卒業論文

工作機械熱変形補償制御用変形センサの開発

P1～P82

平成 15 年 2 月 6 日

指導教官　　長尾　高明　教授

1030089　　浅田　崇

目次

目次

第一章
序論

1-1　序論・本研究の目的

・古代から我々人類は頭と手を使って人間の暮らしに役立つものを作ってきた。そして、人類が発展するにつれて人間の行っていた作業が、機械に移行していくようになった。このような機械を作る機械はマザーマシンとよばれ、高水準の生産性や信頼性を要求される。

熟練者でなくては、高い水準の加工ができなかった汎用工作機械から工作機械の NC 化によって、本やマニュアルを読んだだけで一定の加工ができるようになり、また一日 24 時間の生産が可能となった。

しかし、熟練者が全く必要でなくなったのではなく、高い精度が要求される加工では、熟練者の経験は必要不可欠である。

加工の高精度化、高信頼性化への要求が高まる中で、工作機械が熱や加工反力により変形し、精度が低下する問題はいまだに解決されておらず、熟練者の経験に依存しており、高生産性を実現できていない。

実際、工作機械の熱変形による精度の低下に対し、加工中に発生する熱を冷却することによって抑えてはいるが、発生する全ての熱を解消することは不可能である。熟練者が経験に基づいて補正を加えるかあるいは、機械を事前に数時間ほど空運転することにより熱変形が安定するのを待って使用するといった方法で対処しているのが現状である。熱変形そのものを抑制する手段はいくつか講じられているが、熱伝播、熱膨張を全てなくすことができない限り精度低下は起こってしまう。そこで、発熱を抑制するだけでなく工作機械の熱変形を補償し熱環境に依存しない工作機械を作りだそうという研究であり、マシニングセンタにおいて、熱変形による精度低下を低減する試みである。そこで私たちは、変形センサを設計・開発し、それをマシニングセンタに設置し、熱変形などいろいろな変形、歪みを予測し、データを収集し、精度低下を低減し、加工精度向上のための研究を行った。また、マシニングセンタだけでなく、より熱変形がわかりやすいように実験装置の設計・開発にも取り組んだ。

第二章
加工の知能化

2-1　加工の知能化の基本概念

・ 加工の知能化の概念とは、「工作機械の系の挙動に関する知識に系を支配する基本要素の情報を入力すれば、系に対して所望の動作をさせる情報を作り出すことができ、その情報を用いて加工現象を制御すれば所望の製品が作り出せる。」というものである。ここで知識とは加工により生じる現象そのもの(物理モデル)についての知識と加工プロセスについての知識である。また基本要素とはこの知識と実際の現象を取り込むセンサ、および情報に応じて実際に動作を起こさせるアクチュエータからなる。図 1.1 に加工中に生じる物理現象について示す。

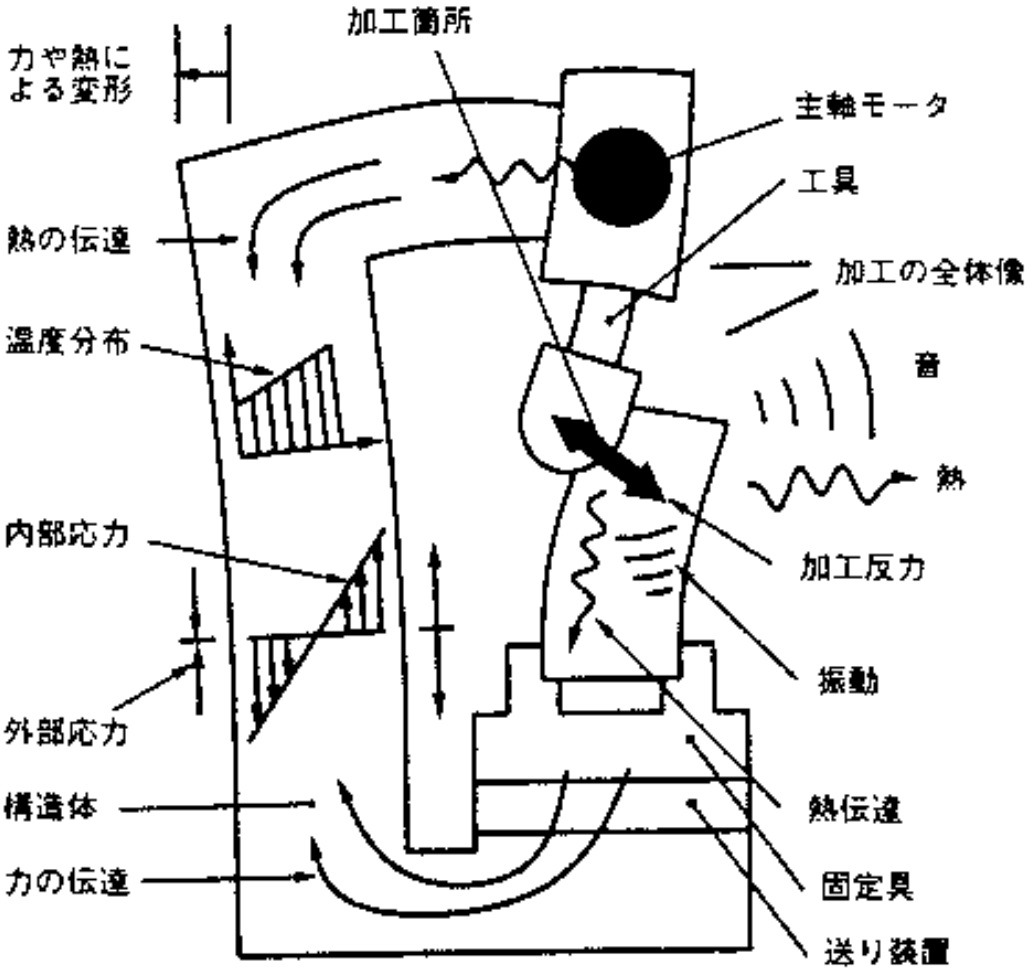

図 1.1　加工中に生じる物理現象

2-2　加工の知能化の基本構成

- **マシニングセンタを始めとする工作機械はこれらの物理現象によって生じる力と熱によって変形し、同時に歪み、温度、音などが発生する。そこで、これらの情報をセンサによって取り込み、設計に関する知識に基づき処理することで目標値と実際とのずれを予測し、機械に設置したアクチュエータによってずれを修正することができれば製品を正確に作り出すことが可能になる。また、センサ情報、アクチュエータの動作内容、加工の結果をデータ・ベース化し、それに基づき内部モデルを修正することで、システム自体が学習、自己進化することも可能である。このような考えを「加工の知能化」と呼び、基本構成を図 1-2 のダイヤグラムに示す。**

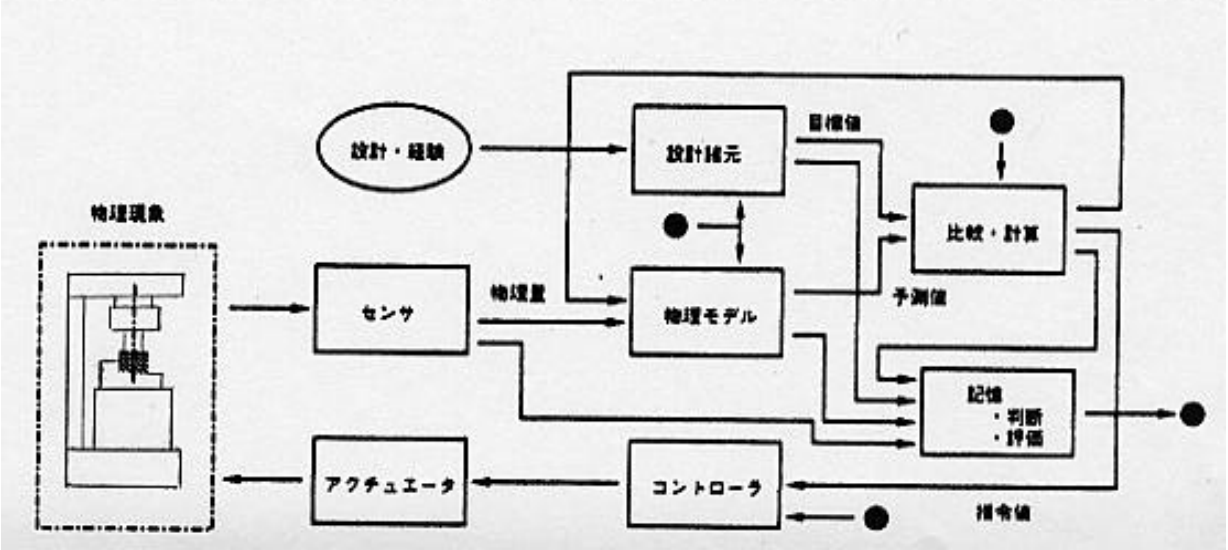

図 1-2　ダイヤグラム

2-3　マシニングセンタにおける熱変形能動補償の意義

・ 前述したように様々な物理現象が加工中に生じているわけだが、このうちマシニングセンタの加工精度を低下させる大きな要因として、加工反力による変形、熱による変形を挙げることができる。このうち加工反力による変形に対処するために、従来は工作機械本体の剛性を高めることによって変形を抑えるという方法がとられてきた。しかし、この方法では、材料の剛性に限界があることや、構造体の重量が重くなり過ぎることなどの制約があり、根本的な解決には至っていない。一方、もう一つの要因として取り上げた熱変形については、従来は発熱する部分を冷やす、熱が他の部分に伝わるのを出来るだけ制御する、といった対策を施すことで少しでも変形に軽減しようとしていたが、発生した熱を全て冷却することが出来ない以上、限界ができてしまう。事実、加工することなく単に主軸を回転させるだけでも､わずか 2 時間で約 40μmもの主軸変位が発生する。実際には、主軸モータの発熱に加え、工具と被加工物との間に生じる加工発熱、周囲の環境の変化に伴う熱の移動など、さらに多くの熱変形の要因が存在しており、これらを全て取り除くことは不可能である。また、マシニングセンタのように多くの部品と複雑な構造で構成されるものでは、加工反力による変形と熱変形を具体的に解析し、モデル化することは非常に困難である。よって、モデル化による熱変形補正は複雑な熱変形を十分に補正できない。しかし、実際問題として加工の高精密化が必要とされている現状では、これらの変形による加工精度の低下は解決されなければならない問題である。

第三章
構成

3-1　システムの概略

・本研究に用いた加工システムの概略を図 3-1 に示す。
次節以降で、それぞれの機能について述べていく。

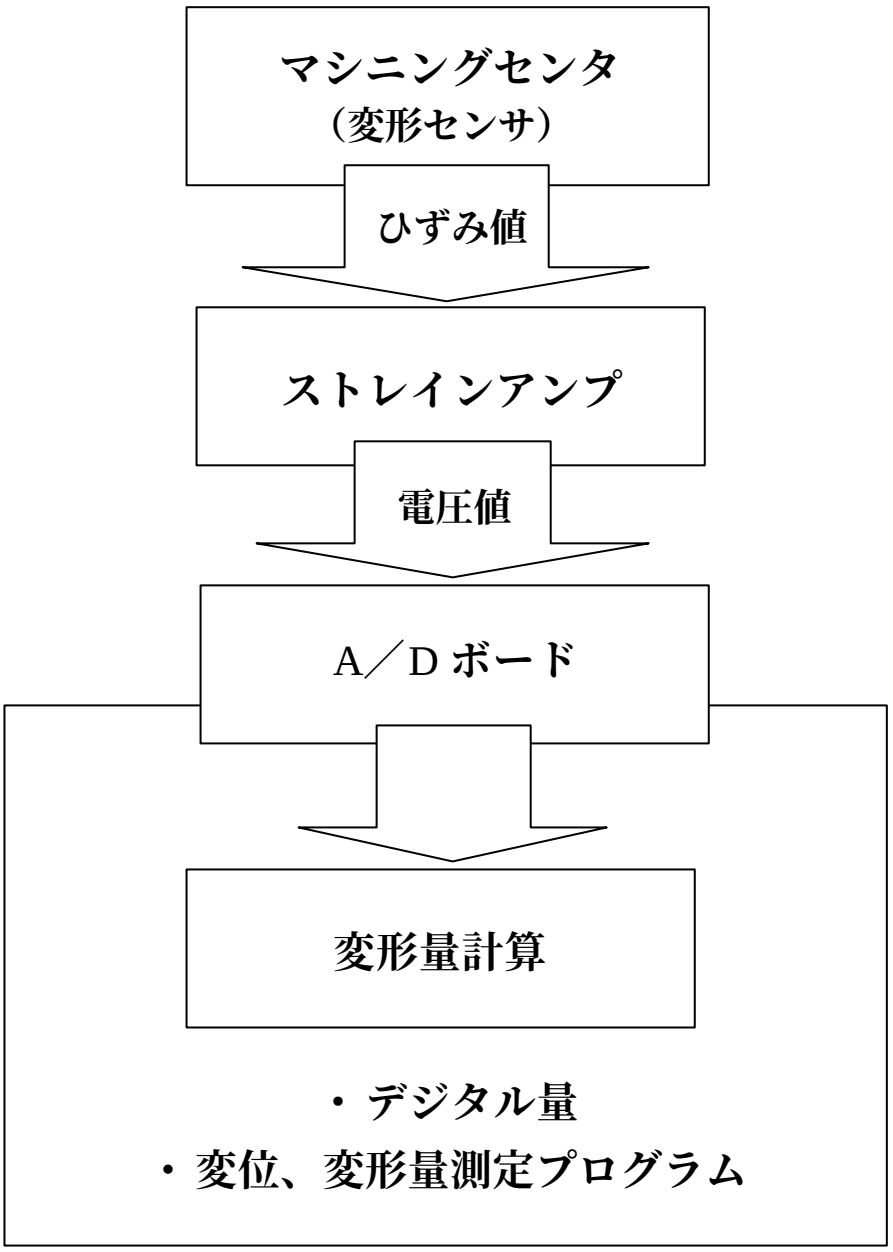

図 3-1　加工システムの概略

3-2　オープン CNC マシニングセンタ

- マシニングセンタには大阪機工株式会社(OKK)の VM4Ⅱ立型マシニングセンタ(図 3-2)を使用した。これは、オープン CNC マシニングセンタといい、外部のパソコンとのデータの送受信や操作命令を受けることとコンピュータ数値制御ができるものである。また、この VM4Ⅱ立型マシニングセンタの仕様を表 3.1 に示す。またマシニングセンタの概略を図 3-3 に示す。

図 3-2　VM4Ⅱ立型マシニングセンタ

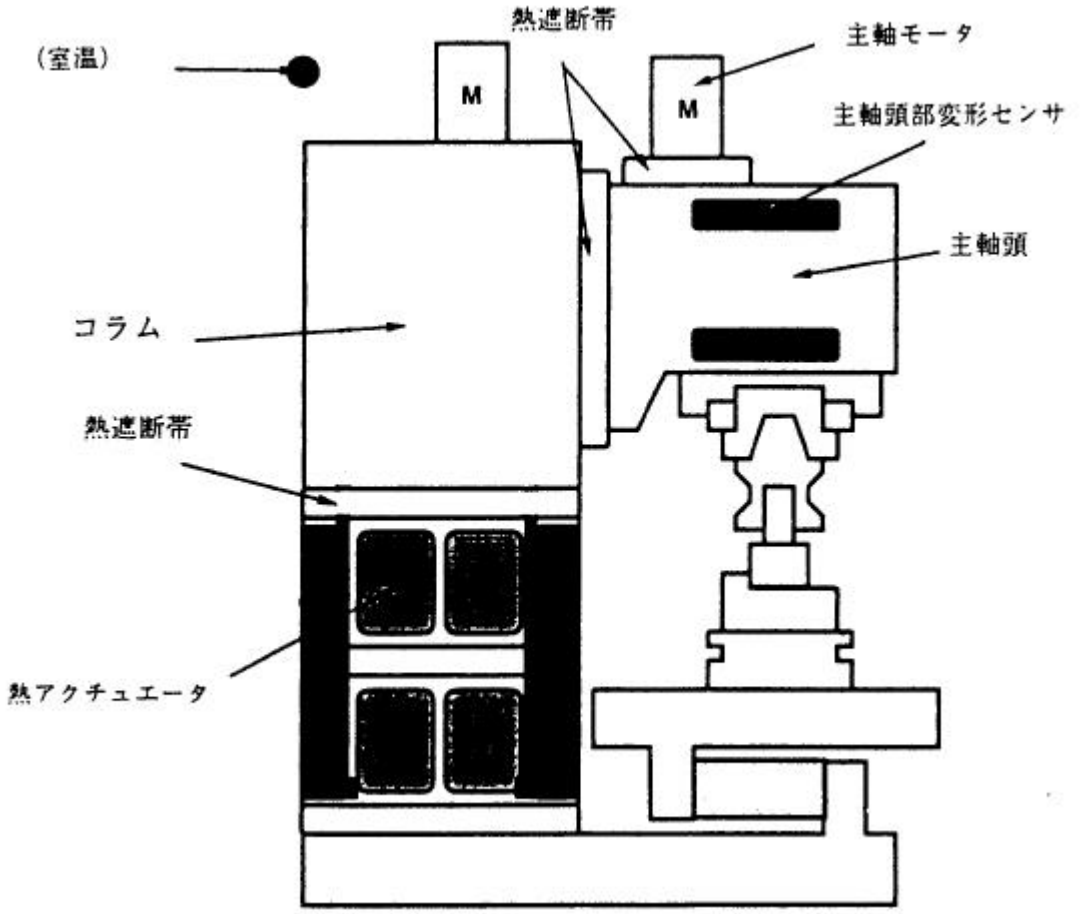

図 3-3　VM4Ⅱ立型マシニングセンタの概略

表 3.1　VM4Ⅱ立型マシニングセンタの仕様

テーブル作業面	800×410mm
最大移動距離	
テーブル左右 （X）	630mm
サドル（前後）（Y）	410mm
ヘッド（上下）（Z）	460mm
切削送り速度	1 ～ 10000 mm／min
早送り速度	XY：24m／min
	Z：16m／min
ジョグ送り速度	2000m／min

・このマシニングセンタには高速かつ広範囲のデータを送受信することが可能な FANUC 社の高速シリアルバス(HSSB：High Speed Serial Bus)(図 3-4) を搭載する。これにより HSSB ボードを組み込んだパソコンからのデータの送受信が可能になる。

図 3-4　FANUC 社の高速シリアルバス

3-3　熱アクチュエータ

本研究に用いるような 3 軸のマシニングセンタでは、構造体の変形による主軸位置の変位のうち並進方向の変位については座標系の原点をずらすことで 1μm刻みでの補正が可能であるが、主軸の傾きについてはＮＣ機能では補正できない。そこで、マシニングセンタのコラム下部に図 3-5 のように加熱板と冷却ファンをコラム前面、背面、側面、ATC 側面に設置する。

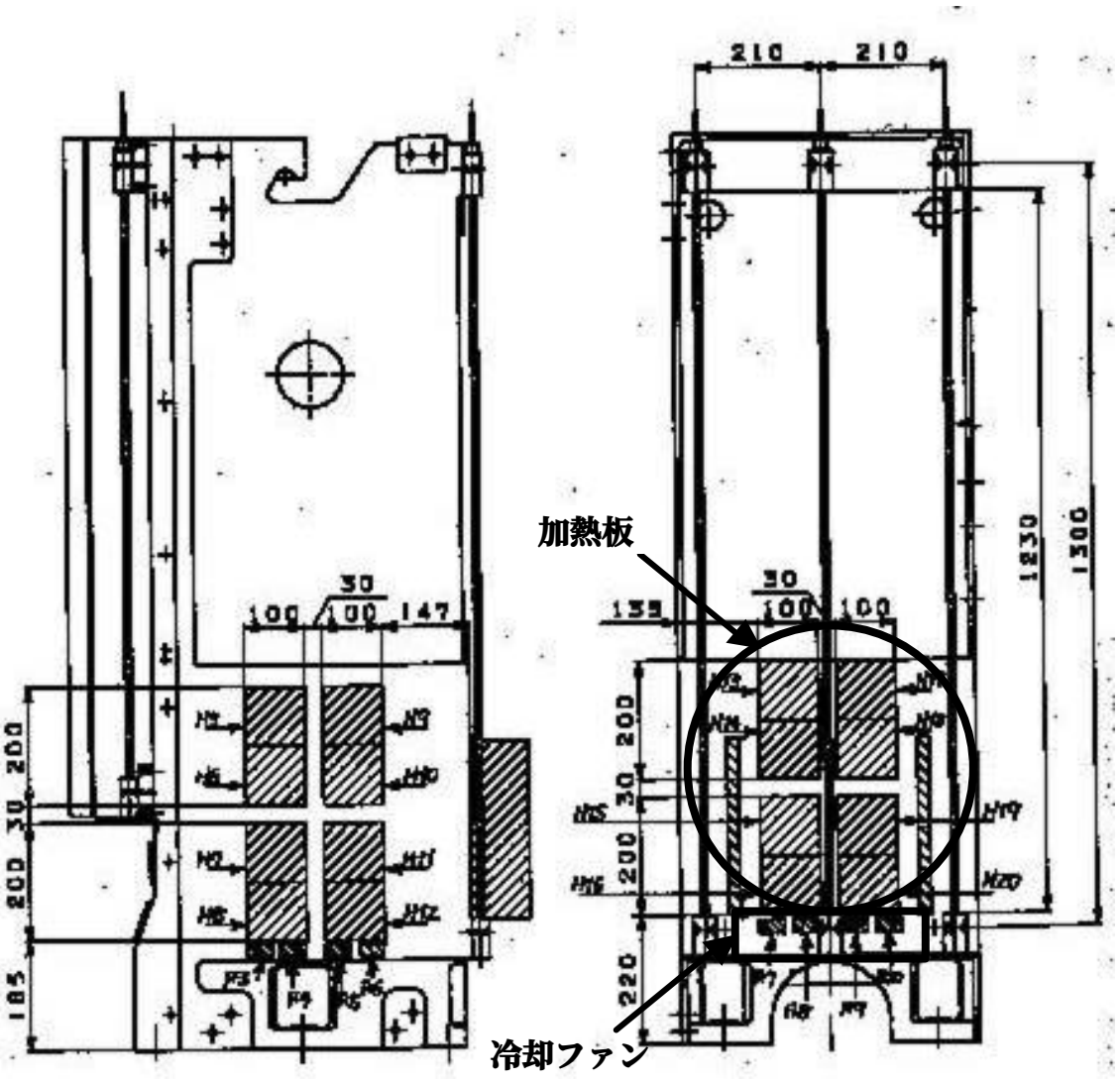

図 3-5　熱アクチュエータの設置図

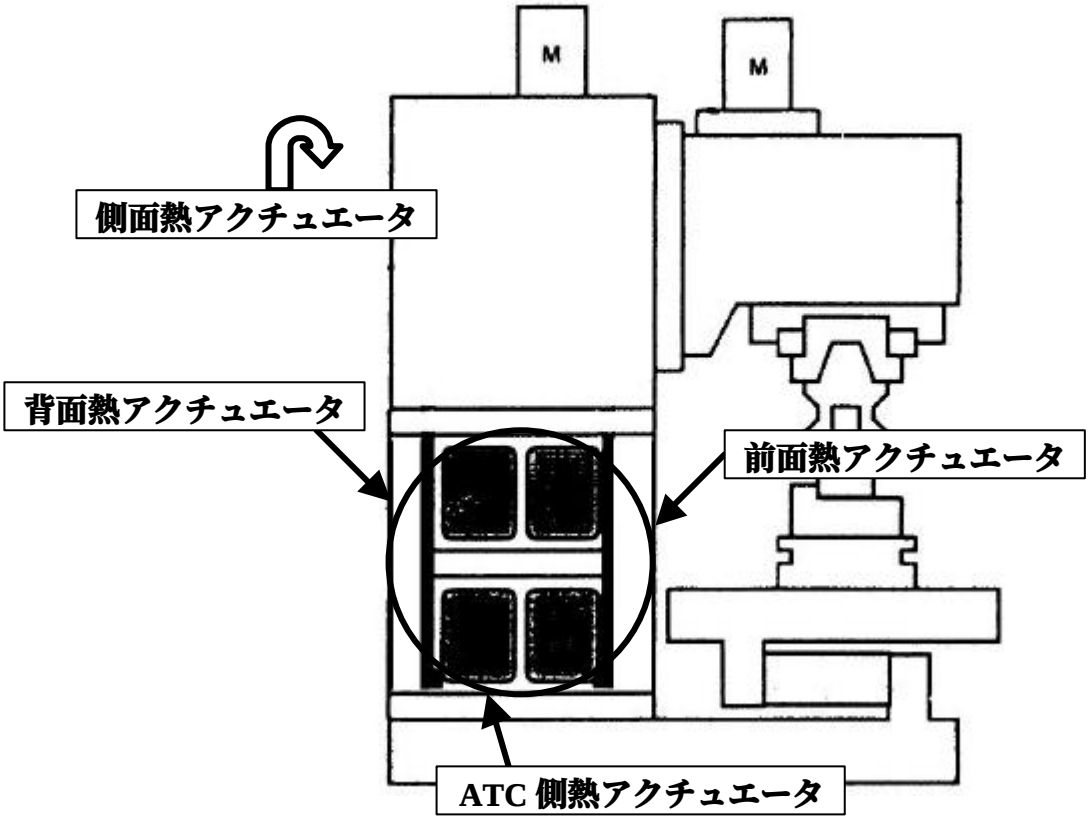

図 3-6　熱アクチュエータ配置図

熱アクチュエータは加熱板と冷却ファンがそれぞれ 14 個で構成されている。

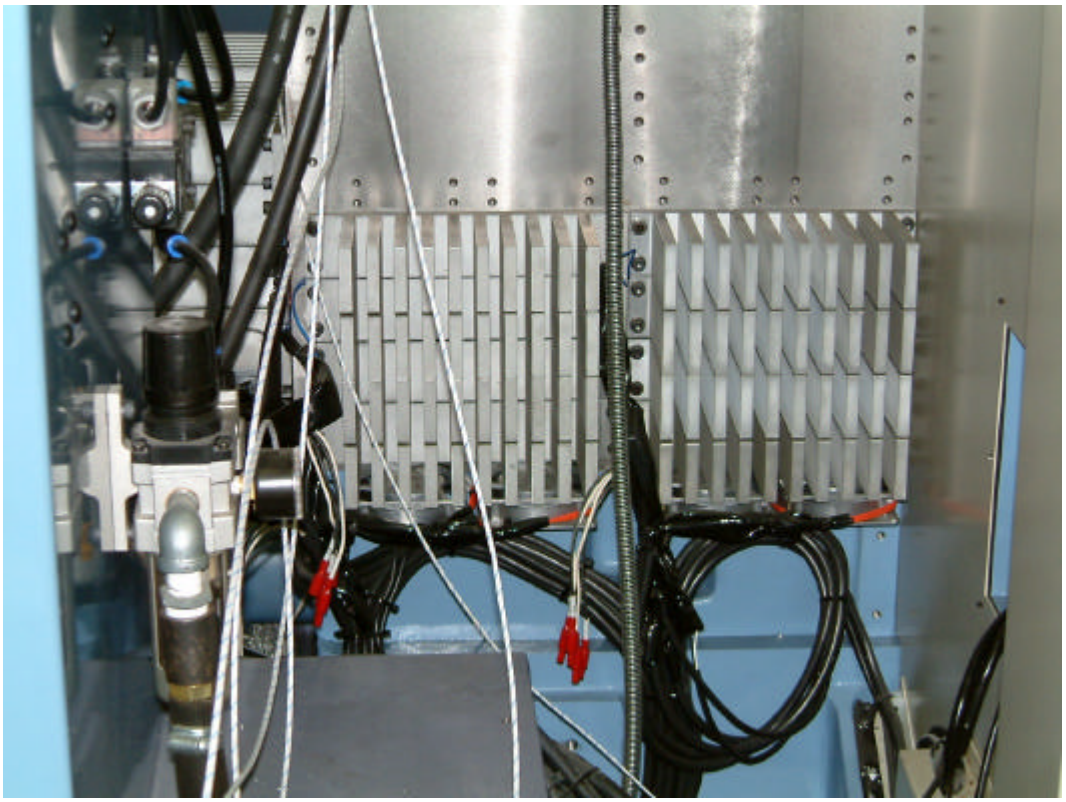

図 3-7　熱アクチュエータ

これらの熱アクチュエータを加熱、冷却することで種々の変形モードで構造体を変形させることが可能である。この変形を必要なモードで変化させることで、熱変形による主軸の傾きの変化を打ち消すことができる。

3-4 リモートコントロールアンプ

・図 3-8 に示すのは、NEC 三栄製リモートコントロールアンプ AH1108 である。特長として、コンピュータによる計測の自動化・無人化に必要なコンポーネントとして、確かな基本仕様に加え誰が使用しても同じデータが得られる信頼性、計測の容易さを実現するオートレンジ、セルフチェック機能の搭載などでシステム計測、フィールド計測に最適な多用途シグナルコンディショナである。また、その仕様を表 3-2 に示す。

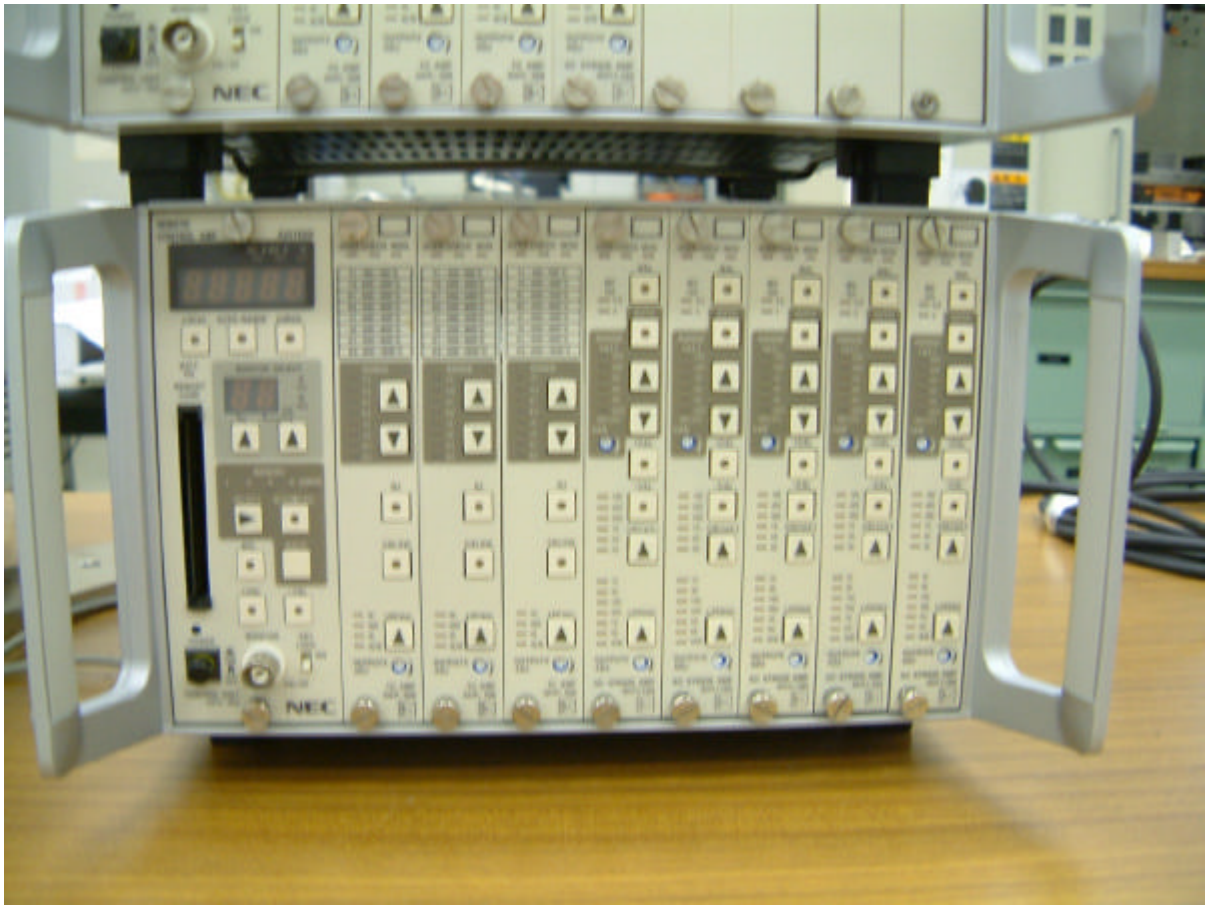

図 3-8　NEC 三栄製リモートコントロールアンプ AH1108

表 3-2　リモートコントロールアンプ AH1108

実装ユニット数	**8 ユニット（AH1108）**
表示部	表示桁数 4 桁 LED 変換回数約 3 回／秒 単位自動表示（V、ACV、℃）
メモリーカード部	シグナルコンディショナの設定値の記憶 再設定（4 通り）
モニタ出力部	モニタチャンネルセレクトスイッチにより選択されたチャンネルのアナログ出力が取り出せる。 同時に、表示部が出力される。
外部インターフェース	GP－IB、RS－232C　標準装置（同時使用不可） 外部インターフェースを利用して、ユニットの各設定、ステータスの読み出し可能（指定 CH、グループ、全 CH）
同期用出力	ブリッジ電源･･･電圧：２ Vrms　周波数：25KhZ 又は 5KhZ リモート機能／全チャンネル･･･オートバランス ±CAL、オートレンジ、セルフチェック
耐震性	29.4m/s^2 （3G）
耐電圧	AC1kV1 分間：コンディショナ入力端子～ケース間
使用温度湿度範囲	－10～40℃、20～80%RH 以内
電源	AC100V±10%（AC120、220、240V 切替え可） AH1108 約 80VA DC10.5V～15V･･･AH1108 約 4A（12V 時）
質量	AH1108：約 9kg（8 ユニット収納時）

※　1G=9.8m/s^2

3-5 ストレインアンプ・熱電対アンプ

・3-4 で示したリモートコントロールアンプをメインとして、NEC 三栄製ＡＣ ストレインアンプ AH11-104 と熱電対アンプ AH-11-109 を使用する。これらを用いて、変形センサの歪みゲージ出力を増幅し、各センサのデータからセンサ位置での実際の歪み量を測定する。また、AC ストレンアンプの仕様を表 3-3、熱電対アンプの仕様を表 3-4 に示す。

表 3-3　AC ストレインアンプの仕様

項目	使用
適用ゲージ抵抗	120Ω　～　1ＫΩ
ブリッジ電源	2V,0.5V 内部切替え AH11-104　正弦波 25 KhZ
平衡調整方式	抵抗分自動バランス、バックアップ約一ヶ月、容量分自動除去
平衡調整範囲	抵抗分±約 2%(±約 1000×10 ひずみ)
電圧感度	ひずみ入力にて 5V 以上(VAR 最大)
測定範囲	200,500,/FS,OFF(VAR 最大,BV=2V)
非直線性	±0.2%/FS 以内
周波数特性	DC～10KHZ±10%

表 3-4　熱電対アンプの仕様

項目	仕様
使用熱電対	T 形(CC)
測定温度範囲	T 形 T1:-150～150℃T2:-200～300℃
温度補償回路	誤差±2℃以内
リニアライザ回路	±0.5%/FS 以内
雑音	7.5℃P-P(K 形-200～1200℃レンジ)
零調整範囲	DC～10KHZ　+1,-3DB

3-6　A/D ボード

- A/D ボードには、インターフェイス社製の高機能 AD 変換ボード PCI-3155(図 3-9)を使用した。概要としては、プログラマブルインプットレンジ機能付きの超高機能、高精度高速 16 ビット AD 変換ボードです。仕様を表 3-5 に示す。

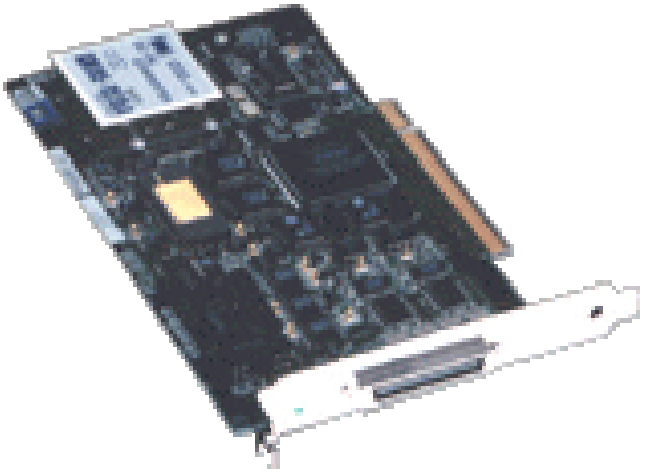

図 3-9　高機能 AD 変換ボード PCI-3155

表 3-5　高機能 AD 変換ボード PCI-3155 の仕様

入力チャンネル数	シングルエンド入力　16CH 差動入力　8CH
入力形式	マルチプレクサ方式
入力レンジ	ユニポーラ:+1V,+2.5V,+5V,+10V バイポーラ:±1V,±2.5V,±5V,±10V
入力インピーダンス	10MΩ(±5%)
入力保護	POWER ON 時　±35V POWER OFF 時　±20V
分解能	12 ビット
変換時間	1SEC(チャンネル固定) 5SEC(チャンネル切り替え)
相対精度	±3LSBMAX(A+25℃)
誤差	±0.2%MAX(0～50℃):全入力レンジ
絶縁方式	非絶縁

3-7　変形センサ

・ 一般に機械は、機械本来の機能を実現するための機構(仕事体)と、その部位を支える機構(構造体)からなる。仕事体で発生する力、熱、振動、音などは、ハウジング、フレーム、コラムなどとよばれる構造体に伝わり、それらを変形させる。この因果関係をあらかじめ知っていれば、逆に構造体に生じた変形(構造体変形)を測ることで、そのときの仕事体の運転状態を知ることができる。この構造体変形を測定するセンサとして「構造体変形センサ(変形センサ)」がある。これは、図 3-10 に示すように、構造体の二点間に固定された二個の弾性ブロックと、その二個の弾性ブロックを結ぶ剛体連結棒とからなる。この二個のブロックは、片方または両方に平行平衡構造を配していることが特徴であり、この構造によって図の連結棒の軸方向(X 方向)のみに柔に弾性変形できる。構造体が変形すると、連結棒は X 方向に剛であるから、図に示すようにブロックの平行平板だけが変形し、これを検出素子で測定する。今回の実験で使用した変形センサについて次の章よりくわしく説明する。

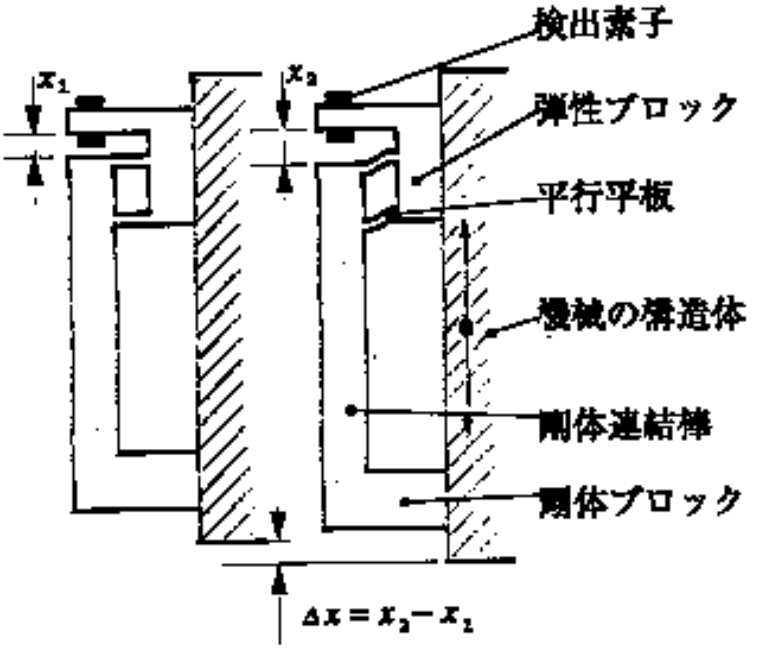

図 3-10　構造体変形センサ(変形センサ)

また、ストレインゲージの張り方は曲げ歪の検出でき引っ張り、圧縮歪を消去でき温度補償のある 2 アクティブゲージ法を採用する。

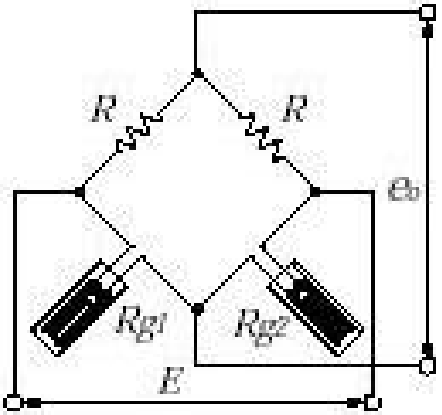

図 3-11　2 アクティブゲージ法の回路図

第四章
変形センサの設計

4-1　変形センサの主要機能

- **変形センサの主要機能は、**
  1,**構造体の二点間の微小変形をセンサ内の平行平板の変形に集中させる**
  2,**任意の方向の変形をそのほかの方向のそれと干渉させない**
  3,**力によって生じた変形と熱によって生じた変形とを分離する**
  4,**その変形を各種の検出素子で検出する**
  である。今回の実験には、これら四つの機能を取り入れた変形センサを開発した。次の項目でそれを表す。変形の検出素子として、安価で時間ドリフトが小さいひずみゲージを用いた。大形建造物では、構造体の変形を知るために、構造体表面に直接、ひずみゲージを貼ることが広く行われている。しかし、機械の構造体では一般に生じるひずみが小さいので、直接にひずみゲージを貼っても十分な出力が得られないことが多い。

4-2　変形センサの図面(設計)

- 今回の実験のために設計した変形センサの図面を CAD を用いて図 4-1 から図 4-3 に表す。
  なぜ、こういった図面、形になったかを、次の項目で表す。

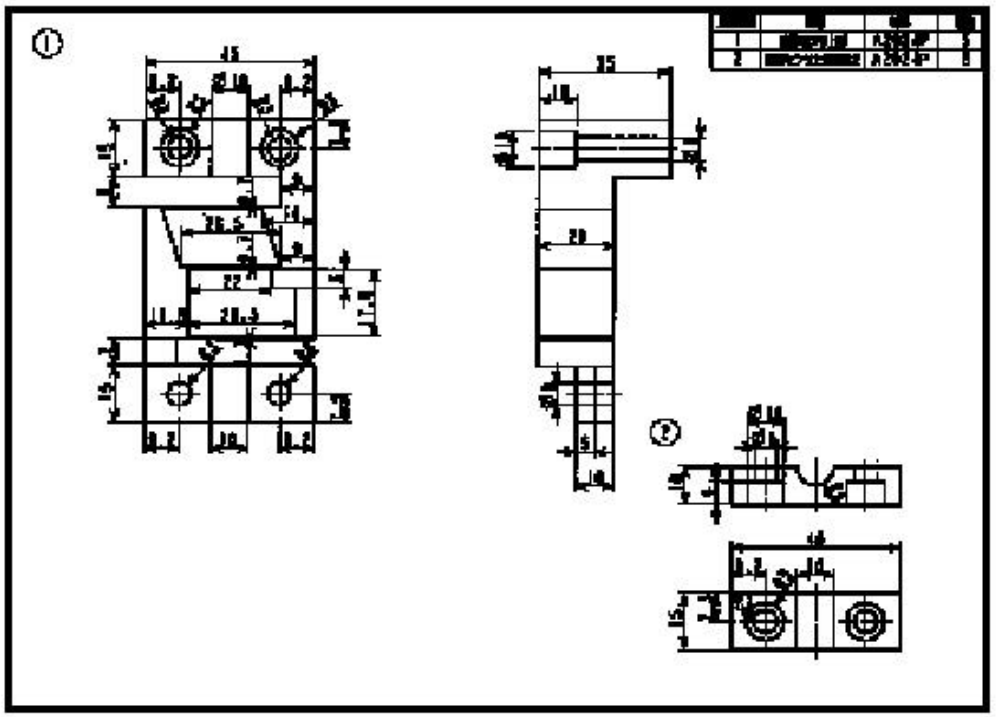

図 4-1　変形センサ(上部)

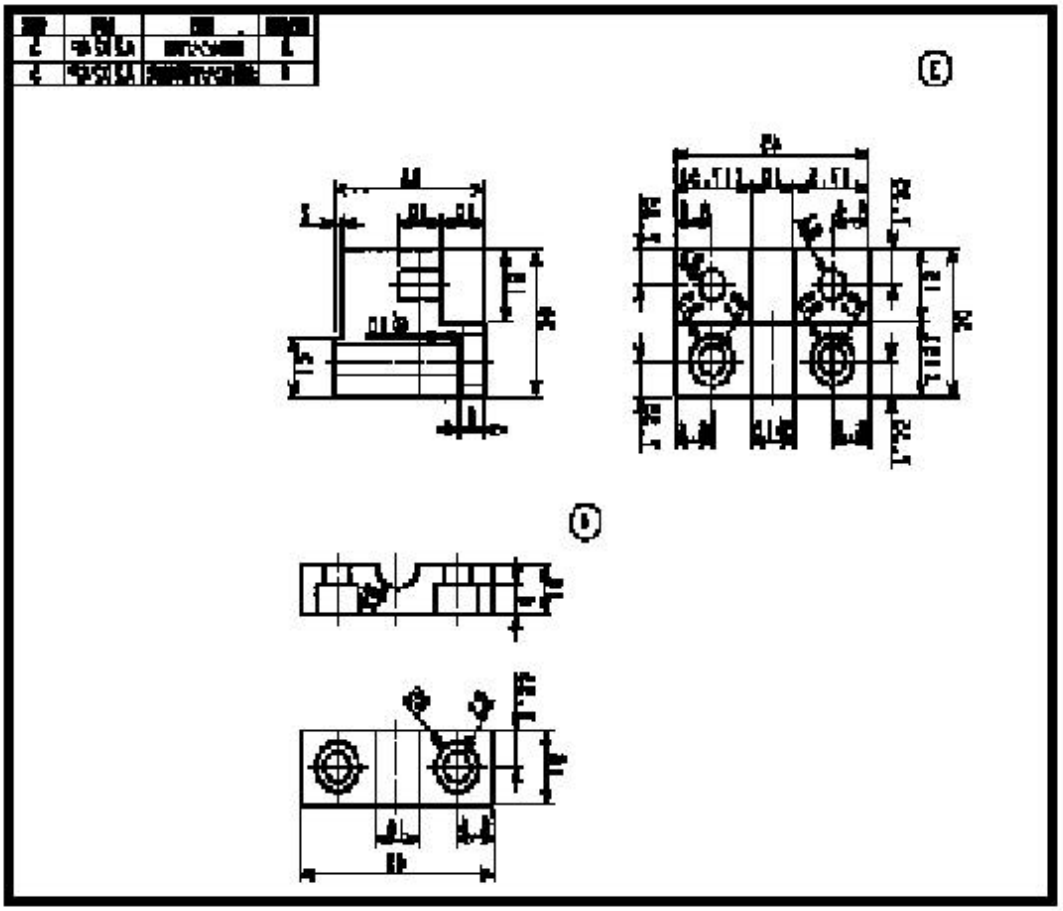

図 4-2　変形センサ(下部)

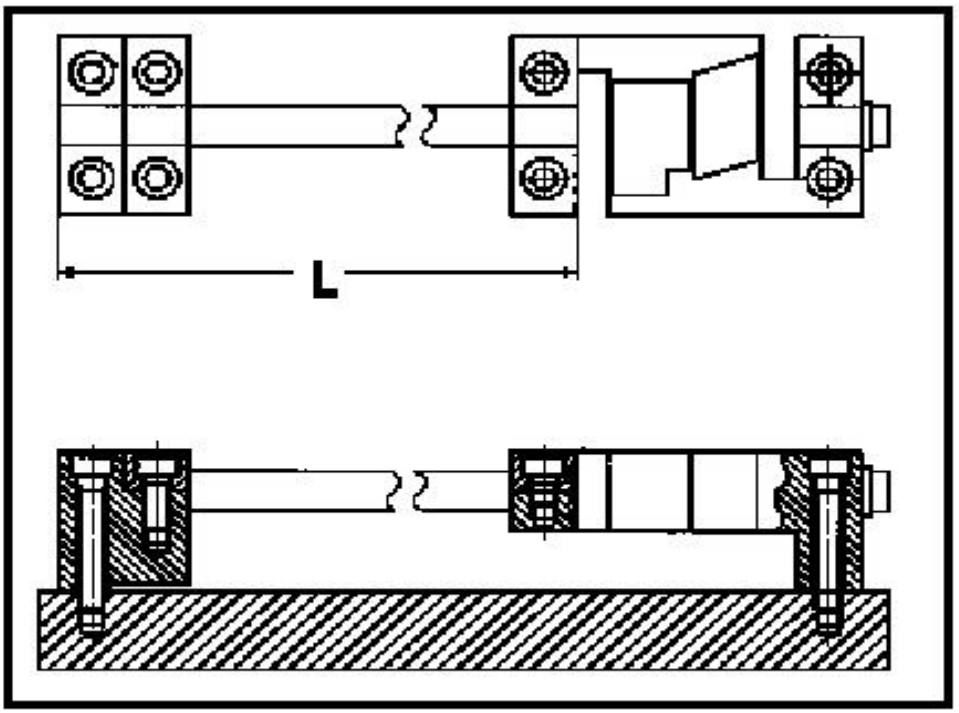

図 4-3　変形センサ(全体図)

ここの変形センサをつなぐ材質にスーパーインバーを仕様する。また、マシニングセンタに設置するため、全長(L)が 4 種類、全部で 5 個の変形センサを作成する。全長の種類は 218 ㎜(A),458 ㎜(B),558 ㎜(C),1023 ㎜(D,E)とする。

4-3　変形センサの説明

- **変形センサの検出素子として、安価で時間ドリフトが小さいひずみゲージを用いた。大形建造物では、構造体の変形を知るために、構造体表面に直接、ひずみゲージを貼ることが広く行われる。しかし、機械の構造体では、一般に生じるひずみが小さいので、直接にひずみゲージを貼っても十分な出力が得られないことが多い。一方、この変形センサは、構造体表面上で局所的にひずみを測っても検出できないほど小さい変形でも、ブロックの低剛性部にひずみを集中させて、変形を効率よく検出できるように設計した。また、変形の干渉防止のために、センサの弾性ブロックに平行平板構造を採用した。平行平板構造とは、可動部と固定部とを二枚の薄板で結んだ構造である。力やモーメントによって両薄板の表面に直行する方向に生じる力によってその方向にのみ変形し、薄板の両端の根本表面には引張りおよび圧縮のひずみが生じる。このようにして、平行平板構造は一軸方向の力を、ほかの力やモーメント成分の干渉を受けずに検出することができる。力変形と熱変形の分離は、変形センサの材質を変えることで実現できる。つまり、変形センサの材質を構造体のそれとまったく同じにすると変形センサには、両者の温度が同じときは、熱膨張差による変形が生じないので、力による変形だけが検出できる。一方、両者の材質が異なると熱膨張差が生じるので、力変形だけでなく、熱変形も検出できる。**

4-4　変形センサの開発

- 変形センサの開発には、前述に述べたことを含めて開発した。今回の開発には、オープン CNC マシニングセンタと三菱社製ワイヤカット放電加工機(SX10)を使用した。マシニングセンタのプログラム、ワイヤカット放電加工機のプログラム、共に付録に記載する。次に、ワイヤカット放電加工機を図 4-4 に表し、使用を表 4-1 に表す。

図 4-4 三菱社製ワイヤカット放電加工機　SX10

最大工作物寸法	800×575×215
工作物最大質量	530
テーブル寸法	590×514
各軸移動量	350×250×220
テーブル早送り速度	1300
使用ワイヤ径	0.07～0.3
最高ワイヤ送り速度	250
ワイヤ張力	0.5～25
外形寸法	1235×1570×2270
機械本体質量	1600

表 4-1　三菱社製ワイヤカット放電加工機　SX10

実際に設計・開発した変形センサを図 4-5 に示す。

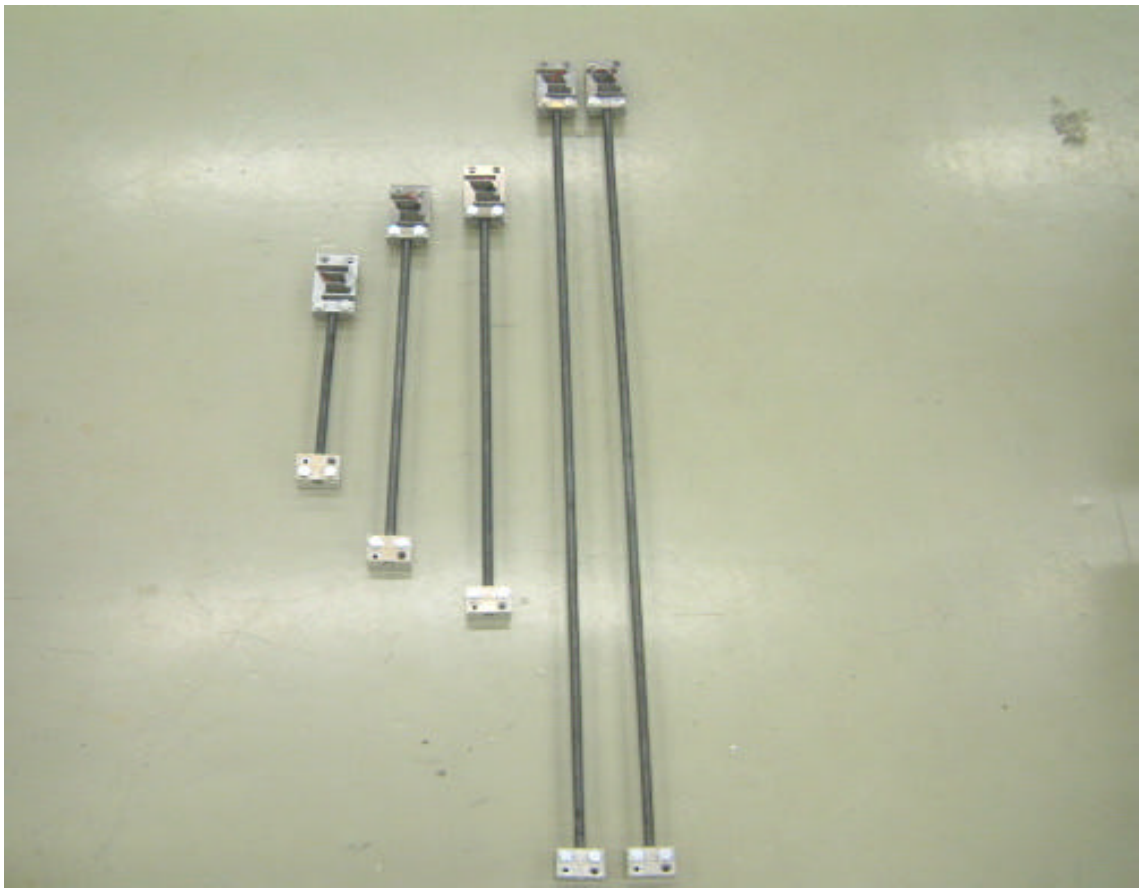

図 4-5　変形センサ

第五章
変形センサの
性能評価

5-1　実験装置の設計・開発

・今回の実験でより正確なデータがとれるように実験装置を設計・開発した。その概要をまず図 5-1 に示す。

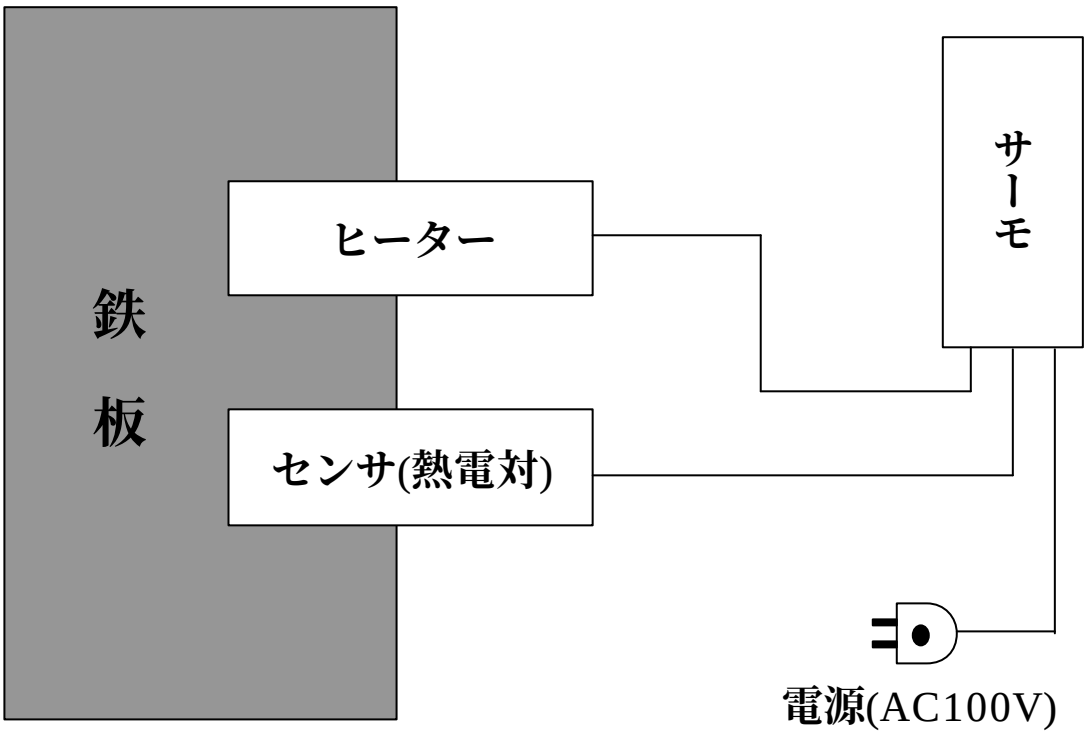

図 5-1　実験装置の概要

・ この実験装置はマシニングセンタにより近い状態を保つために、加熱板、鉄板、センサ(熱電対)を使用した。これらは、熱変形、熱歪みなど、実際にマシニングセンタにどういったことが起こっているかを比べやすく、また、他のいろいろな実験にも応用できる。この実験装置に使用したものをこれから説明する。

・マシニングセンタにおける熱アクチュエータの代わりに八光商事株式会製のシリコンラバーヒーターを使用した。実験装置自体の大きさ、重量収納、価格などを考慮し、いままでの金属ヒーターに代わる柔軟性がありまた薄型の面状発熱体を使用した。また値段も低価格である。このシリコンラバーヒーターを操作、制御するために、八光商事株式会社製のデジタルファインサーモ DG2 を使用した。特長として、ヒーターと電源をつなぎ、センサを入れるだけで温度コントロールできる。また、広い温度をカバーすることができ、設定可能な温度範囲は、0℃～600℃まで 1 レンジでカバーすることができる。あと、熱の測定のために八光商事株式会社製の熱電対 HT-150(フォーク端子取付タイプ)を使用する。特長としては、測定できる温度領域が広いことである。極低音の-269℃(金・鉄-クロメル)から、高温では 1600℃(白金-白金・ロジウム)と広い範囲をカバーすることができる。また、接触式の温度センサーとしては最も高い温度で使用することができる。こういった特長などを考慮し、実験装置の設計・開発とする。これらの仕様を表 5-1 から表 5-3 に示す。実際に設計・開発したものを図 5-2 に示す。

連続使用温度	200℃
最大使用温度	250℃
寸法　最大	幅 500 ㎜×3000 ㎜
寸法　最小	幅 25 ㎜×50 ㎜
寸法　厚さ	1.5 ㎜
寸法　リード線長さ	300 ㎜
備考	表面シリコーンゴム

表 5-1　シリコンラバーヒーターの仕様

被覆熱電対種類	ガラス編組被覆　0.65 単線
測温接点	接地形
常用限度	200℃

表 5-2　熱電対の仕様

型番	DGC1150
入力電圧	AC100V(50/60Hz)
最大負荷	1.5kW(抵抗負荷)
温度設定範囲	0℃～600℃
温度表示精度	±(1%F.S.+2.5℃)
室温補正精度	±3℃
制御方式	ON/OFF 制御
使用環境	0℃～40℃(温度 85%以下)
センサー	K タイプ熱電対
外形寸法	幅 66 ㎜×長さ 210 ㎜×厚さ 33 ㎜

表 5-3　デジタルファインサーモ DG2 の仕様

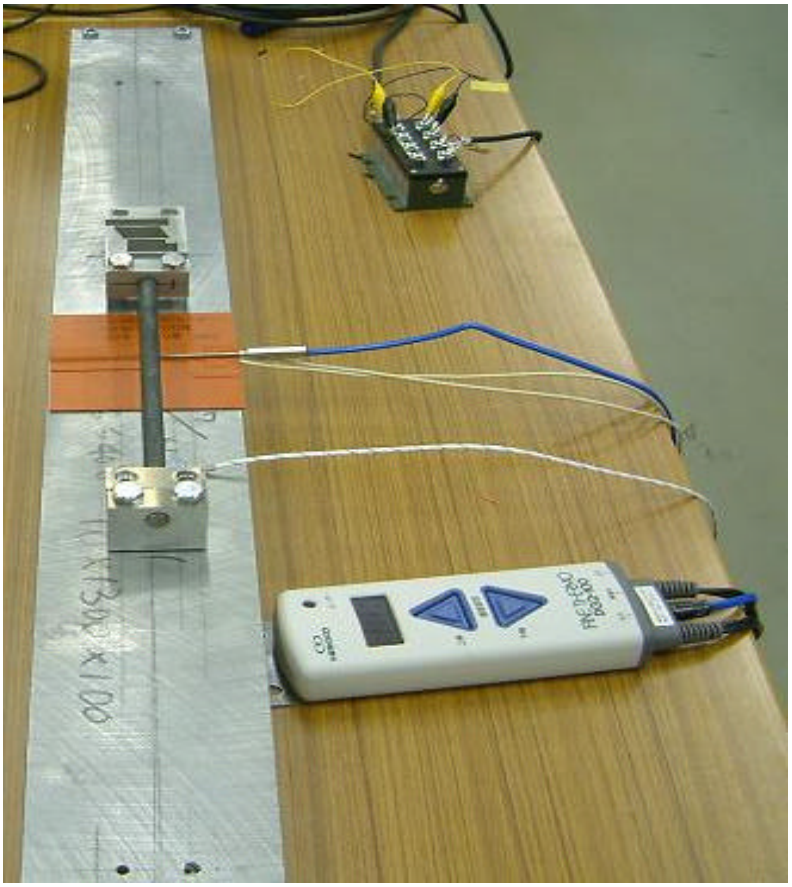

図 5-1　実験装置

5-2 歪測定プログラム

・今回歪を測定するために、VisualC++を使用してデータ測定プログラムを作成した。操作画面を図 2.14 に示す。測定条件を変えられるようにし、データをメモ帳に保存できるようにした。

A／D変換ボードから取得するデジタル値をまず電圧に変えなければならない。

電圧＝（レンジ上限－レンジ上限）×デジタル値 / 分解能＋レンジ下限　　（2.1）

今回はレンジ±5Vで分解能が 65536 であるので

電圧＝10×デジタル値 / 65536－5　　（2.2）

という式が成り立つ。

次に電圧から歪を求める。

$$e = (K \times e \times \text{ゲージ法}) / 4 \qquad (2.3)$$

ここで、K はゲージ率、e はひずみ量（10^{-6}）、E はブリッジ電圧である。

ゲージ法は 2 アクティブゲージなので、パネル表示校正値×1/2 である。

また今回は変形センサの性能テストの際に熱電対を使用して温度変化もはかるのでデジタル値から温度を求める。今回は－150℃～150℃までを測定できるようにする。レンジは±5Vで、分解能が 65536 である。

温度＝30.0×デジタル値－32768 / 65536　　(2.4)

この計算式をもとに変形センサ 5 本分、5 チャンネルの測定を行うことのできるプログラムを作成。

5-3　実験(実験装置)

・いよいよ実際に実験を行っていく。まずは、実際に変形センサが正常に動くかどうかを変形センサ、ブリッジボックス、ストレインアンプ、実験装置を使い確かめた。数値が安定するか、歪が発生したとき、数値は変化するのかといった点を調べた。結果、変形センサ 5 個すべて正常に動いた。今回の実験は、シリコンラバーヒーターを使い、実験装置を 30 分加熱しその後、自然冷却 30 分行う。そのデータを次に表す。また、これらのデータの Strain[μst]が－の数値になるのは、実験装置のシリコンラバーヒーターを変形センサの中央に設置したため、鉄が熱変形で膨張し、変形センサを引っ張るという現象が起こったからである。熱変形で膨張した分をαとし、図 5-2 に示す。

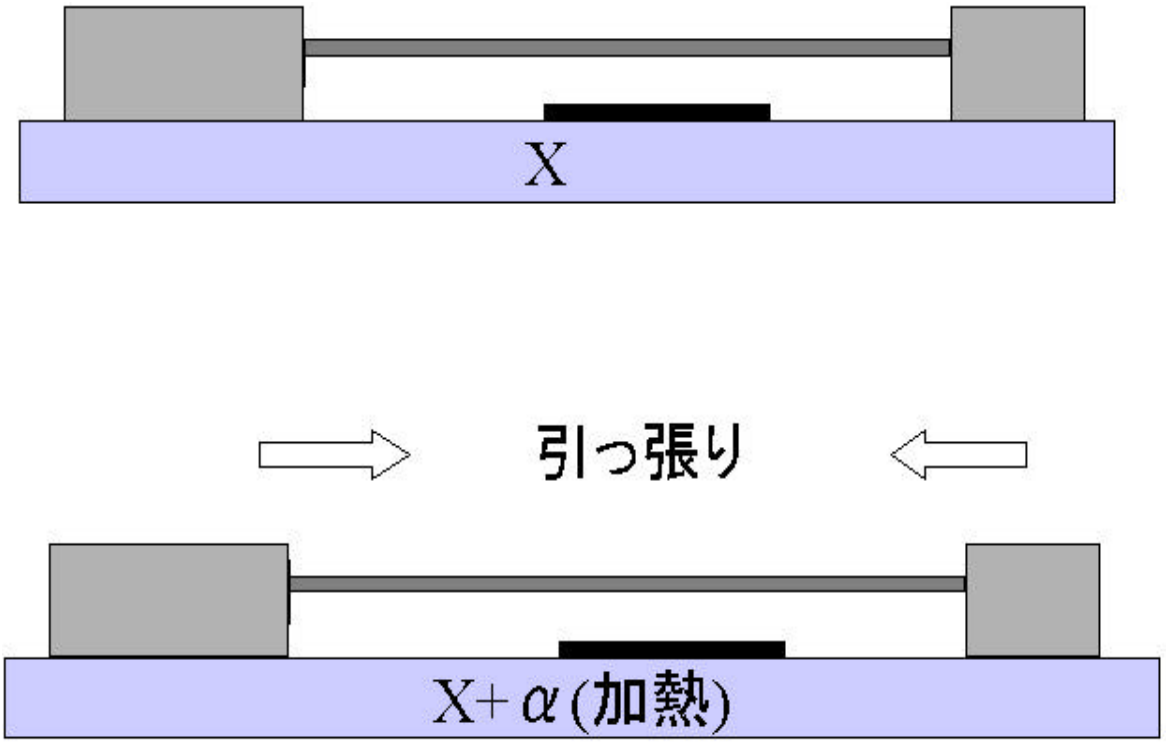

図 5-2　実験装置の熱変形

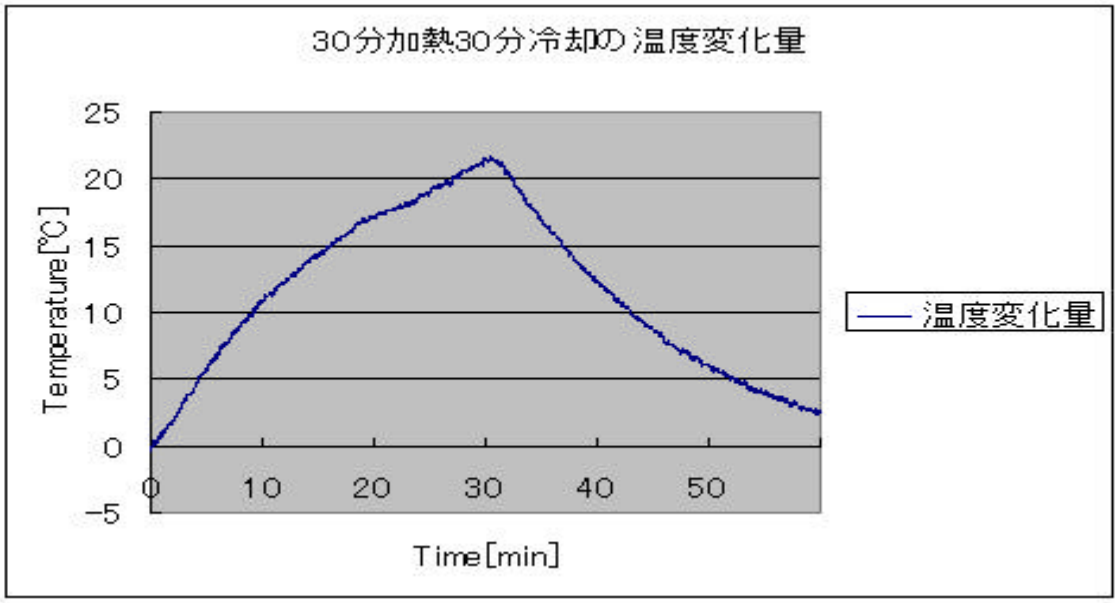

図 5-3　30 分加熱 30 分冷却の温度変化量

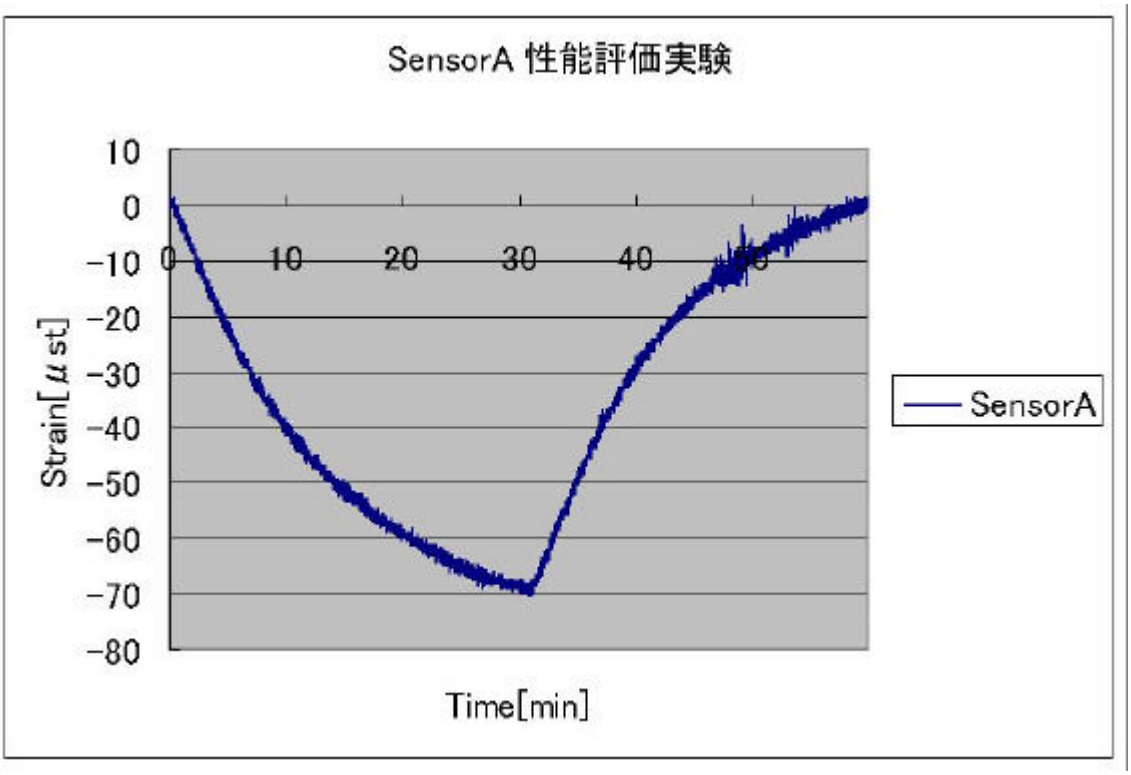

図 5-4　センサ A 性能評価実験

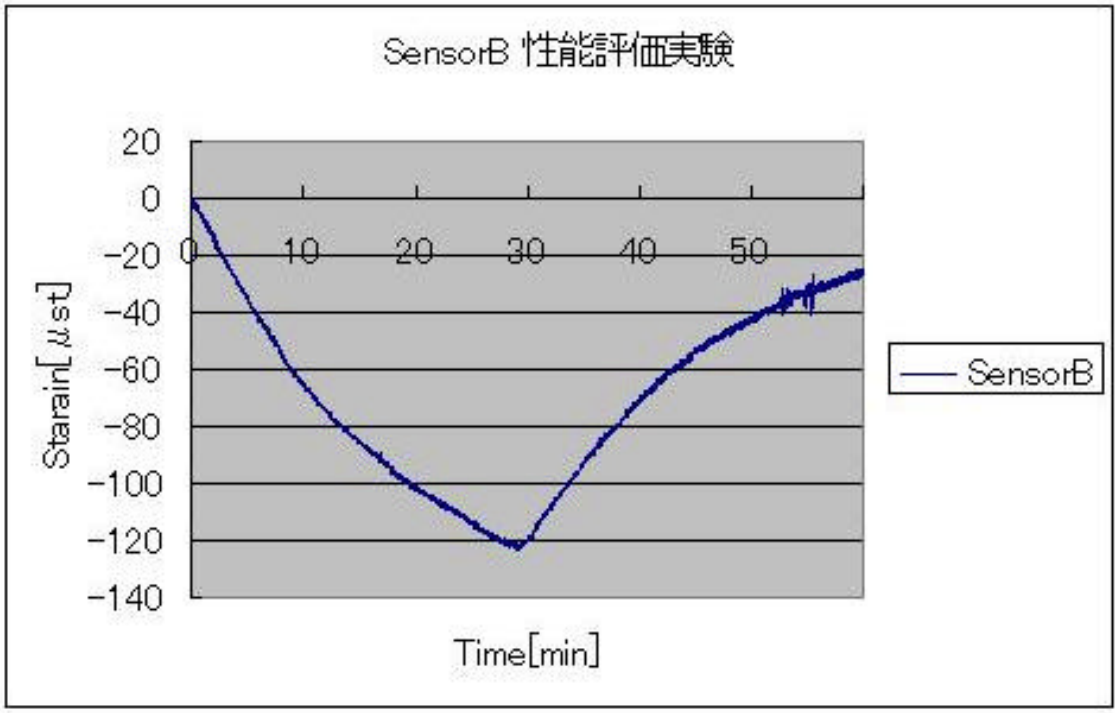

図 5-5　センサ B 性能評価実験

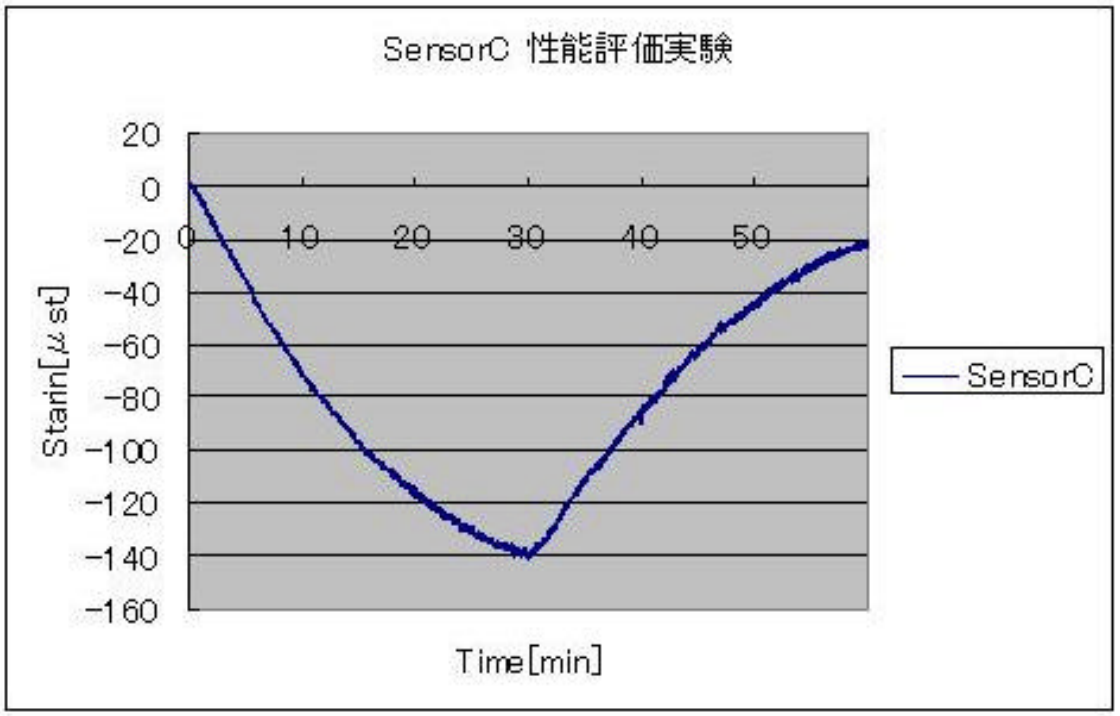

図 5-6　センサ C 性能評価実験

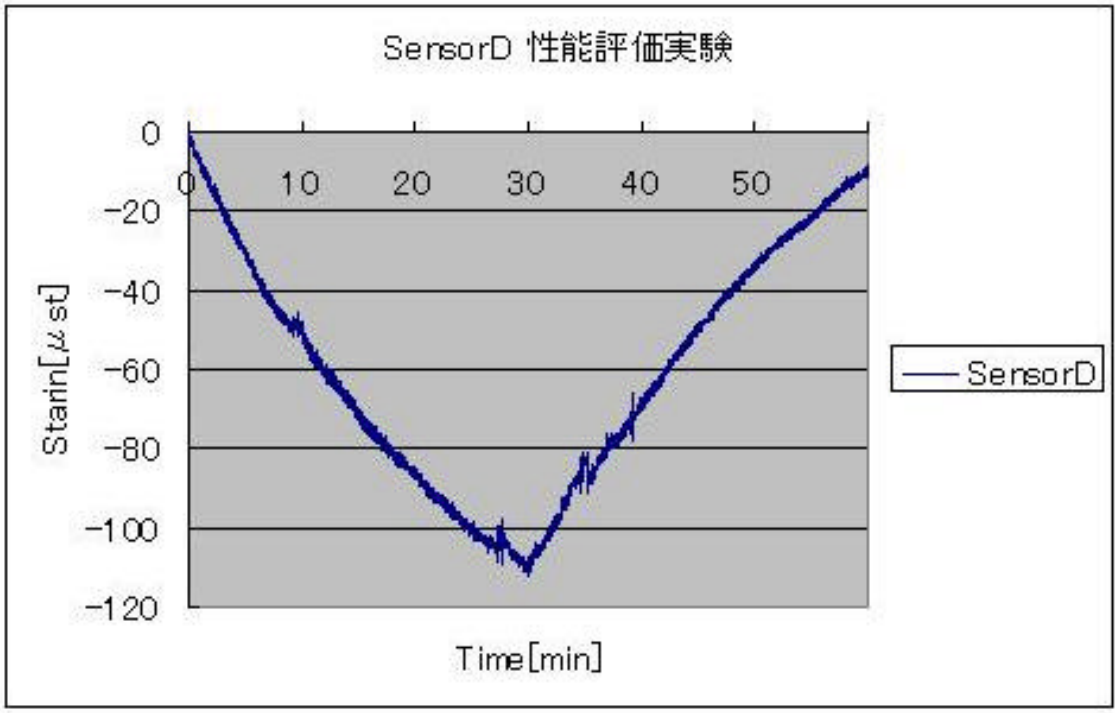

図 5-7　センサ D 性能評価実験

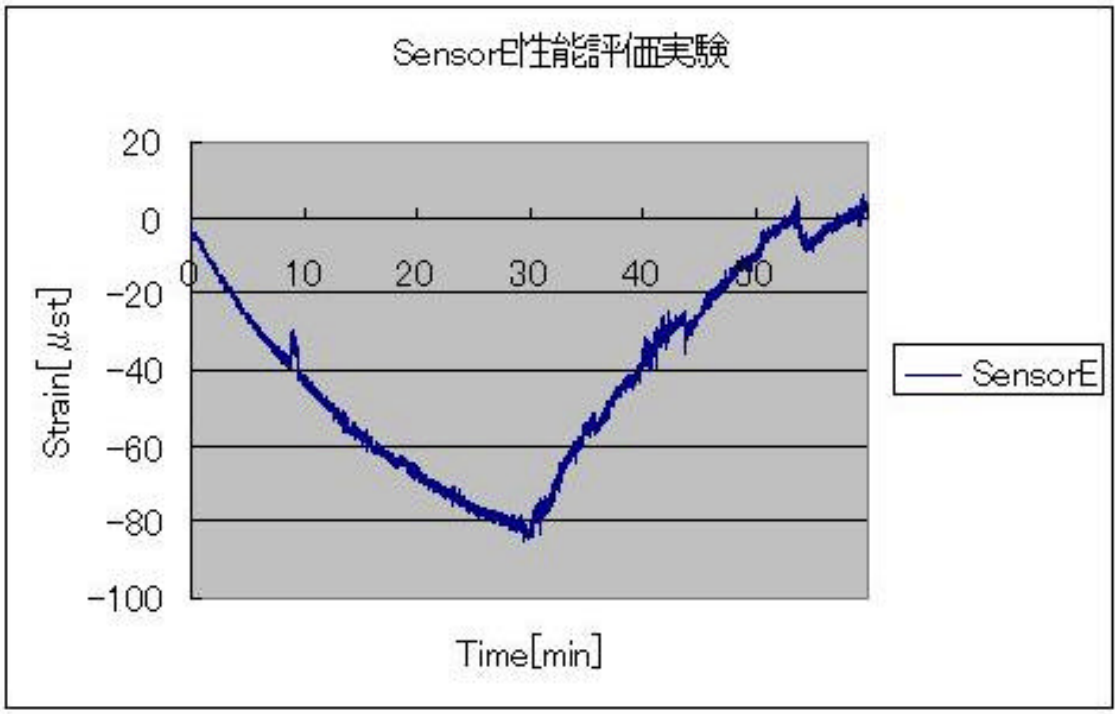

図 5-8　センサ E 性能評価実験

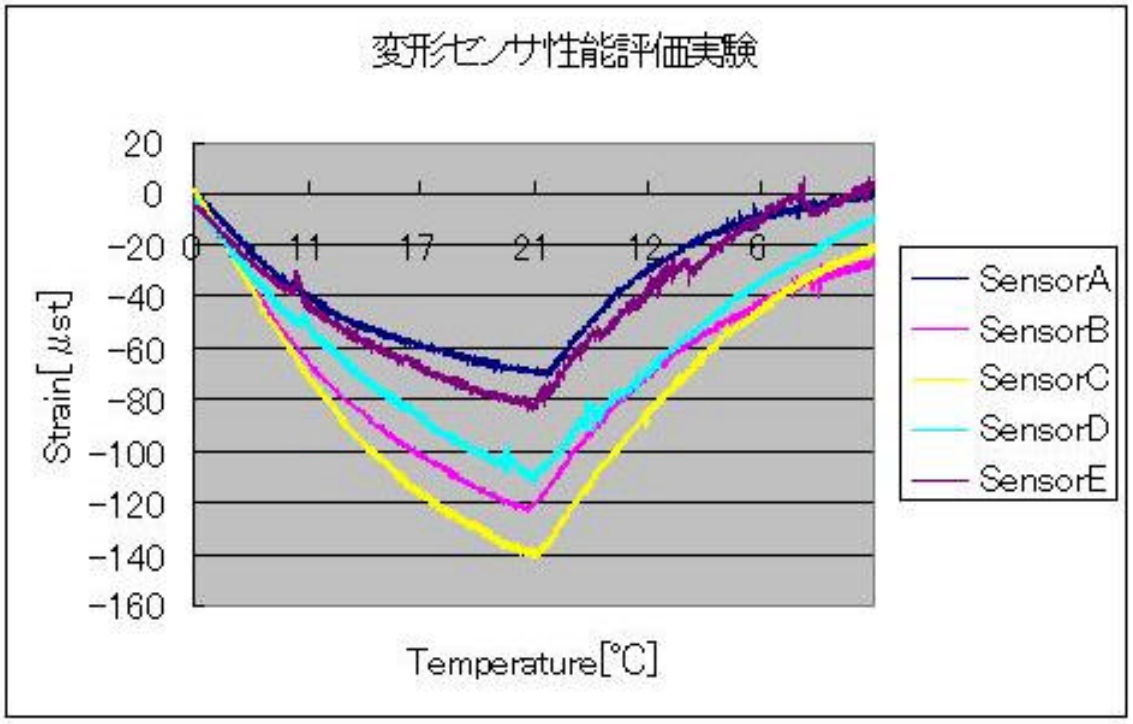

図 5-9　変形センサ性能評価実験

以上の結果により 5 本の変形センサは実験に必要な性能を持っているといえる。

5-4　実験(マシニングセンタ)

・ 次は変形センサをマシニングセンタに設置し、熱アクチュエーターを使い 30 分加熱 30 分冷却という実験を行う。この実験方法は実験装置と同じである。マシニングセンタの設置されている 4 個の熱アクチュエーター(ATC,前面,側面,背面)を動かし、それぞれのデータを測定する。まずは変形センサのマシニングセンタへの設置図を図 5-9 に示す。

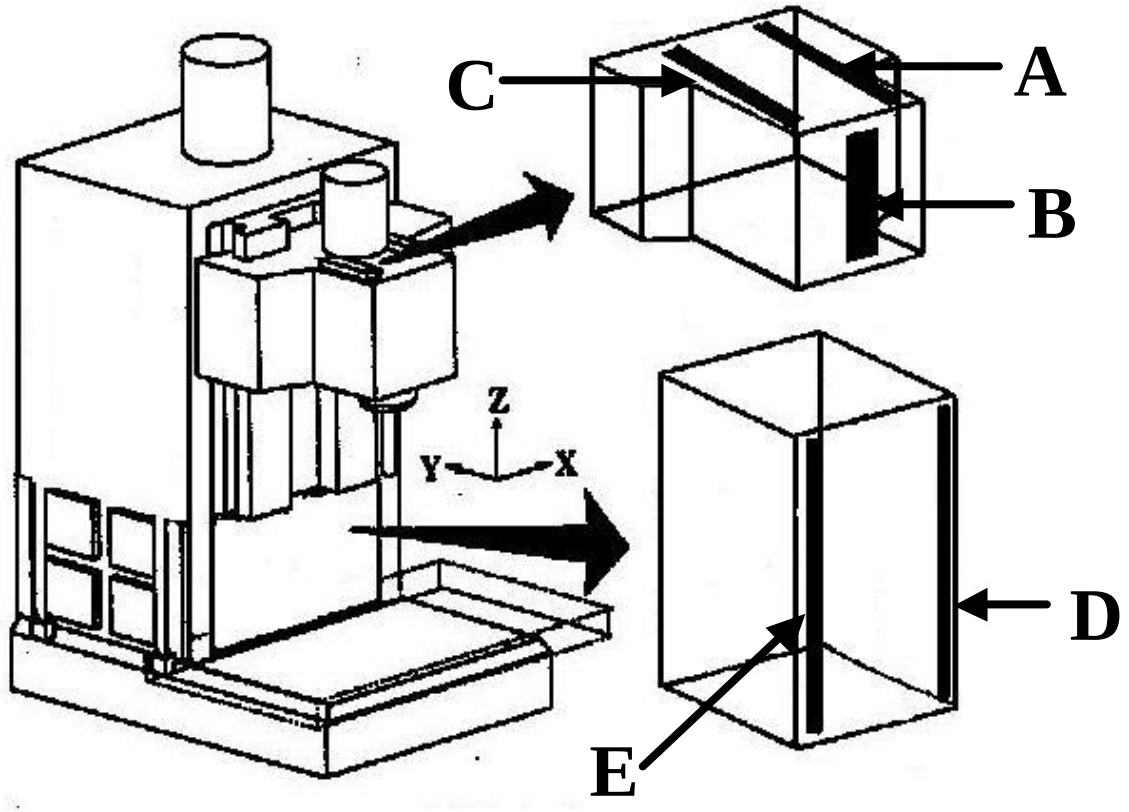

図 5-9　変形センサのマシニングセンタへの設置図

・上の図の A,B,C,D,E が変形センサの取り付け位置である。

・次に設置した変形センサ全てのデータを次から示す。

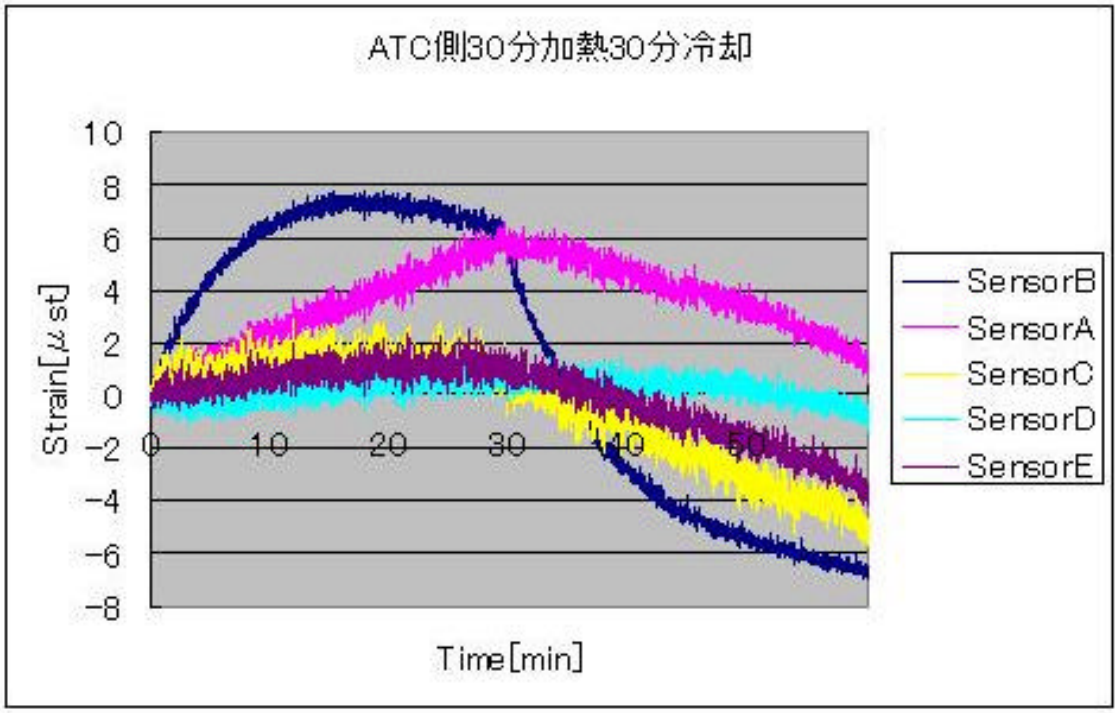

図 5-10　ATC 側 30 分加熱 30 分冷却

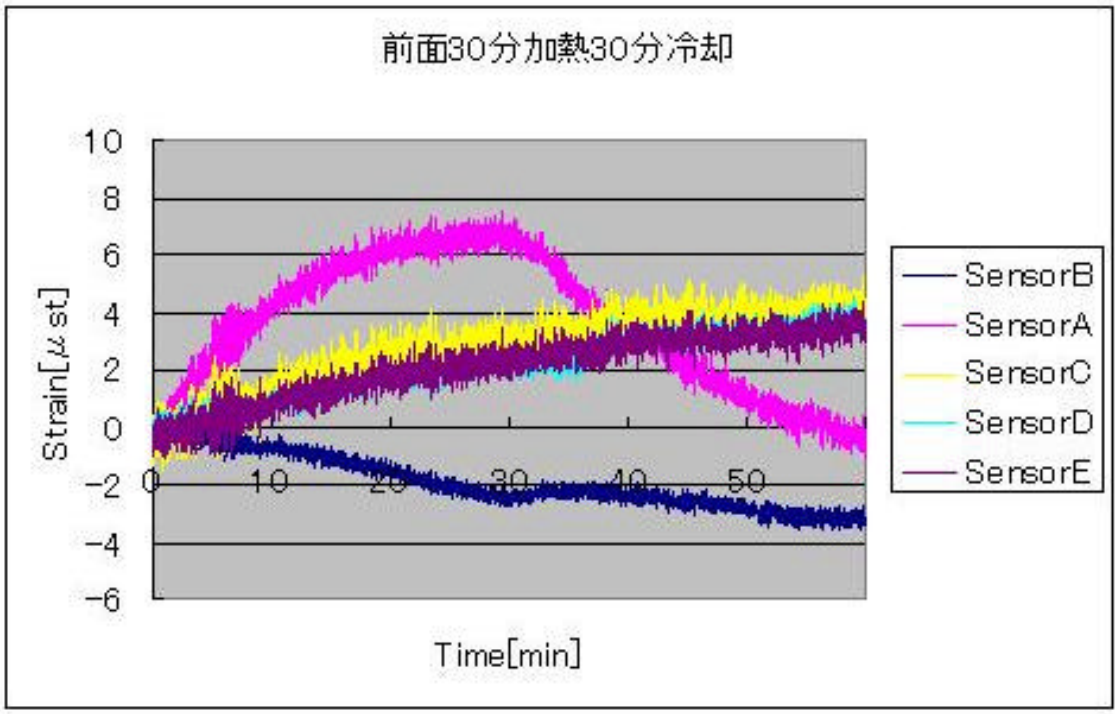

図 5-11　前面 30 分加熱 30 分冷却

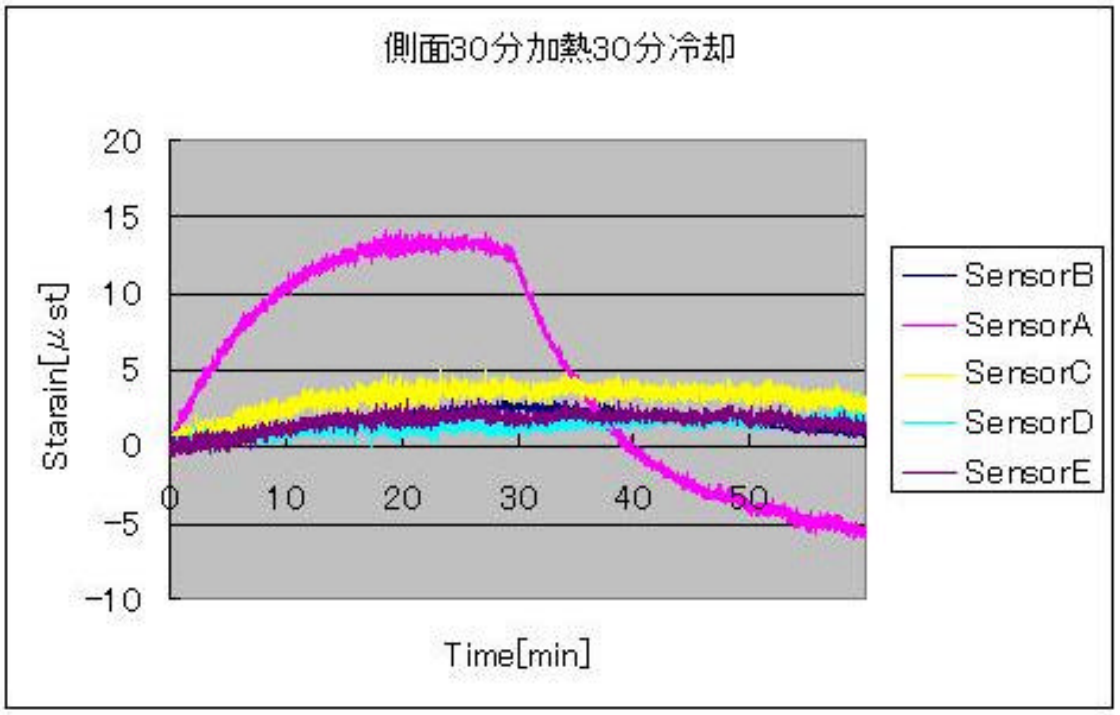

図 5-12　側面 30 分加熱 30 分冷却

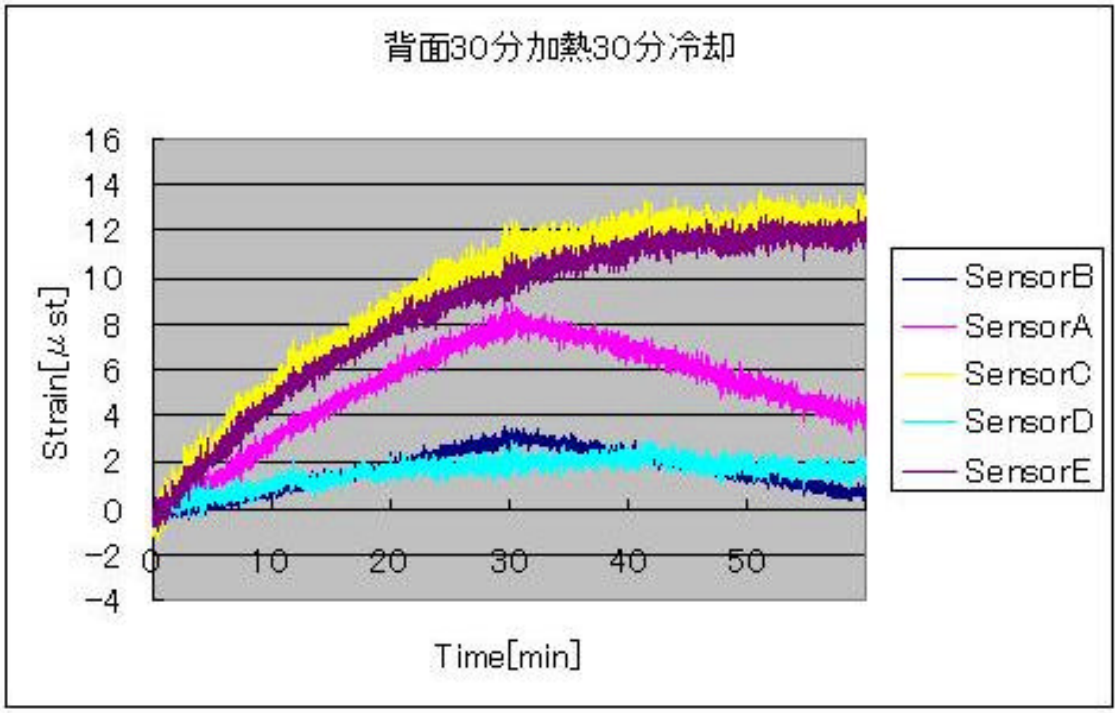

図 5-13　背面 30 分加熱 30 分冷却

第六章
結果

6-1　結果

- **結果として、マシニングセンタの熱変形による誤差を補償・修正するまで到達することはできなかった。変形センサの性能としては満足できるものができた。マシニングセンタへの設置をもう少し考慮できたらよかったと思う。マシニングセンタと実験装置の実験で誤差が生じたのは、マシニングセンタに設置された熱アクチュエーターの精度、冷却ファンの精度が悪いと考えられる。熱アクチュエーターの数を増やすことにより、その誤差は解消できると考えられる。実際に加工中のデータを測定できればよかったと思う。これからの目標とすれば、これらのデータを測定し、マシニングセンタの熱変形による誤差を補償・修正するプログラムを開発していければと考える。**

第七章
総括と展望

7-1　総括

・変形センサを個人で開発するということを簡単に考えすぎていたように思う。実際、開発をはじめ完成までにかなりの時間がかかった。その次に実験装置の開発にも時間がかかった。実際に変形センサをマシニングセンタに取り付け、全長を測り、一旦、固定する。そこでまた問題が発生した。マシニングセンタのカバーを外すと、変形センサの取り付け予定の場所にコードが邪魔で取り付けの変更をしなければならなかった。そうしたことをふまえて実験装置を開発した。まず、実験装置で変形センサが正常に動くか、ストレインアンプが動くか試した。そして、30 分加熱 30 分冷却するという実験を行った。次に、変形センサに取り付け、熱アクチュエータを使い、30 分加熱 30 分冷却を行った。こういった実験を行い、みえてきた点などを次の展望で述べる。

7-2　展望

・総括で述べた点などをふまえて展望とする。まずは変形センサを考える。
1,歪ゲージの取り付け位置をもっと取り付けやすくするべき
2,マシニングセンタに取り付けることを考えての形の変更
が、考えられる。次にマシニングセンタを考える。
1,熱アクチュエータの精度を上げる
2,熱アクチュエータの数を増やす
3,冷却ファンの精度を上げる
4,冷却ファンの数を増やす
といった点が考えられる。

こういった点を改善することにより熱歪変形を制御することに近づくと考えられる。今回できなかった試みとして、実際にものを加工したときの熱歪変形のデータを取ってみるといったことが考えられる。そのデータをもとに熱変形を予測するシステムを作り、主軸の傾きを制御できるような遠隔制御システムが必要になっていくだろう。具体的には、有限要素法による構造シミュレーションにより、幾何学的な関係を明らかにすることや、それに、熱流束や熱変形を加味したモデルが考えられる。こういった点を解決し、熱変形の情報を基に主軸の傾きや歪みを制御することのできる高精密マシニングセンタの開発が課題となっていくだろう。

謝辞

謝辞

- **本研究を進めるにあたり、多くの方々に御指導、御協力いただきました。指導教官である長尾教授には日頃から親身な御指導を頂きました。おかげでこの二年間を大変有意義なものにすることができました。**
  院生の更谷さん、木崎さん、上條さんには、たいへんお世話になりました。マシニングセンタの基礎から普段の生活までいろいろ迷惑かけました。小林研究室の皆様にもたいへんお世話になりました。おかげで楽しい二年間を過ごせました。特に橋本さん、小山くんには変形センサの加工をしていただき助かりました。最後に、学部の高谷くんとたいへん感謝しています。更谷さんと高谷くんと共に、卒業研究ができ光栄に思います。最後まで迷惑かけました。長尾研究室、小林研究室のみなさんと有意義な時間を共有できたことをうれしく思います。ありがとうございました。

参考文献

参考文献

・田口将,変形情報に基づく姿勢制御によるマシニングセンタの高精密化の研究
(1991 年東京大学大学院工学系研究科産業機械工学専攻修士論文)

・堤和久,熱変形能動補償型マシニングセンタによる加工精度の研究
(1996 年東京大学工学部産業機械工学科卒業論文)

・花山良平,熱変形能動補償型高精密マシニングセンタによる
加工精度向上の研究
(1998 年東京大学工学部産業機械工学科卒業論文)

・センサ活用 141 の実践ノウハウ　松井邦彦著　CQ 出版社

・知能化生産システム 長尾高明 畑村洋太郎 光石衛 中尾政之著 朝倉書店

・ひずみゲージによるひずみ測定入門　歴史から測定まで
高橋　賞・河合　正安　著　　大成社

付録

付録

変形センサを作るまでの加工プログラム

１：マシニング・センタのプログラム

インバーの設置穴あけ加工。

```
O1805
    G90 G00 G54 X0. Y0. Z0.;
    G43 H12;
    S1000 M03 M08 F60;
    G90 Z0.;
    G00 X22.5 Y25.0;
    G01 Z-28.0;
    G01 Z0.0;
    G00 X0.0 Y0.0;
    M05 M09;
    M02;
```

固定端の穴あけ加工。

```
    O1806
    G90 G00 G54 X0.0 Y0.0 Z0.0;
    G43 H12;
    S1000 M03 M08 F60;
    G90 Z0.0;
    G00 X8.6 Y72.65;
    G01 Z-47.0;
    G01 Z0.0;
    G00 X36.4 Y72.65;
    G01 Z-47.0;
    G01 Z0.0;
    G00 X25.0 Y50.0;
    G01 Z-47.0;
    G01 Z0.0;
    G00 X19.25 Y31.475;
    G01 Z-47.0;
    G01 Z0.0;
    G00 X8.6 Y7.35;
    G01 Z-47.0;
    G01 Z0.0;
    G00 X36.4 Y7.35;
    G01 Z-47.0;
    G01 Z0.0;
    G00 G28X0.Y0.Z0.;
    M05M09;
    T06;
    M06;
    G00 G90 G54 X0. Y0. Z0.;
    G43 H13;
    S1000 M03 M08 F60;
    G90 Z0.0;
    G00 X8.6 Y72.65;
    G01 Z-16.0;
    G01 Z0.0;
    G00 X36.4 Y72.65;
    G01 Z-16.0;
    G01 Z0.0;
    G00 X8.6 Y7.35;
    G01 Z-16.0;
    G01 Z0.0;
    G00 X36.4 Y7.35;
    G01 Z-16.0;
    G01 Z0.0;
    G00 X0.0 Y0.0;
    M05 M09;
    M02;
```

変形センサの裏面エンドミル加

工。

```
O1807
G90 G00 G54 X0.0 Y0.0 Z0.0;
G43 H12;
S1500 M03 M08 F60;
G90 Z0.0;
G00 Y60.0;
G01 Z-12.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
X45.0;
G01 Z0.0;
G00 X0.0 Y60.0;
G01 Z-15.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
X45.0;
G01 Z0.0;
G00 X0.0 Y60.0;
G01 Z-17.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
```

```
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
X45.0;
G01 Z0.0;
G00 X0.0 Y60.0;
G01 Z-20.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
```

```
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
X45.0;
G01 Z0.0;
G00 X0.0 Y60.0;
G01 Z-22.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
```

```
X45.0;
G01 Z0.0;
G00 X0.0 Y60.0;
G01 Z-25.0;
X45.0;
Y55.0;
X0.0;
Y50.0;
X45.0;
Y45.0;
X0.0;
Y40.0;
X45.0;
Y35.0;
X0.0;
Y30.0;
X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y15.0;
X0.0;
Y10.0;
X45.0;
Y5.0;
X0.0;
Y0.0;
X45.0;
G01 Z0.0;
G00 X0.0 Y0.0;
M05 M09;
M02;
```

変形センサ部品の穴あけ加工。

```
O1809
G90 G00 G54 X0.0 Y0.0 Z0.0;
G43 H12;
S1000 M03 M08 F60;
G90 Z0.0;
G00 X8.6 Y22.65;
G01 Z-33.0;
G00 Z0.0;
G00 X36.4 Y22.65;
G01 Z-33.0;
G00 Z0.0;
G00 X8.6 Y7.35;
G01 Z-47.0;
G00 Z0.0;
G00 X36.4 Y7.35;
G01 Z-47.0;
G00 Z0.0;
G00 G28 X0.0 Y0.0 Z0.0;
M05 M09;
T06;
M06;
G00 G90 G54 X0. Y0. Z0.;
G43 H13;
S1000 M03 M08 F60;
G90 Z0.0;
G00 X8.6 Y22.65;
G01 Z-16.0;
G01 Z0.0;
G00 X36.4 Y22.65;
G01 Z-16.0;
G01 Z0.0;
G00 X8.6 Y7.35;
G01 Z-16.0;
G01 Z0.0;
G00 X36.4 Y7.35;
G01 Z-16.0;
G01 Z0.0;
```

```
G00 X0.0 Y0.0;
M05 M09;
M02;
```

変形センサ部品の裏加工。

```
O1810;
G90 G00 G54 X0.0 Y0.0 Z0.0;
G43 H13;
S1500 M03 M08 F60;
G90 Z0.0;
G00 Y30.0;
G01 Z-12.0;
G01 X45.0;
Y25.0;
X0.0;
Y20.0;
X45.0;
Y17.0;
X0.0;
G01 Z0.0;
G00 X0.0 Y0.0;
M05 M09;
M02;
```

変形センサ下部、インバー設置穴あけ加工。

```
O1811
G90 G00 G54 X0. Y0. Z0.;
G43 H12;
S1000 M03 M08 F60;
G90 Z0.;
G00 X22.5 Y25.0;
G01 Z-33.0;
G01 Z0.0;
G00 X0.0 Y0.0;
M05 M09;
M02;
```

２．ワイヤーカット放電加工機のプログラム。

コの字の加工。

```
%
F0.5
M80
M82
M84
G41
H1=140
G90
G92X1.Y0
G01X0Y0
G01X-36.5Y0
G01X-36.5Y-8.5
G01X1.Y-8.5
G40
M02
%
```

端面加工直線。

```
%
H1=140
G90
G92X0Y0
G42
G01X100.Y0.
%
```

平行四辺形加工。

```
%
F0.5
M80
```

M82
M84
G90
G92X0Y0
G41
G01X0.Y5.5
G01X-5.Y5.5
G01X-11.Y-9.5
G01X15.5Y-9.5
G01X21.5Y5.5
G01X-1.Y5.5
G40
M02
%

変な台形加工。
%
F0.5
M80
M82
M84
G90
G92X0Y0
G41
G01X0Y9.5
G01X-13.5Y9.5
G01X-13.5Y5.1
G01X-20.Y5.1
G01X-20.Y-5.5
G01X8.5Y-5.5
G01X8.5Y9.5
G01X-1.Y9.5
G40
M02
%

留め金加工。
%
F0.2
M80
M82
M84
G90
G92X0.Y1.
G41
G01X0.Y-15.
G01X15.Y-15.
G40
M02
%

// **改良変形センサ** DLG.CPP : **インプリメンテーション ファイル**
//

#INCLUDE "STDAFX.H"
#INCLUDE "**改良変形センサ**.H"
#INCLUDE "**改良変形センサ** DLG.H"
#INCLUDE"FBIAD.H"

```
#IFDEF _DEBUG
#DEFINE NEW DEBUG_NEW
#UNDEF THIS_FILE
STATIC CHAR THIS_FILE[] = __FILE__;
HANDLE HDEVICEHANDLE;
WORD WSMPDATA[30000][5];
INT SAMPLINGNUM;
INT I;
ULONG ULSMPLNUM;
ADSMPLREQ SMPLCONFIG;
INT
CH1,CH2,CH3,CH4,CH5,X1,X2,YNEW1,YNEW2,YNEW3,YNEW4,YNEW5,
YOLD1,YOLD2,YOLD3,YOLD4,YOLD5,XA,SAVEDATA;
INT     GRAFU;
CSTRING A,B,C;
CHAR DUMMY[256];
FILE *FP;
INT SAMPLINGPERIOD=0;
INT SAMPLINGFREQ=0;
#ENDIF

/////////////////////////////////////////////////////////////////////////////
// アプリケーションのバージョン情報で使われている CABOUTDLG ダイアログ

CLASS CABOUTDLG : PUBLIC CDIALOG
{
PUBLIC:
        CABOUTDLG();

// ダイアログ データ
        //{{AFX_DATA(CABOUTDLG)
        ENUM { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // CLASSWIZARD は仮想関数のオーバーライドを生成します
```

```
        //{{AFX_VIRTUAL(CABOUTDLG)
        PROTECTED:
        VIRTUAL VOID DODATAEXCHANGE(CDATAEXCHANGE* PDX);
// DDX/DDV のサポート
        //}}AFX_VIRTUAL

// インプリメンテーション
PROTECTED:
        //{{AFX_MSG(CABOUTDLG)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CABOUTDLG::CABOUTDLG() : CDIALOG(CABOUTDLG::IDD)
{
        //{{AFX_DATA_INIT(CABOUTDLG)
        //}}AFX_DATA_INIT
}

VOID CABOUTDLG::DODATAEXCHANGE(CDATAEXCHANGE* PDX)
{
        CDIALOG::DODATAEXCHANGE(PDX);
        //{{AFX_DATA_MAP(CABOUTDLG)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CABOUTDLG, CDIALOG)
        //{{AFX_MSG_MAP(CABOUTDLG)
                // メッセージ ハンドラがありません。
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMYDLG ダイアログ

CMYDLG::CMYDLG(CWND* PPARENT /*=NULL*/)
```

```
    : CDIALOG(CMYDLG::IDD, PPARENT)
{
    //{{AFX_DATA_INIT(CMYDLG)
    M_MESSAGE = _T("");
    M_E1CH = _T("");
    M_E2CH = _T("");
    M_E3CH = _T("");
    M_E4CH = _T("");
    M_E5CH = _T("");
    M_CH1 = FALSE;
    M_CH10 = FALSE;
    M_CH11 = FALSE;
    M_CH12 = FALSE;
    M_CH13 = FALSE;
    M_CH14 = FALSE;
    M_CH2 = FALSE;
    M_CH3 = FALSE;
    M_CH4 = FALSE;
    M_CH5 = FALSE;
    M_CH6 = FALSE;
    M_CH7 = FALSE;
    M_CH8 = FALSE;
    M_CH9 = FALSE;
    //}}AFX_DATA_INIT
    // メモ: LOADICON は WIN32 の DESTROYICON のサブシーケンスを要求しません。
    M_HICON = AFXGETAPP()->LOADICON(IDR_MAINFRAME);
}

VOID CMYDLG::DODATAEXCHANGE(CDATAEXCHANGE* PDX)
{
    CDIALOG::DODATAEXCHANGE(PDX);
    //{{AFX_DATA_MAP(CMYDLG)
    DDX_CONTROL(PDX, IDC_PICT, M_PICT);
    DDX_CONTROL(PDX, IDC_PICT5, M_5CH);
    DDX_CONTROL(PDX, IDC_PICT4, M_4CH);
```

```
        DDX_CONTROL(PDX, IDC_PICT3, M_3CH);
        DDX_CONTROL(PDX, IDC_PICT2, M_2CH);
        DDX_CONTROL(PDX, IDC_PICT1, M_1CH);
        DDX_CONTROL(PDX, IDC_SAMPLING, M_SAMPLING);
        DDX_CONTROL(PDX, IDC_SMPNUM, M_SMPNUM);
        DDX_CONTROL(PDX, IDC_SMPFREQ, M_SMPFREQ);
        DDX_CONTROL(PDX, IDC_STOP, M_STOP);
        DDX_CONTROL(PDX, IDC_START, M_START);
        DDX_CONTROL(PDX, IDC_SET, M_SET);
        DDX_CONTROL(PDX, IDC_ADCLOSE, M_ADCLOSE);
        DDX_CONTROL(PDX, IDC_ADOPEN, M_ADOPEN);
        DDX_TEXT(PDX, IDC_EDIT_MESSAGE, M_MESSAGE);
        DDX_TEXT(PDX, IDC_CH1, M_E1CH);
        DDX_TEXT(PDX, IDC_CH2, M_E2CH);
        DDX_TEXT(PDX, IDC_CH3, M_E3CH);
        DDX_TEXT(PDX, IDC_CH4, M_E4CH);
        DDX_TEXT(PDX, IDC_CH5, M_E5CH);
        DDX_CHECK(PDX, IDC_CHECK1, M_CH1);
        DDX_CHECK(PDX, IDC_CHECK10, M_CH10);
        DDX_CHECK(PDX, IDC_CHECK11, M_CH11);
        DDX_CHECK(PDX, IDC_CHECK12, M_CH12);
        DDX_CHECK(PDX, IDC_CHECK13, M_CH13);
        DDX_CHECK(PDX, IDC_CHECK14, M_CH14);
        DDX_CHECK(PDX, IDC_CHECK2, M_CH2);
        DDX_CHECK(PDX, IDC_CHECK3, M_CH3);
        DDX_CHECK(PDX, IDC_CHECK4, M_CH4);
        DDX_CHECK(PDX, IDC_CHECK5, M_CH5);
        DDX_CHECK(PDX, IDC_CHECK6, M_CH6);
        DDX_CHECK(PDX, IDC_CHECK7, M_CH7);
        DDX_CHECK(PDX, IDC_CHECK8, M_CH8);
        DDX_CHECK(PDX, IDC_CHECK9, M_CH9);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMYDLG, CDIALOG)
        //{{AFX_MSG_MAP(CMYDLG)
```

```
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_ADOPEN, ONADOPEN)
	ON_BN_CLICKED(IDC_ADCLOSE, ONADCLOSE)
	ON_BN_CLICKED(IDC_START, ONSTART)
	ON_BN_CLICKED(IDC_SET, ONSET)
	ON_BN_CLICKED(IDC_RADIO1, ONRADIO1)
	ON_BN_CLICKED(IDC_RADIO2, ONRADIO2)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_STOP, ONSTOP)
	ON_BN_CLICKED(IDC_SERVERCON, ONSERVERCON)
	ON_BN_CLICKED(IDC_CHECK1, ONCHECK1)
	ON_BN_CLICKED(IDC_CHECK2, ONCHECK2)
	ON_BN_CLICKED(IDC_CHECK3, ONCHECK3)
	ON_BN_CLICKED(IDC_CHECK4, ONCHECK4)
	ON_BN_CLICKED(IDC_CHECK5, ONCHECK5)
	ON_BN_CLICKED(IDC_CHECK6, ONCHECK6)
	ON_BN_CLICKED(IDC_CHECK7, ONCHECK7)
	ON_BN_CLICKED(IDC_CHECK8, ONCHECK8)
	ON_BN_CLICKED(IDC_CHECK9, ONCHECK9)
	ON_BN_CLICKED(IDC_CHECK10, ONCHECK10)
	ON_BN_CLICKED(IDC_CHECK11, ONCHECK11)
	ON_BN_CLICKED(IDC_CHECK12, ONCHECK12)
	ON_BN_CLICKED(IDC_CHECK13, ONCHECK13)
	ON_BN_CLICKED(IDC_CHECK14, ONCHECK14)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMYDLG メッセージ ハンドラ

BOOL CMYDLG::ONINITDIALOG()
{
	CDIALOG::ONINITDIALOG();
```

```
        // "バージョン情報..." メニュー項目をシステム メニューへ追加します。

        // IDM_ABOUTBOX はコマンド メニューの範囲でなければなりません。
        ASSERT((IDM_ABOUTBOX & 0XFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0XF000);

        CMENU* PSYSMENU = GETSYSTEMMENU(FALSE);
        IF (PSYSMENU != NULL)
        {
                CSTRING STRABOUTMENU;
                STRABOUTMENU.LOADSTRING(IDS_ABOUTBOX);
                IF (!STRABOUTMENU.ISEMPTY())
                {
                        PSYSMENU->APPENDMENU(MF_SEPARATOR);
                        PSYSMENU->APPENDMENU(MF_STRING,
IDM_ABOUTBOX, STRABOUTMENU);
                }
        }

        // このダイアログ用のアイコンを設定します。フレームワークはアプリケーションのメイン
        // ウィンドウがダイアログでない時は自動的に設定しません。
        SETICON(M_HICON, TRUE);                 // 大きいアイコンを設定
        SETICON(M_HICON, FALSE);                // 小さいアイコンを設定

        // TODO: 特別な初期化を行う時はこの場所に追加してください。
        M_ADCLOSE.ENABLEWINDOW(FALSE);
    M_ADOPEN.ENABLEWINDOW(TRUE);
        M_START.ENABLEWINDOW(FALSE);
    M_STOP.ENABLEWINDOW(FALSE);


        RETURN TRUE;  // TRUE を返すとコントロールに設定したフォー
```

カスは失われません。
}

```
VOID CMYDLG::ONSYSCOMMAND(UINT NID, LPARAM LPARAM)
{
        IF ((NID & 0XFFF0) == IDM_ABOUTBOX)
        {
                CABOUTDLG DLGABOUT;
                DLGABOUT.DOMODAL();
        }
        ELSE
        {
                CDIALOG::ONSYSCOMMAND(NID, LPARAM);
        }
}
```

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションは DOCUMENT/VIEW
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

```
VOID CMYDLG::ONPAINT()
{
        IF (ISICONIC())
        {
                CPAINTDC DC(THIS); // 描画用のデバイス コンテキスト

                SENDMESSAGE(WM_ICONERASEBKGND,     (WPARAM)
DC.GETSAFEHDC(), 0);

                // クライアントの矩形領域内の中央
                INT CXICON = GETSYSTEMMETRICS(SM_CXICON);
                INT CYICON = GETSYSTEMMETRICS(SM_CYICON);
                CRECT RECT;
```

```
            GETCLIENTRECT(&RECT);
            INT X = (RECT.WIDTH() - CXICON + 1) / 2;
            INT Y = (RECT.HEIGHT() - CYICON + 1) / 2;

            // アイコンを描画します。
            DC.DRAWICON(X, Y, M_HICON);
        }
        ELSE
        {
            CDIALOG::ONPAINT();
        }
}

// システムは、ユーザーが最小化ウィンドウをドラッグしている間、
// カーソルを表示するためにここを呼び出します。
HCURSOR CMYDLG::ONQUERYDRAGICON()
{
        RETURN (HCURSOR) M_HICON;
}

VOID CMYDLG::ONADOPEN()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
        M_ADCLOSE.ENABLEWINDOW(TRUE);
        M_ADOPEN.ENABLEWINDOW(FALSE);
        M_MESSAGE="AD ボードオープンしました";
         HDEVICEHANDLE = ADOPEN("FBIAD1");
        IF (HDEVICEHANDLE == INVALID_HANDLE_VALUE){
            MESSAGEBOX("DEVICE FBIAD1 は使用できません");
            EXIT(0);
        }
        ADGETSAMPLINGCONFIG(HDEVICEHANDLE,&SMPLCONFIG)
;
   SMPLCONFIG.ULSMPLNUM = SAMPLINGNUM;
   SMPLCONFIG.ULCHCOUNT =5;
```

```
        SMPLCONFIG.ULSAMPLINGMODE = AD_IO_SAMPLING;
        SMPLCONFIG.ULSINGLEDIFF = AD_INPUT_SINGLE;
        SMPLCONFIG.FSMPLFREQ= SAMPLINGFREQ;
        SMPLCONFIG.SMPLCHREQ[0].ULCHNO = 1;
    SMPLCONFIG.SMPLCHREQ[0].ULRANGE = AD_5V;
    SMPLCONFIG.SMPLCHREQ[1].ULCHNO = 2;
    SMPLCONFIG.SMPLCHREQ[1].ULRANGE = AD_5V;
    SMPLCONFIG.SMPLCHREQ[2].ULCHNO = 3;
    SMPLCONFIG.SMPLCHREQ[2].ULRANGE = AD_5V;
        SMPLCONFIG.SMPLCHREQ[3].ULCHNO = 4;
    SMPLCONFIG.SMPLCHREQ[3].ULRANGE = AD_5V;
        SMPLCONFIG.SMPLCHREQ[4].ULCHNO = 5;
    SMPLCONFIG.SMPLCHREQ[4].ULRANGE = AD_5V;

        ADSETSAMPLINGCONFIG(HDEVICEHANDLE,&SMPLCONFIG);
        UPDATEDATA(FALSE);
}

VOID CMYDLG::ONADCLOSE()
{
        M_MESSAGE="AD ボード閉じました";
        M_ADCLOSE.ENABLEWINDOW(FALSE);
        M_ADOPEN.ENABLEWINDOW(TRUE);
        M_START.ENABLEWINDOW(FALSE);
    M_STOP.ENABLEWINDOW(FALSE);
        M_SET.ENABLEWINDOW(TRUE);
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
         ADCLOSE(HDEVICEHANDLE);
         UPDATEDATA(FALSE);
}

VOID CMYDLG::ONSTART()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
```

```
        M_MESSAGE="サンプリング開始します";
        M_START.ENABLEWINDOW(FALSE);
        M_STOP.ENABLEWINDOW(TRUE);
        M_SMPFREQ.GETWINDOWTEXT(B);
   SAMPLINGFREQ=ATOI(B);
        M_SMPNUM.GETWINDOWTEXT(C);
 SAMPLINGNUM=ATOI(C);
        IF(GRAFU==1){
SETTIMER(1,45,NULL);

//1CH の色
 CBRUSH NEWBRUSH;
 CDC* PDC=M_1CH.GETDC();
 NEWBRUSH.CREATESOLIDBRUSH(RGB(255,0,0));
  PDC->SELECTOBJECT(&NEWBRUSH);
  PDC->RECTANGLE(0,0,10,10);
//2CH の色
        CDC* PDC2=M_2CH.GETDC();
        CBRUSH NEWBRUSH2;
 NEWBRUSH2.CREATESOLIDBRUSH(RGB(0,0,255));
  PDC2->SELECTOBJECT(&NEWBRUSH2);
  PDC2->RECTANGLE(0,0,10,10);
        //3CH の色
        CDC* PDC3=M_3CH.GETDC();
        CBRUSH NEWBRUSH3;
 NEWBRUSH3.CREATESOLIDBRUSH(RGB(0,255,0));
  PDC3->SELECTOBJECT(&NEWBRUSH3);
  PDC3->RECTANGLE(0,0,10,10);
        //4CH の色
        CDC* PDC4=M_4CH.GETDC();
        CBRUSH NEWBRUSH4;
 NEWBRUSH4.CREATESOLIDBRUSH(RGB(0,255,255));
  PDC4->SELECTOBJECT(&NEWBRUSH4);
   PDC4->RECTANGLE(0,0,10,10);
        //5CH の色
        CDC* PDC5=M_5CH.GETDC();
```

```
        CBRUSH NEWBRUSH5;
    NEWBRUSH5.CREATESOLIDBRUSH(RGB(255,0,255));
     PDC5->SELECTOBJECT(&NEWBRUSH5);
     PDC5->RECTANGLE(0,0,10,10);
        //座標軸
        CDC*PDC6=M_PICT.GETDC();
        PDC6->MOVETO(0,200);
      PDC6->LINETO(410,200);
     PDC6->MOVETO(10,0);
     PDC6->LINETO(10,310);
    }
    IF(SAVEDATA==1){
        ADSTARTSAMPLING(HDEVICEHANDLE, FLAG_SYNC);
        ULSMPLNUM=SAMPLINGNUM;
        ADGETSAMPLINGDATA(HDEVICEHANDLE, &WSMPDATA[0][0],
&ULSMPLNUM);
        FP = FOPEN("SAMPLEDATA.TXT","W+");
        IF(FP==NULL){
                FPRINTF(STDERR,"¥N ¥T CANNOT OPEN
SAMPLEDATA.TXT¥N¥T");
        }
        FOR (I=0; I<SAMPLINGNUM; I++)

        FPRINTF(FP,"%.2F %F %F %F
     %F                                                %F
¥N",I*1/SMPLCONFIG.FSMPLFREQ,(10.0*WSMPDATA[I][0]/65536-5.0)*0
.9,
        (10.0*WSMPDATA[I][1]/65536-5.0)*0.9,(10.0*WSMPDATA[I][2]/655
36-5.0)*0.9,(10.0*WSMPDATA[I][3]/65536-5.0)*0.9,(10.0*WSMPDATA[I][4]/
65536-5.0)*0.9);
                FCLOSE(FP);
    M_MESSAGE="サンプリング完了しました";
        ADSTOPSAMPLING(HDEVICEHANDLE);
}
        UPDATEDATA(FALSE);
```

```
}

VOID CMYDLG::ONSET()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
  M_START.ENABLEWINDOW(TRUE);
  M_SAMPLING.GETWINDOWTEXT(A);
  SAMPLINGPERIOD=ATOI(A);
  M_SMPFREQ.GETWINDOWTEXT(B);
  SAMPLINGFREQ=ATOI(B);
  SAMPLINGNUM=SAMPLINGPERIOD*60*SAMPLINGFREQ;
  CHAR DSP[100];
  SPRINTF(DSP,"%D",SAMPLINGNUM);
  M_SMPNUM.SETWINDOWTEXT(DSP);
}

VOID CMYDLG::ONRADIO1()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
        M_SAMPLING.ENABLEWINDOW(TRUE);
        M_SMPNUM.ENABLEWINDOW(TRUE);
        SAVEDATA=1;
        GRAFU=0;
}

VOID CMYDLG::ONRADIO2()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
   M_SAMPLING.ENABLEWINDOW(FALSE);
        M_SMPNUM.ENABLEWINDOW(FALSE);
        SAVEDATA=0;
        GRAFU=1;
```

```
}

VOID CMYDLG::ONTIMER(UINT NIDEVENT)
{
        // TODO: この位置にメッセージ ハンドラ用のコードを追加するかまたはデフォルトの処理を呼び出してください

        ADSTARTSAMPLING(HDEVICEHANDLE, FLAG_ASYNC);
        ULSMPLNUM=1000;
        ADGETSAMPLINGDATA(HDEVICEHANDLE, &WSMPDATA[0][0], &ULSMPLNUM);
     SPRINTF(DUMMY,  "%F",(10.0*WSMPDATA[0][0]/65536-5.0)/2.0);
     M_E1CH=DUMMY;
          SPRINTF(DUMMY, "%F",(10.0*WSMPDATA[1][0]/65536-5.0)/2.0);
     M_E2CH=DUMMY;
          SPRINTF(DUMMY, "%F",(10.0*WSMPDATA[2][0]/65536-5.0)/2.0);
     M_E3CH=DUMMY;
          SPRINTF(DUMMY, "%F",(10.0*WSMPDATA[3][0]/65536-5.0)/2.0);
     M_E4CH=DUMMY;
          SPRINTF(DUMMY, "%F",(10.0*WSMPDATA[4][0]/65536-5.0)/2.0);
     M_E5CH=DUMMY;
        //描画
        CH1=ATOI(M_E1CH)*100.0;
   CH2=ATOI(M_E2CH)*100.0;
        CH3=ATOI(M_E3CH)*100.0;
        CH4=ATOI(M_E4CH)*100.0;
        CH5=ATOI(M_E5CH)*100.0;
        X1=X1+1;
        X2=X1;
        YNEW1=((CH1/4)-200)*-1;
   YNEW2=((CH2/4)-200)*-1;
        YNEW3=((CH3/4)-200)*-1;
        YNEW4=((CH4/4)-200)*-1;
        YNEW5=((CH5/4)-200)*-1;
        CDC* PDC=M_PICT.GETDC();
          CPEN
```

```
BLUEPEN,REDPEN,GREENPEN,BRACKPEN,CIANPEN,MAZENDAPEN
;
        REDPEN.CREATEPEN(PS_SOLID,1,RGB(255,0,0));
          PDC->SELECTOBJECT(&REDPEN);
            PDC->MOVETO(XA,YOLD1);
            PDC->LINETO(X2,YNEW1);
           YOLD1=YNEW1;
        BLUEPEN.CREATEPEN(PS_SOLID,1,RGB(0,0,255));
          PDC->SELECTOBJECT(&BLUEPEN);
            PDC->MOVETO(XA,YOLD2);
            PDC->LINETO(X2,YNEW2);
           YOLD2=YNEW2;
        GREENPEN.CREATEPEN(PS_SOLID,1,RGB(0,255,0));
          PDC->SELECTOBJECT(&GREENPEN);
            PDC->MOVETO(XA,YOLD3);
            PDC->LINETO(X2,YNEW3);
           YOLD3=YNEW3;
        CIANPEN.CREATEPEN(PS_SOLID,1,RGB(0,255,255));
          PDC->SELECTOBJECT(&CIANPEN);
            PDC->MOVETO(XA,YOLD4);
            PDC->LINETO(X2,YNEW4);
           YOLD4=YNEW4;
        MAZENDAPEN.CREATEPEN(PS_SOLID,1,RGB(255,0,255));
          PDC->SELECTOBJECT(&MAZENDAPEN);
            PDC->MOVETO(XA,YOLD5);
            PDC->LINETO(X2,YNEW5);
           YOLD5=YNEW5;
                XA=X2;
                IF(X1>410){
           CDC*PDC=M_PICT.GETDC();
           CRECT MYRECT;
           M_PICT.GETCLIENTRECT(MYRECT);
            PDC->FILLSOLIDRECT(MYRECT,RGB(255,255,255));
                 PDC->SELECTOBJECT(&BRACKPEN);
                     PDC->MOVETO(0,200);
         PDC->LINETO(410,200);
```

```
            PDC->MOVETO(10,0);
            PDC->LINETO(10,310);
          X1=9;
          XA=10;
         YOLD1=200;YOLD2=200; YOLD3=200;YOLD4=200;YOLD5=200;
                  }
   UPDATEDATA(FALSE);
       CDIALOG::ONTIMER(NIDEVENT);
}

VOID CMYDLG::ONSTOP()
{
       // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
       M_MESSAGE="測定を停止します";
        KILLTIMER(1);
        M_STOP.ENABLEWINDOW(FALSE);
       M_START.ENABLEWINDOW(TRUE);
       UPDATEDATA(FALSE);
}

VOID CMYDLG::ONSERVERCON()
{
/*     // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
                SOCK = NEW CCOMM;
       IF (SOCK->INITIALIZE("210.163.149.97") == FALSE){
                SOCK->~CCOMM();
                AFXMESSAGEBOX("SOCKET CONNECT FAILED");
       }*/
}

VOID CMYDLG::ONCHECK1()
{
/*     // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
```

```
        IF(M_CH1 == TRUE){
                M_CH1 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="前面左ヒーター停止";
        }
        ELSE IF (M_CH1 == FALSE){
                M_CH1 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="前面左ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "G"); */
}

VOID CMYDLG::ONCHECK2()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH2 == TRUE){
                M_CH2 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="前面右ヒーター停止";
        }
        ELSE IF (M_CH2 == FALSE){
                M_CH2 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="前面右ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
    UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "H"); */
}
```

```
VOID CMYDLG::ONCHECK3()
{
        // TODO:  この位置にコントロール通知ハンドラ用のコードを追加してください
/*              IF(M_CH3 == TRUE){
                M_CH3 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="背面左ヒーター停止";
        }
        ELSE IF (M_CH3 == FALSE){
                M_CH3 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="背面左ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "I"); */
}

VOID CMYDLG::ONCHECK4()
{
        // TODO:  この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH4 == TRUE){
                M_CH4 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="背面右ヒーター停止";
        }
        ELSE IF (M_CH4 == FALSE){
                M_CH4 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="背面右ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
```

```
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "J"); */
}

VOID CMYDLG::ONCHECK5()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH5 == TRUE){
                M_CH5 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="側面前ヒーター停止";
        }
        ELSE IF (M_CH5 == FALSE){
                M_CH5 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="側面前ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "K"); */
}

VOID CMYDLG::ONCHECK6()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH6 == TRUE){
                M_CH6 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="側面後ヒーター停止";
        }
        ELSE IF (M_CH6 == FALSE){
                M_CH6 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
```

```
            M_MESSAGE ="側面後ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "L"); */
}

VOID CMYDLG::ONCHECK7()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH7 == TRUE){
                M_CH7 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="ATC 側前ヒーター停止";
        }
        ELSE IF (M_CH7 == FALSE){
                M_CH7 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="ATC 側前ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
     UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "M"); */
}

VOID CMYDLG::ONCHECK8()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH8 == TRUE){
                M_CH8 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="ATC 側後ヒーター停止";
```

```
        }
        ELSE IF (M_CH8 == FALSE){
                M_CH8 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="ATC 側後ヒーター稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "N"); */
}

VOID CMYDLG::ONCHECK9()
{
        // TODO:  この位置にコントロール通知ハンドラ用のコードを追加してください
/*              IF(M_CH9 == TRUE){
                CH9 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="前面冷却ファン停止";
        }
        ELSE IF (CH9 == FALSE){
                CH9 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="前面冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "A"); */
}

VOID CMYDLG::ONCHECK10()
{
        // TODO:  この位置にコントロール通知ハンドラ用のコードを追加してください
```

```
/*      IF(M_CH10 == TRUE){
            M_CH10 = FALSE;
         SDATA.FV[0] = DOUBLE(0);
         M_MESSAGE ="背面冷却ファン停止";
        }
        ELSE IF (M_CH10 == FALSE){
            M_CH10 = TRUE;
         SDATA.FV[0] = DOUBLE(1);
         M_MESSAGE ="背面冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "C"); */
}

VOID CMYDLG::ONCHECK11()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*              IF(M_CH11 == TRUE){
                M_CH11 = FALSE;
         SDATA.FV[0] = DOUBLE(0);
         M_MESSAGE ="側面前冷却ファン停止";
        }
        ELSE IF (M_CH11 == FALSE){
            M_CH11 = TRUE;
         SDATA.FV[0] = DOUBLE(1);
         M_MESSAGE ="側面前冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "B"); */
}
```

```
VOID CMYDLG::ONCHECK12()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
        /*      IF(M_CH12 == TRUE){
                M_CH12 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="側面後冷却ファン停止";
        }
        ELSE IF (M_CH12 == FALSE){
                M_CH12 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="側面後冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "E"); */
}

VOID CMYDLG::ONCHECK13()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*      IF(M_CH13 == TRUE){
                M_CH13 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="ATC 側前冷却ファン停止";
        }
        ELSE IF (M_CH13 == FALSE){
                M_CH13 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="ATC 側前冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
```

```
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "F"); */
}

VOID CMYDLG::ONCHECK14()
{
        // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
/*              IF(M_CH14 == TRUE){
                M_CH14 = FALSE;
          SDATA.FV[0] = DOUBLE(0);
          M_MESSAGE ="ATC 側後冷却ファン停止";
        }
        ELSE IF (M_CH14 == FALSE){
                M_CH14 = TRUE;
          SDATA.FV[0] = DOUBLE(1);
          M_MESSAGE ="ATC 側後冷却ファン稼動";

        }
    SOCK->SOCKWRITE(SDATA);
   UPDATEDATA(FALSE);
        SPRINTF(SDATA.COMMAND, "D"); */
}
```