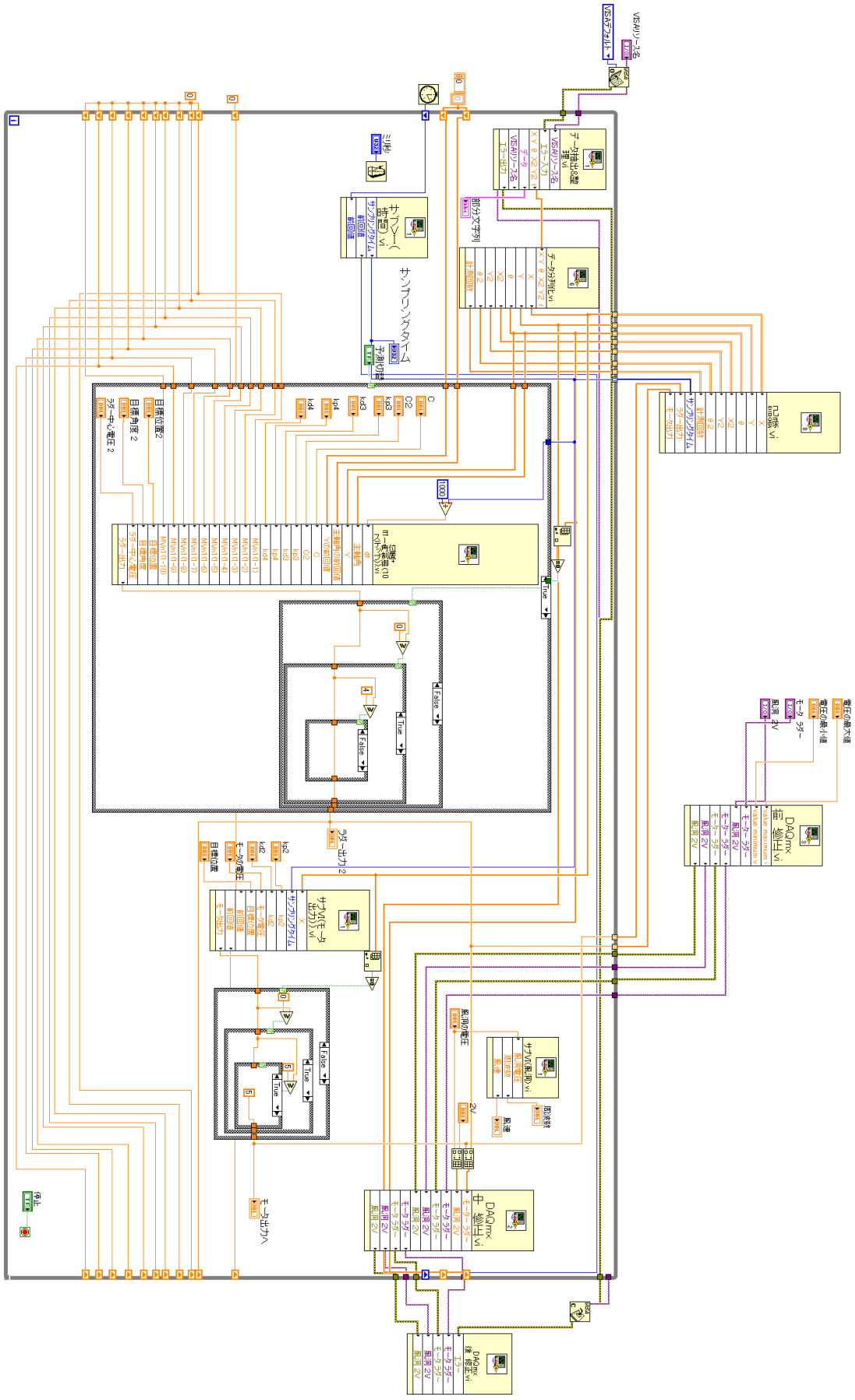
```
        TimeCounter.Text = 0
    End Sub

    Private Sub Button15_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button15.Click
        Timer4.Enabled = True
        Button13.Enabled = False
        Button14.Enabled = False
        Button15.Enabled = False
    End Sub

    Private Sub Button16_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button16.Click
        Timer4.Enabled = False
        Button13.Enabled = True
        Button14.Enabled = True
        Button15.Enabled = True
        TimeCounter.Text = 0
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        '外乱発生プログラム
        Dim a As Integer
        Timer1.Enabled = False
        For a = 0 To 50
            Engunitsout(1) = Single.Parse(2.4)
            ULStat = DaqBoard0.FromEngUnits(Range, Engunitsout(1), DataValuesOut(1))
            ULStat = DaqBoard0.AOut(Chan1, Range, DataValuesOut(1))
        Next a
        Timer1.Enabled = True
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        Count1 = 1
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        End
    End Sub
```

```
    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button8.Click
        v = 0.01
        Offset1_txt.Text = (Offset1_txt.Text + v).ToString("0.00") 'ここで指定した場所の数値を
増やす
    End Sub

    Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
        v = 0.01
        Offset1_txt.Text = (Offset1_txt.Text - v).ToString("0.00") 'ここで指定した場所の数値を
増やす
    End Sub

    Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button10.Click
        v = 0.01
        wind_tunnel_txt.Text = (wind_tunnel_txt.Text + v).ToString("0.00") 'ここで指定した場所
の数値を増やす
    End Sub

    Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button11.Click
        v = 0.01
        wind_tunnel_txt.Text = (wind_tunnel_txt.Text - v).ToString("0.00") 'ここで指定した場所
の数値を減らす
    End Sub



    Private Sub Angle_adjustment_txt_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Angle_adjustment_txt.TextChanged

    End Sub



End Class
```

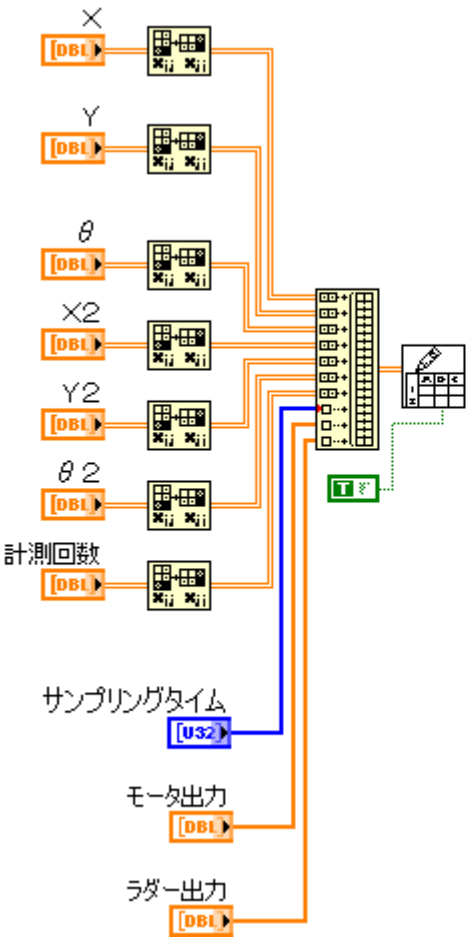
付録 2 制御パネル（Visual Basic）



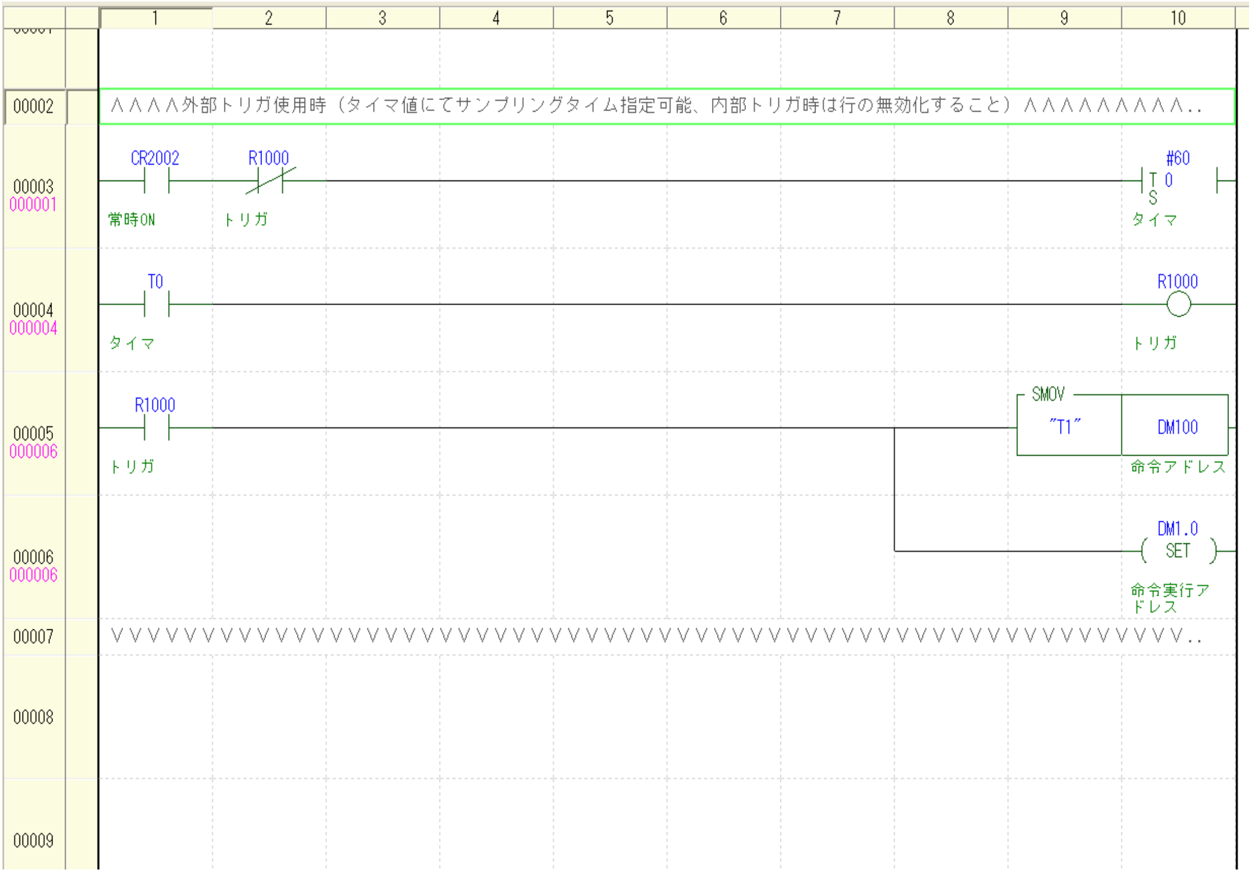
付録 3 飛行制御プログラム（LabVIEW）

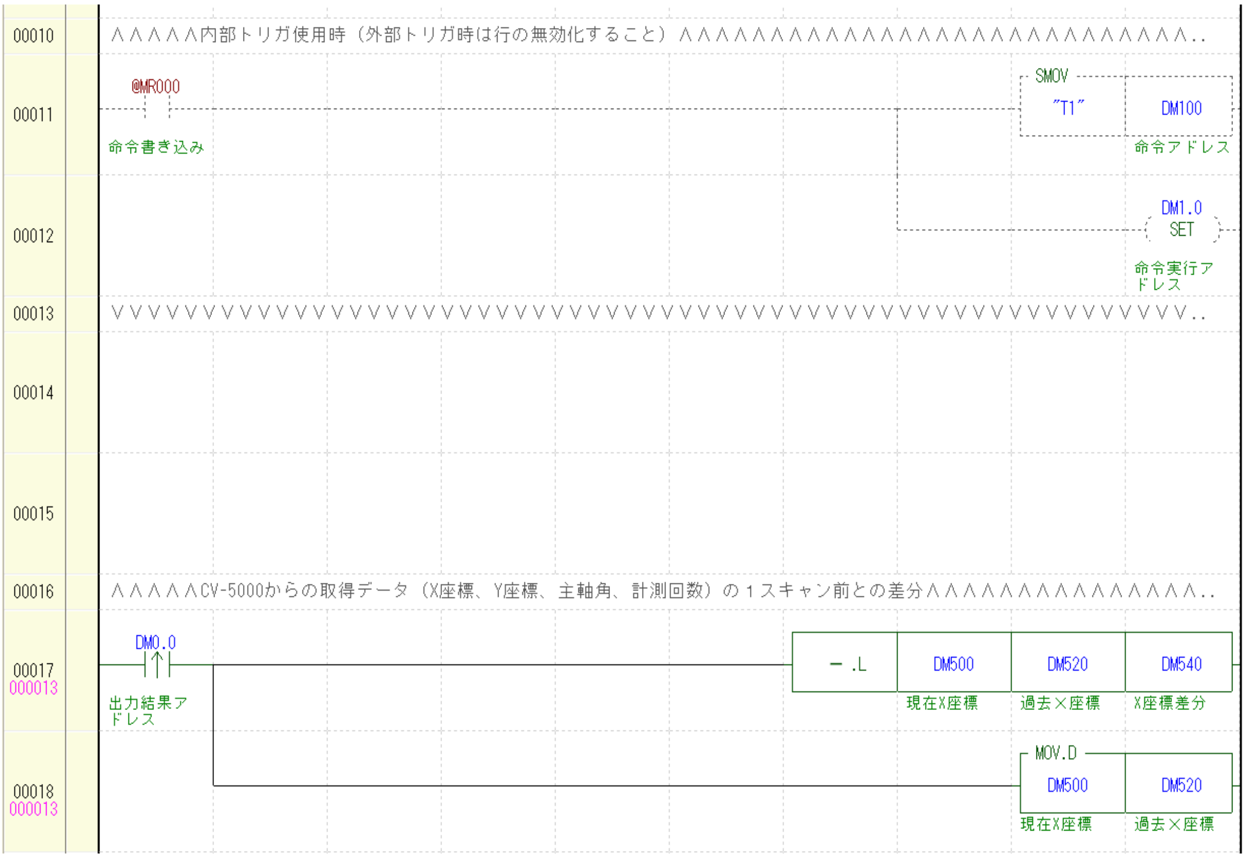


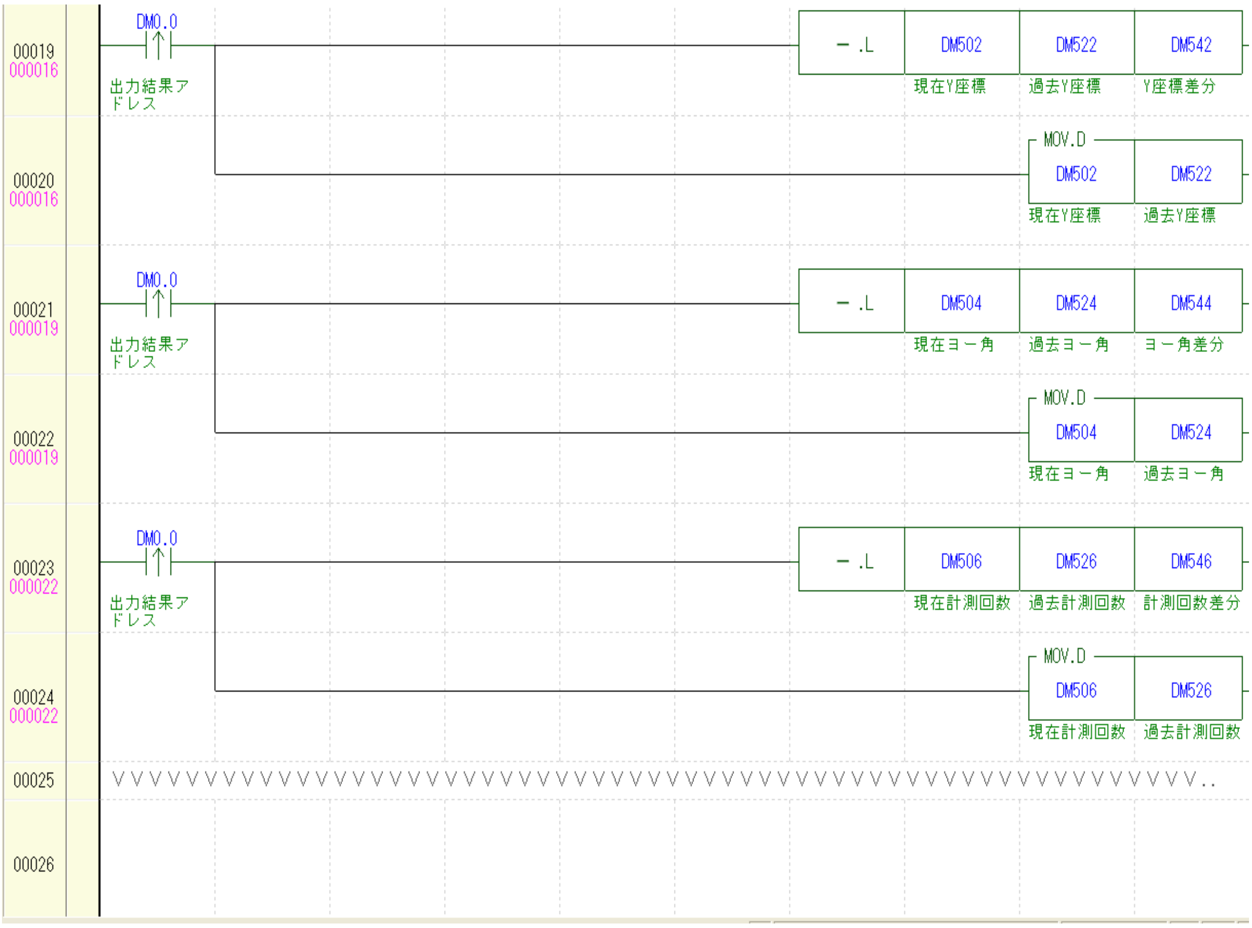
付録 4 データ保存プログラム（LabVIEW）

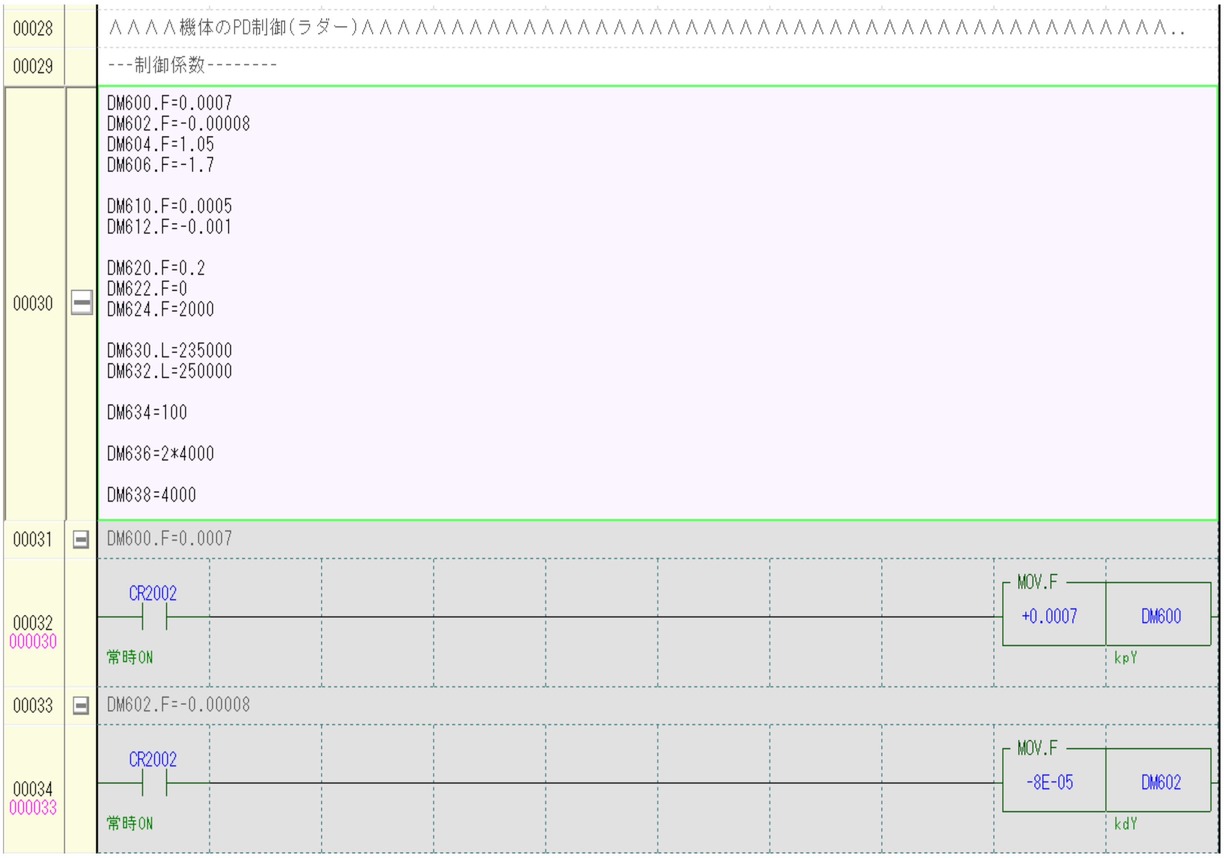


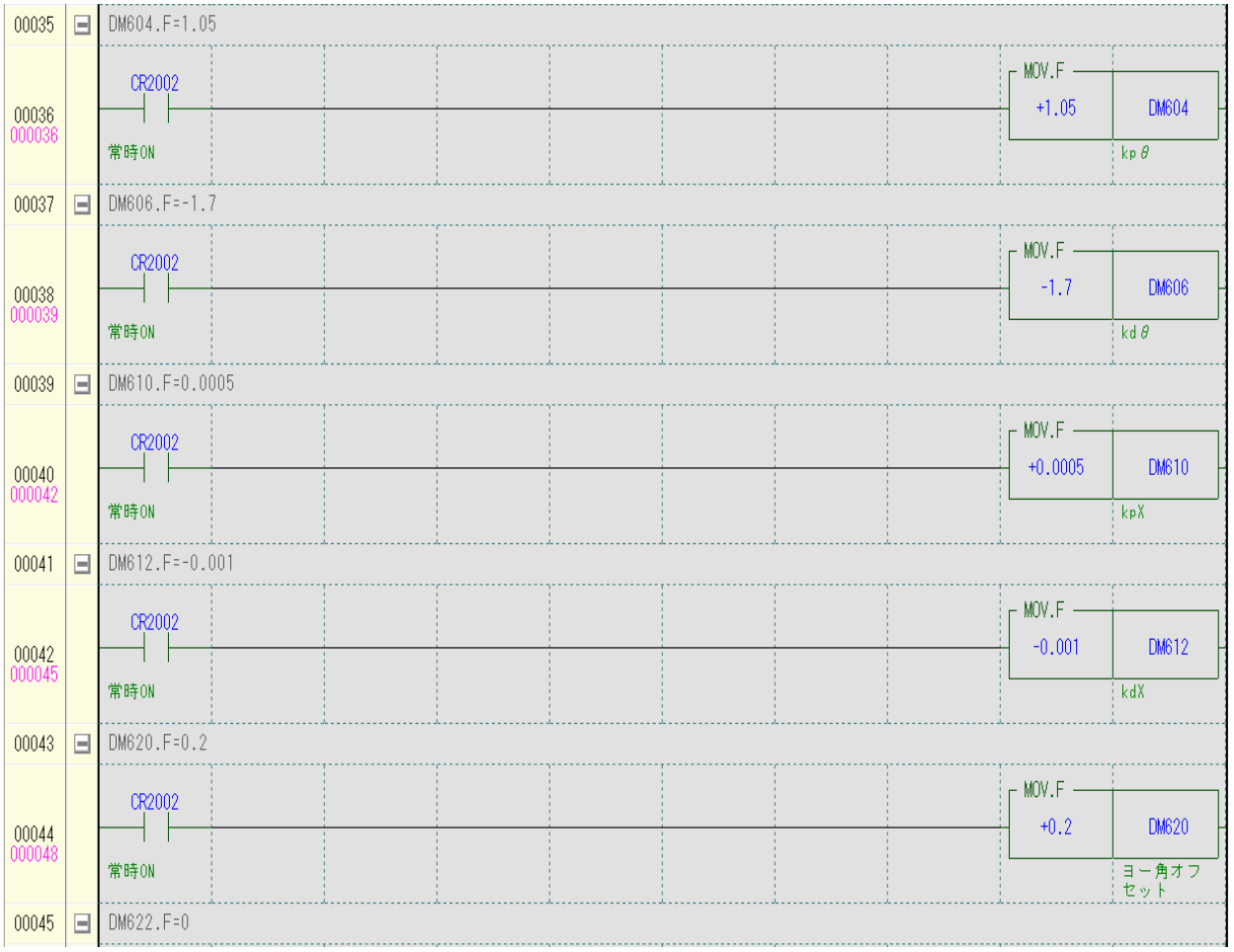
付録 5 飛行制御プログラム（KV-5000 ラダープログラム）

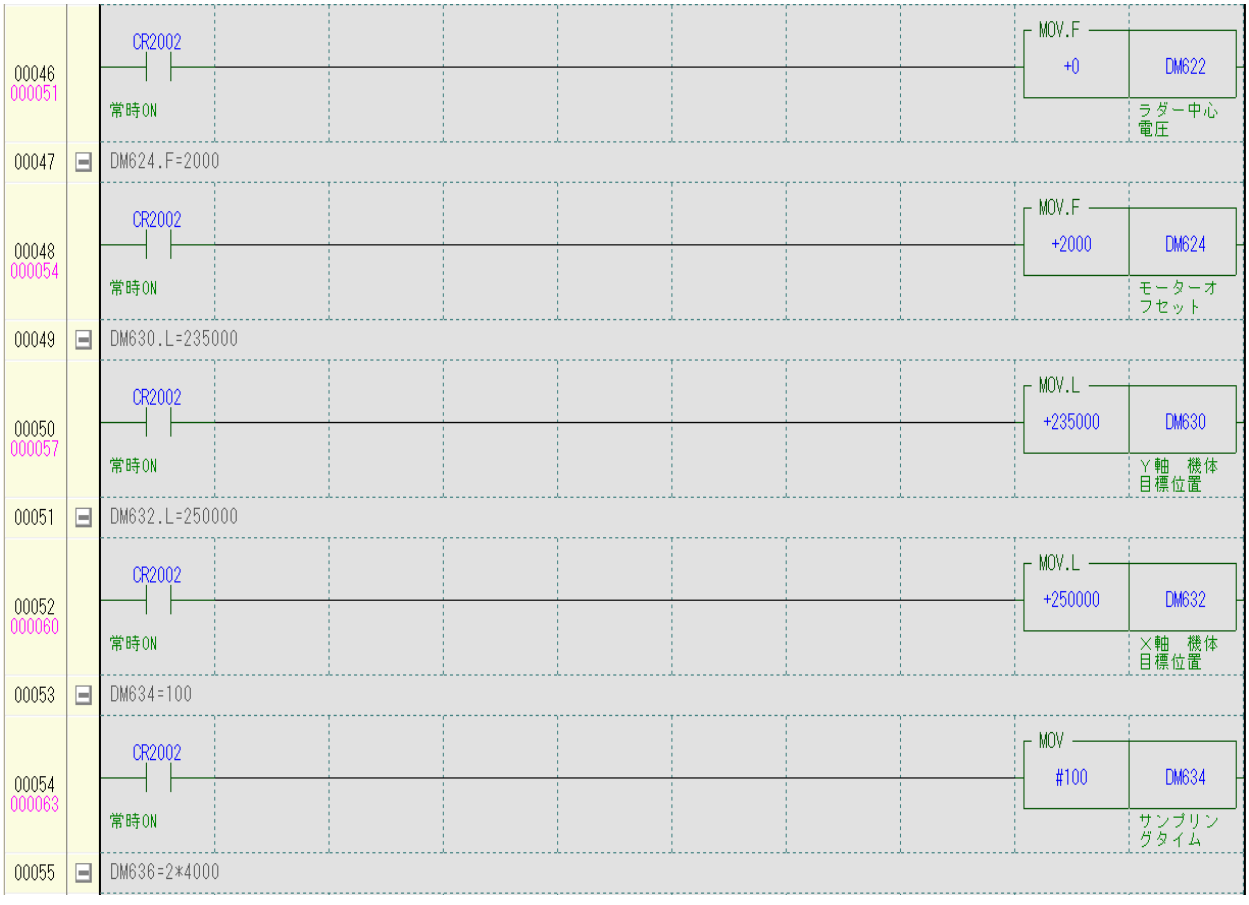


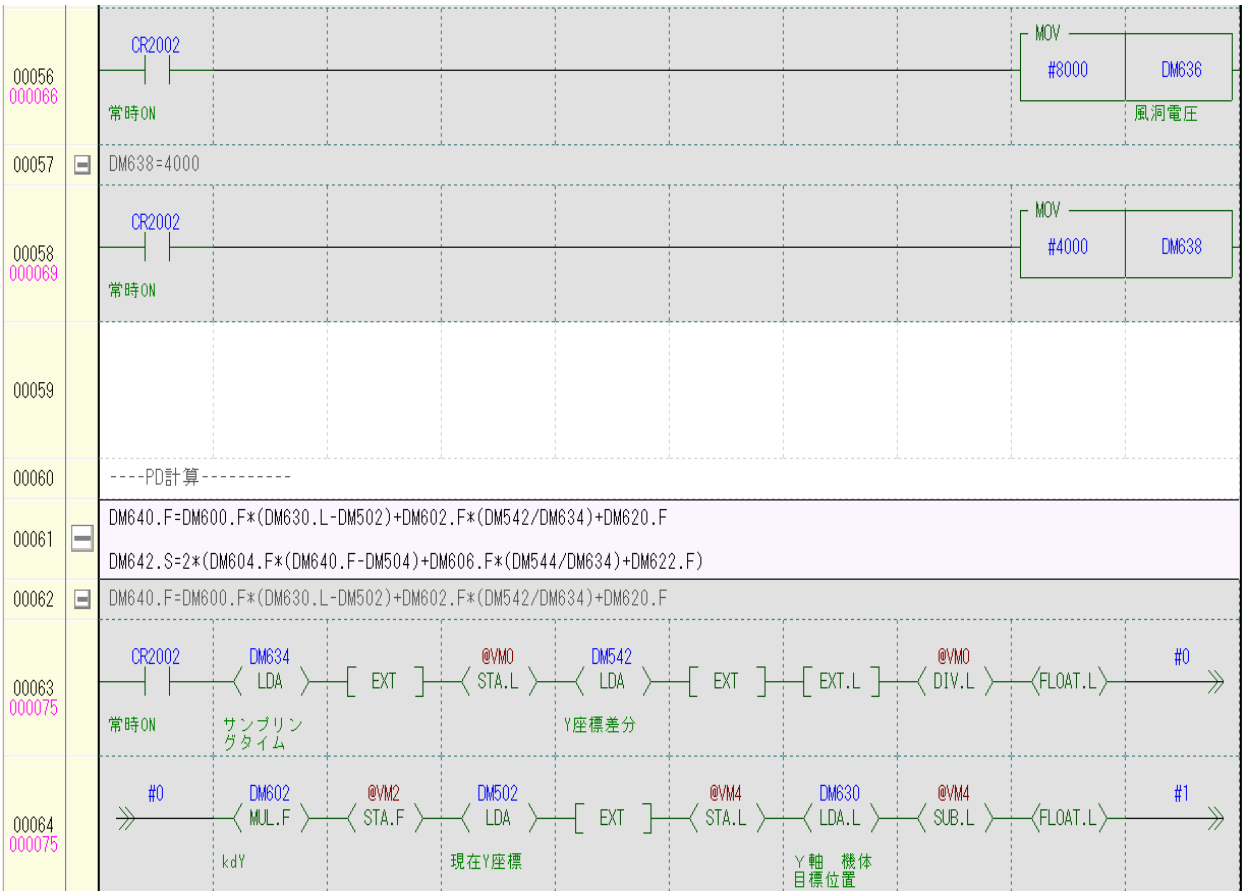


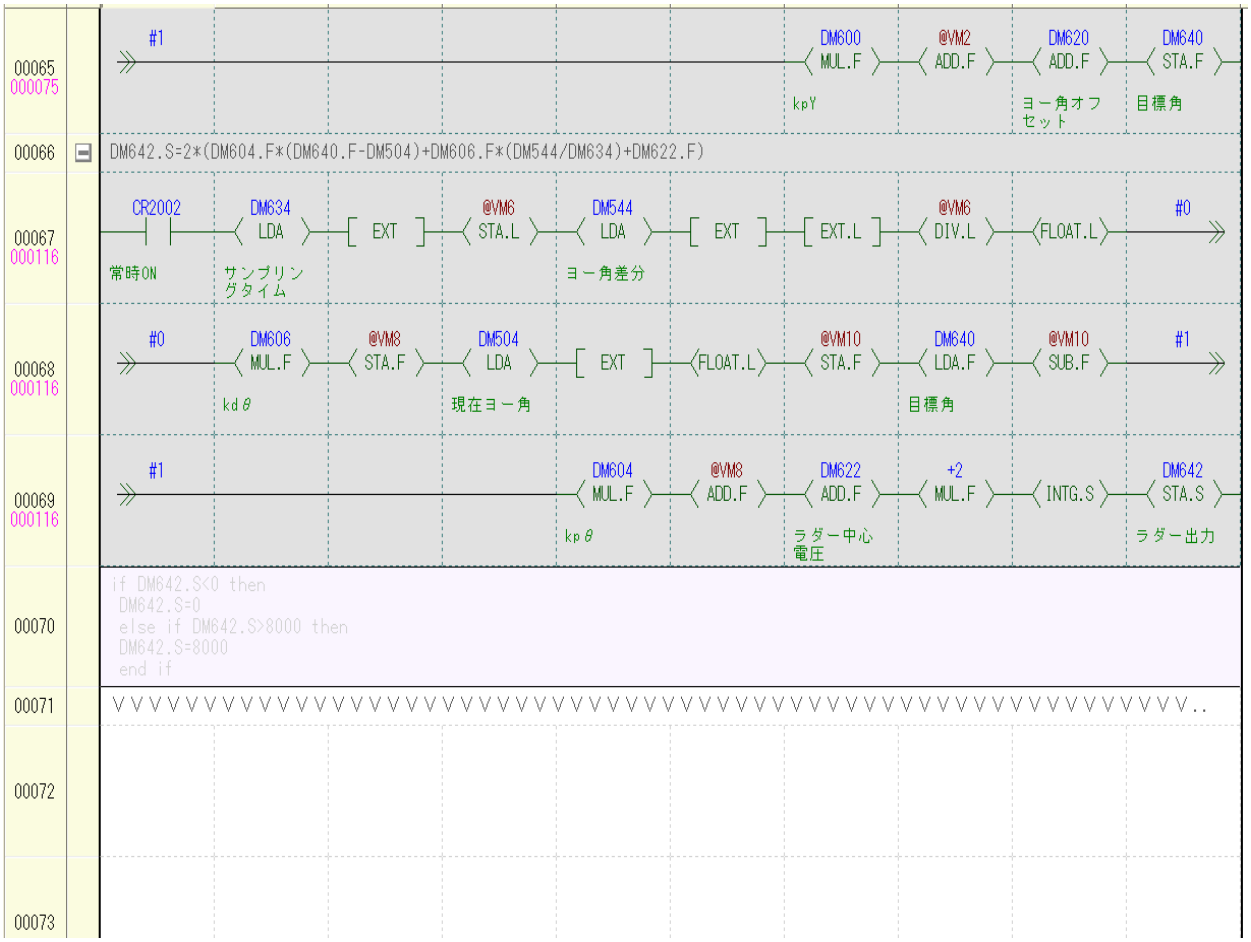

00019 000016
DM0.0
出力結果アドレス
－.L
DM502
DM522
DM542
現在Y座標
過去Y座標
Y座標差分
00020 000016
MOV.D
DM502
DM522
現在Y座標
過去Y座標
00021 000019
DM0.0
出力結果アドレス
－.L
DM504
DM524
DM544
現在ヨー角
過去ヨー角
ヨー角差分
00022 000019
MOV.D
DM504
DM524
現在ヨー角
過去ヨー角
00023 000022
DM0.0
出力結果アドレス
－.L
DM506
DM526
DM546
現在計測回数
過去計測回数
計測回数差分
00024 000022
MOV.D
DM506
DM526
現在計測回数
過去計測回数
00025 VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV..
00026

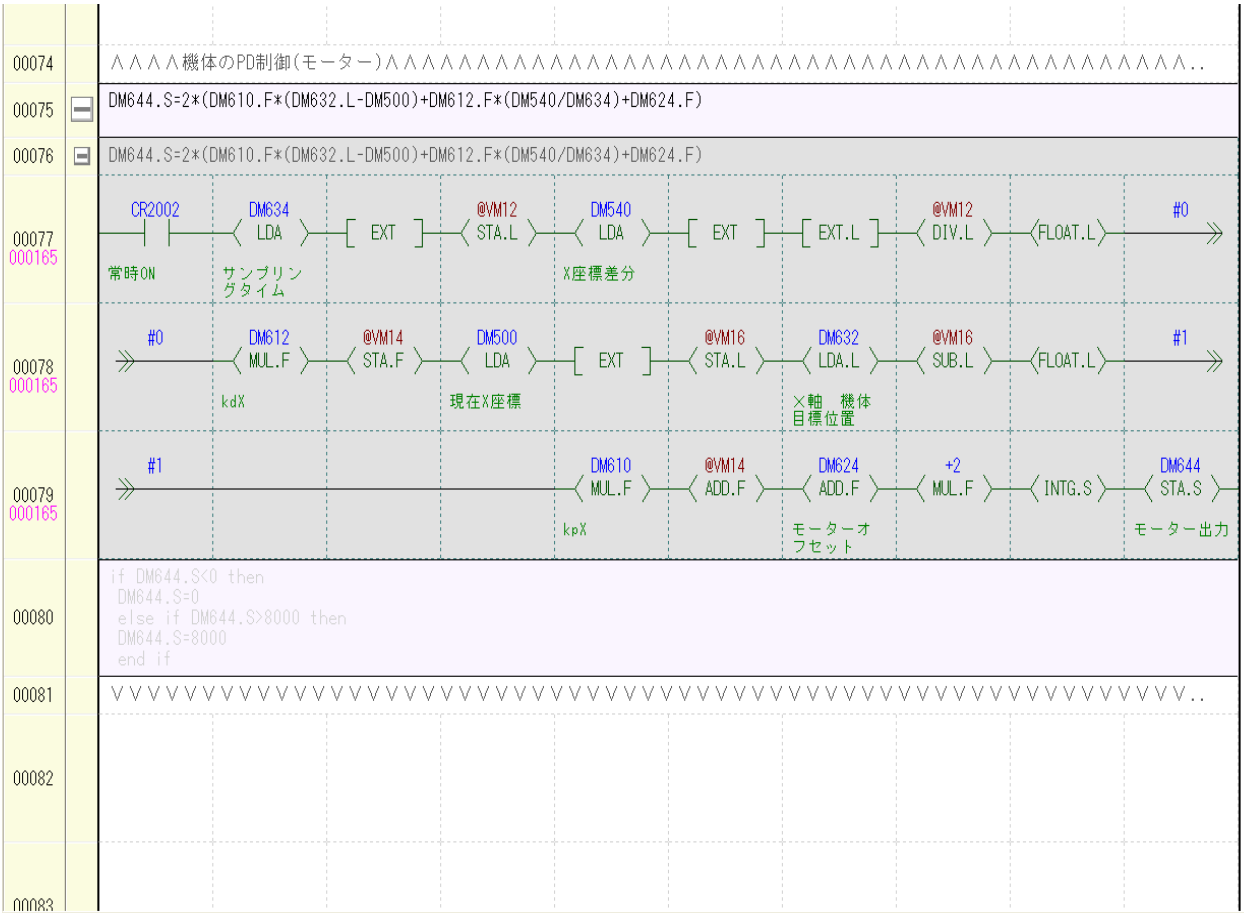

00028 ∧∧∧∧機体のPD制御(ラダー)∧∧∧..
00029 ---制御係数--------
00030
DM600.F=0.0007
DM602.F=-0.00008
DM604.F=1.05
DM606.F=-1.7
DM610.F=0.0005
DM612.F=-0.001
DM620.F=0.2
DM622.F=0
DM624.F=2000
DM630.L=235000
DM632.L=250000
DM634=100
DM636=2*4000
DM638=4000
00031 DM600.F=0.0007
00032 000030
CR2002
常時ON
MOV.F
+0.0007
DM600
kpY
00033 DM602.F=-0.00008
00034 000033
CR2002
常時ON
MOV.F
-8E-05
DM602
kdY

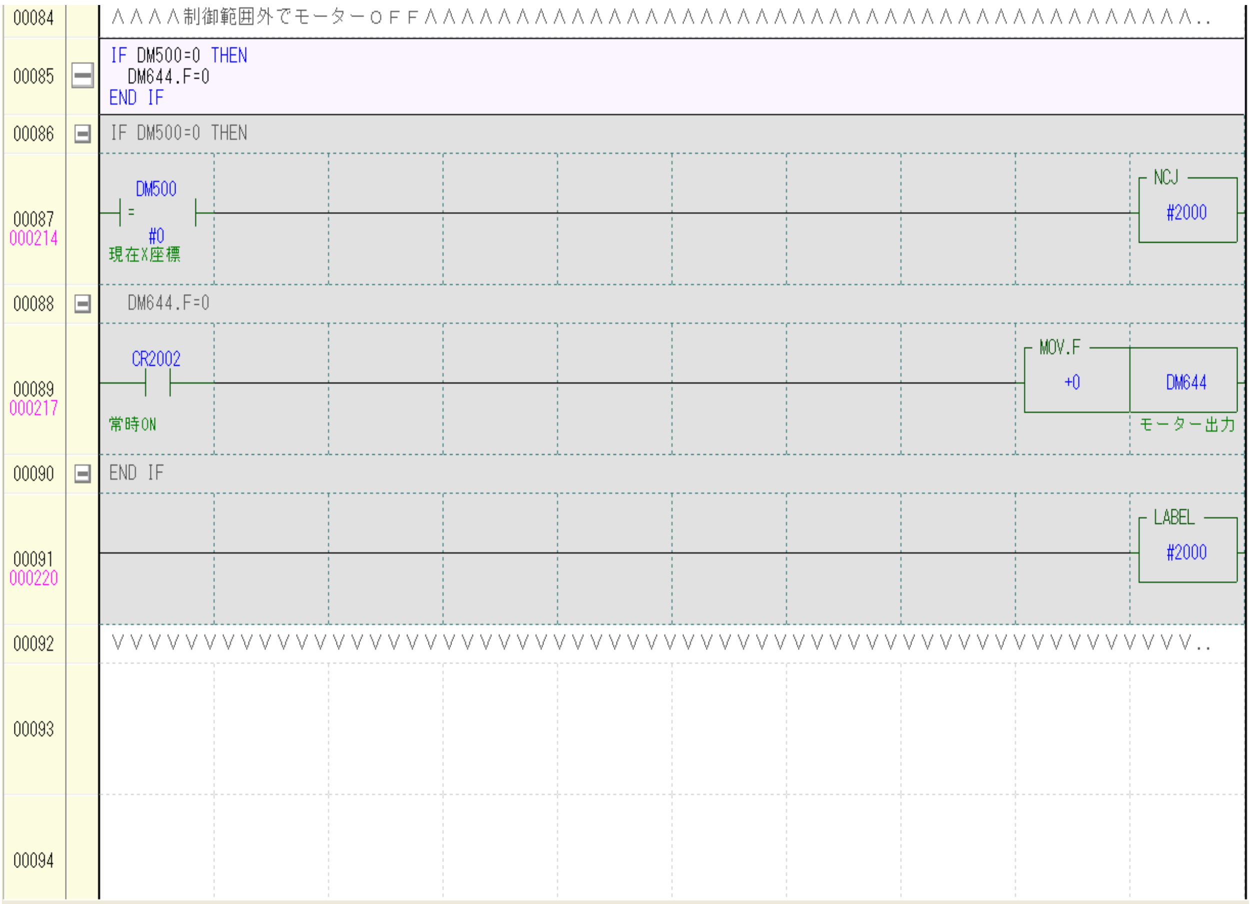

00035
DM604.F=1.05
00036
000036
CR2002
常時ON
MOV.F
+1.05
DM604
kpθ
00037
DM606.F=-1.7
00038
000039
CR2002
常時ON
MOV.F
-1.7
DM606
kdθ
00039
DM610.F=0.0005
00040
000042
CR2002
常時ON
MOV.F
+0.0005
DM610
kpX
00041
DM612.F=-0.001
00042
000045
CR2002
常時ON
MOV.F
-0.001
DM612
kdX
00043
DM620.F=0.2
00044
000048
CR2002
常時ON
MOV.F
+0.2
DM620
ヨー角オフセット
00045
DM622.F=0

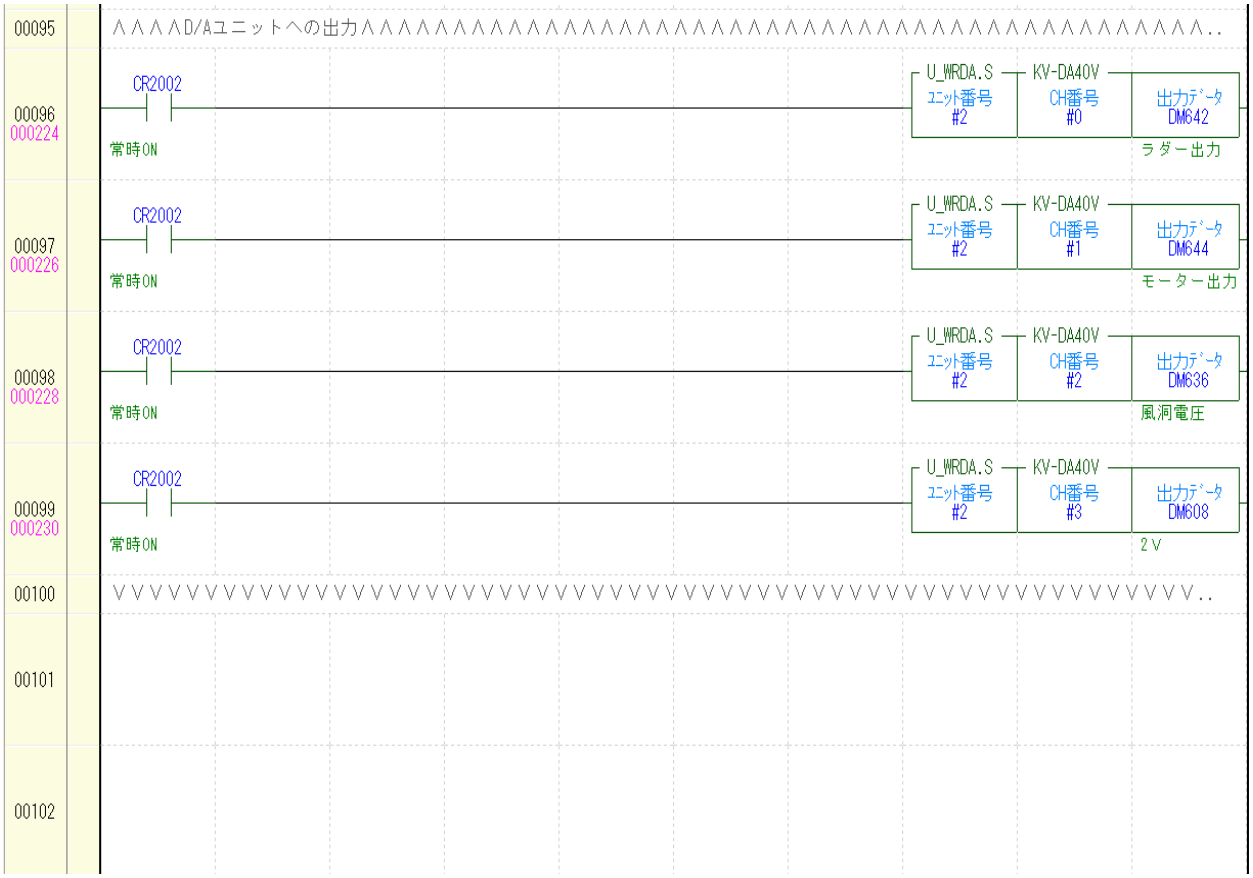

00046
000051
CR2002
常時ON
MOV.F
+0
DM622
ラダー中心電圧
00047
DM624.F=2000
00048
000054
CR2002
常時ON
MOV.F
+2000
DM624
モーターオフセット
00049
DM630.L=235000
00050
000057
CR2002
常時ON
MOV.L
+235000
DM630
Y軸　機体目標位置
00051
DM632.L=250000
00052
000060
CR2002
常時ON
MOV.L
+250000
DM632
X軸　機体目標位置
00053
DM634=100
00054
000063
CR2002
常時ON
MOV
#100
DM634
サンプリングタイム
00055
DM636=2*4000

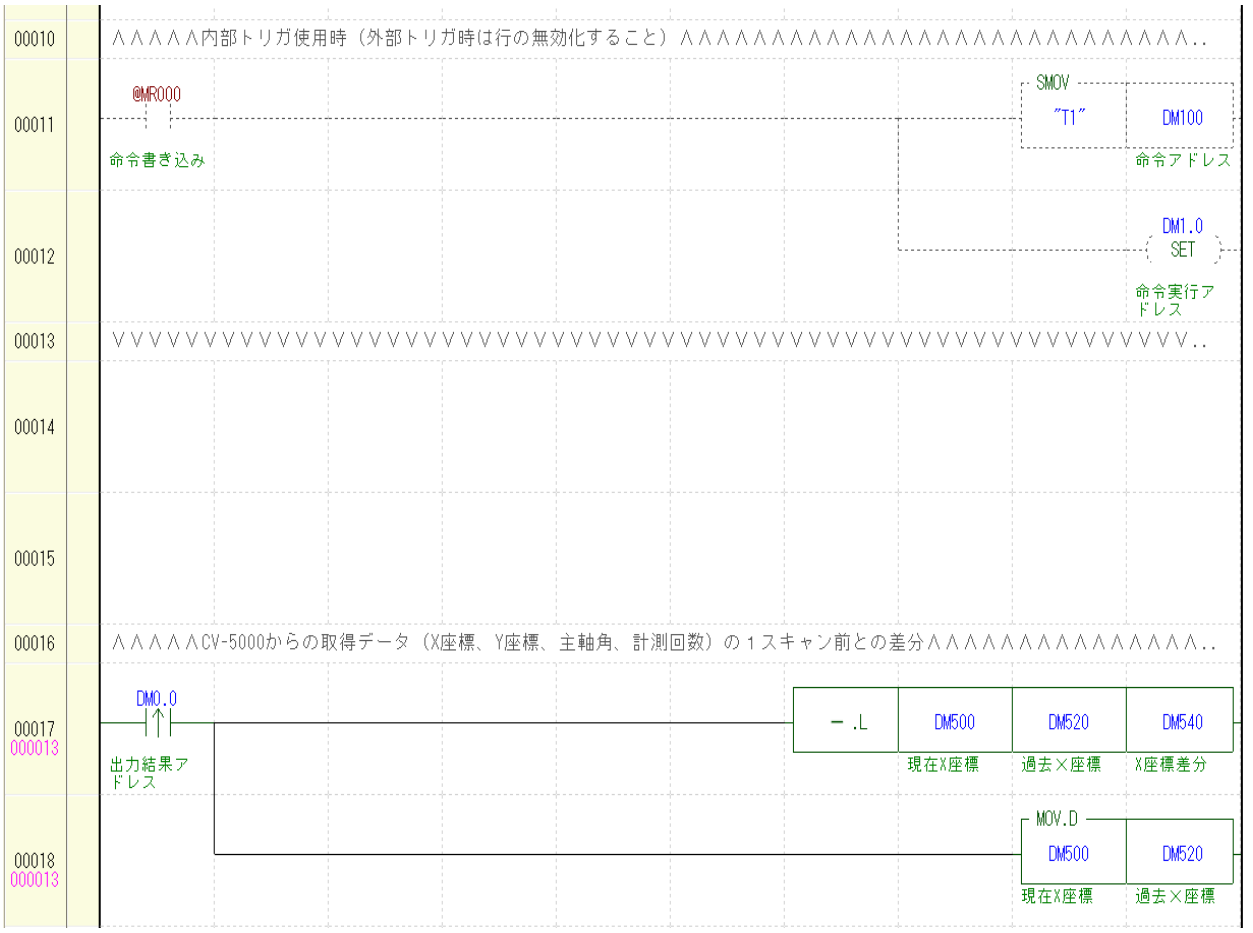

00056
000066
CR2002
常時ON
MOV
#8000
DM636
風洞電圧
00057
DM638=4000
00058
000069
CR2002
常時ON
MOV
#4000
DM638
00059
00060
----PD計算----------
00061
DM640.F=DM600.F*(DM630.L-DM502)+DM602.F*(DM542/DM634)+DM620.F
DM642.S=2*(DM604.F*(DM640.F-DM504)+DM606.F*(DM544/DM634)+DM622.F)
00062
DM640.F=DM600.F*(DM630.L-DM502)+DM602.F*(DM542/DM634)+DM620.F
00063
000075
CR2002
常時ON
DM634
LDA
サンプリングタイム
EXT
@VM0
STA.L
DM542
LDA
Y座標差分
EXT
EXT.L
@VM0
DIV.L
FLOAT.L
#0
00064
000075
#0
DM602
MUL.F
kdY
@VM2
STA.F
DM502
LDA
現在Y座標
EXT
@VM4
STA.L
DM630
LDA.L
Y軸　機体
目標位置
@VM4
SUB.L
FLOAT.L
#1


00065
000075
#1
DM600
MUL.F
kpY
@VM2
ADD.F
DM620
ADD.F
ヨー角オフセット
DM640
STA.F
目標角
00066
DM642.S=2*(DM604.F*(DM640.F-DM504)+DM606.F*(DM544/DM634)+DM622.F)
00067
000116
CR2002
常時ON
DM634
LDA
サンプリングタイム
EXT
@VM6
STA.L
DM544
LDA
ヨー角差分
EXT
EXT.L
@VM6
DIV.L
FLOAT.L
#0
00068
000116
#0
DM606
MUL.F
kdθ
@VM8
STA.F
DM504
LDA
現在ヨー角
EXT
FLOAT.L
@VM10
STA.F
DM640
LDA.F
目標角
@VM10
SUB.F
#1
00069
000116
#1
DM604
MUL.F
kpθ
@VM8
ADD.F
DM622
ADD.F
ラダー中心
電圧
+2
MUL.F
INTG.S
DM642
STA.S
ラダー出力
00070
if DM642.S<0 then
DM642.S=0
else if DM642.S>8000 then
DM642.S=8000
end if
00071
00072
00073


00074
∧∧∧∧機体のPD制御(モーター)∧∧..
00075
DM644.S=2*(DM610.F*(DM632.L-DM500)+DM612.F*(DM540/DM634)+DM624.F)
00076
DM644.S=2*(DM610.F*(DM632.L-DM500)+DM612.F*(DM540/DM634)+DM624.F)
00077
000165
CR2002
常時ON
DM634
LDA
サンプリングタイム
EXT
@VM12
STA.L
DM540
LDA
X座標差分
EXT
EXT.L
@VM12
DIV.L
FLOAT.L
#0
00078
000165
#0
DM612
MUL.F
kdX
@VM14
STA.F
DM500
LDA
現在X座標
EXT
@VM16
STA.L
DM632
LDA.L
X軸 機体目標位置
@VM16
SUB.L
FLOAT.L
#1
00079
000165
#1
DM610
MUL.F
kpX
@VM14
ADD.F
DM624
ADD.F
モーターオフセット
+2
MUL.F
INTG.S
DM644
STA.S
モーター出力
00080
if DM644.S<0 then
DM644.S=0
else if DM644.S>8000 then
DM644.S=8000
end if
00081
∨∨..
00082
00083


00084
∧∧∧∧制御範囲外でモーターＯＦＦ∧∧∧..
00085
IF DM500=0 THEN
DM644.F=0
END IF
00086
IF DM500=0 THEN
00087
000214
DM500
=
#0
現在X座標
NCJ
#2000
00088
DM644.F=0
00089
000217
CR2002
常時ON
MOV.F
+0
DM644
モーター出力
00090
END IF
00091
000220
LABEL
#2000
00092
∨∨..
00093
00094


00095
D/Aユニットへの出力
00096
000224
CR2002
常時ON
U_WRDA.S
KV-DA40V
ユニット番号 #2
CH番号 #0
出力データ DM642
ラダー出力
00097
000226
CR2002
常時ON
U_WRDA.S
KV-DA40V
ユニット番号 #2
CH番号 #1
出力データ DM644
モーター出力
00098
000228
CR2002
常時ON
U_WRDA.S
KV-DA40V
ユニット番号 #2
CH番号 #2
出力データ DM636
風洞電圧
00099
000230
CR2002
常時ON
U_WRDA.S
KV-DA40V
ユニット番号 #2
CH番号 #3
出力データ DM608
2V
00100
00101
00102


00010
内部トリガ使用時（外部トリガ時は行の無効化すること）
00011
@MR000
命令書き込み
SMOV
"T1"
DM100
命令アドレス
00012
DM1.0
SET
命令実行アドレス
00013
00014
00015
00016
CV-5000からの取得データ（X座標、Y座標、主軸角、計測回数）の１スキャン前との差分
00017
000013
DM0.0
出力結果アドレス
－.L
DM500
現在X座標
DM520
過去×座標
DM540
X座標差分
00018
000013
MOV.D
DM500
現在X座標
DM520
過去×座標