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－P91　　**完**

平成 15 年 2 月 28 日提出

指導教官　　**長尾高明教授**

1030118　　**高谷　周一郎**

目次

第１章

緒言

1.1　研究の背景

人類はこの地球上に誕生してから暮らしを豊かにするために様々な物を開発してきた。そして、物作りのシステムが大幅に変わる出来事がイギリスで起こった。それは、産業革命だ。それまで人類は手作業で製品を作ってきたが、機械の誕生で作業の効率があがり大量に生産ができるようになった。しかし、マザーマシンの誕生で高水準の生産性や信頼性が要求されるようになった。

一昔前までは、高水準の製品を生産するには熟練者でなくては作れなかったが、汎用工作機械から NC 化された工作機械が開発されたことによりマニュアルやプログラムを学ぶだけである程度の物は精度良く製品を生産ができるようになった。しかし、高い精度が要求せれる加工については、熟練者の経験がまだまだ必要だということはいまだ変わらない。

今現在、日本は戦後始まって以来の大不況で今まで日本を支えてきた、町工場が次々と潰れて来ている。その原因として人件費の安い中国に目をつけた国内外企業が工場を作り加工を受注し、その結果近年中国が力をつけある程度の加工ができるようになり日本の仕事を取るようになったことが要因だと考えられる。人件費に打ち勝つことのできない日本が、中国の加工産業に打ち勝つためには、加工精度を上げて行くしか打開策が見出せない。では、加工精度を向上させるにはどうすれば良いのだろうか。

加工精度の要求が高まる中で、工作機械が熱や加工反力によって変形し高精度加工ができないというのが実情だ。

実際、変形による精度の低下をおさえるためにファンを使い機械を冷やしながら加工を行ったり、技術者が変形を考慮して加工したり、数時間工作機械を空運転させて変形を安定させてから加工を行うことしか精度をあげる手段がない。しかし、すべての変形を考慮することはできない。

そこで私は、変形センサを設計・開発をし、ANSYS・実験装置・マシニングセンタで変形量を調べた。

1.2　一般の工場で行われている精度向上のための作業

工作機械を運転すると、主軸回転速度、早送り繰り返し頻度、切削油剤の使用状況、照明の当て加減等、その運転状況と室内の温度、温度変化の度合いによって機械に変形がおこり加工に誤差が生じる。

短時間のよる加工による誤差は微少だが、長時間かかる加工では、加工位置間の寸法や機械原点とテーブル上の基準との距離に誤差生じ、それは無視することができない量になる。

熱変形は、同じ運動状態でも室内環境の違い、季節などによって誤差が数 10μになることがある。また、発熱置と遠い部分との変形時間の誤差が生まれるという問題も発生する。

ウォーミングアップでの精度向上

大阪機工(株)VM 4—Ⅱは、ソフトスケールの採用により、無負荷運転に対する熱変形補正を行いますが､出勤してから冷え切った機械を 100%フル起動させるとねじの伸びや主軸の変位が急激な変化を生じるため、補正に遅れが生じることがある。

この遅れを防ぐ為に、運転時に予めウォーミングアップすることで変形させておけば加工開始時点からの変形が少なくなりソフトスケールの効果もより有効的に作用できる。

主軸回転のウォーミングアップ

主軸軸受のなじみを良くするために、30 分程度慣らし運転を行う。回転速度と時間の関係は下表に示す。

表 1.1 標準仕様機(6000min^{-1})

主軸回転速度(min^{-1})	時間(min^{-1})
500	5
2000	10
4000	10
6000	5

第２章

加工の知能化の基礎概念

２.１　　加工知能化の基礎概念

加工というのは、提示された寸法で正確かつ効率よく作り出すことを目標にする。しかし、正確にそれも効率良く加工するというのはとても困難でそれは、これから改善されるべき問題である。

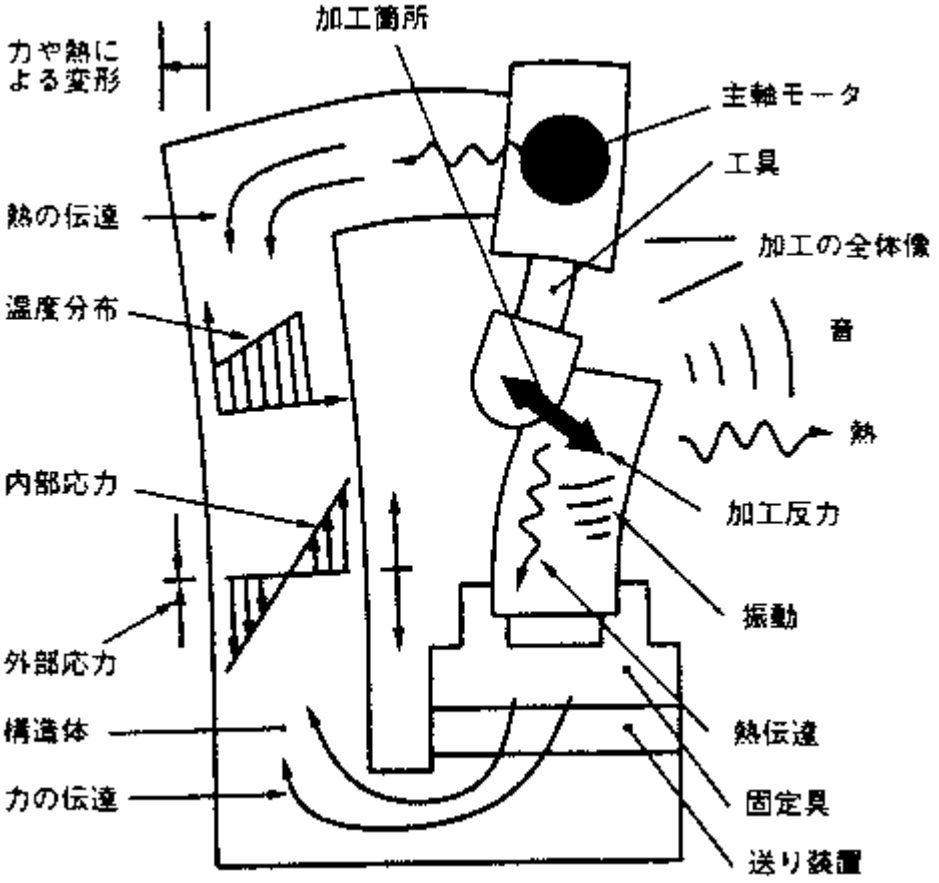


図２.１　加工中に生じる物理現象

その物理現象の原因として考えられるのは、図２.１に示すような加工中に生じる様々な物理現象である。マシニングセンタはもちろんのこと工作機械はこれらの現象によって、力と熱で変形し、また、歪み、音、温度などで高精度加工ができないということがわかっている。そこで、私たちは高精度加工低下の原因となる情報をセンサによって取り組み、機械の設定数値とセンサで得た数値とのずれを把握し、ファン、アクチュエータで誤差を修正すれば正確な製品を作り出すことが可能になると考えている。また、マシニングセンタとコンピ

ュータを繋げることにより、センサで得たデータ、ファン、アクチュエータの動作内容、加工状態をデータ・ベース化し、それに基づいて内部モデルを修正することで、機械自体が学習し「加工の知能化」するようになる。その基本構成を図２.２に示す。

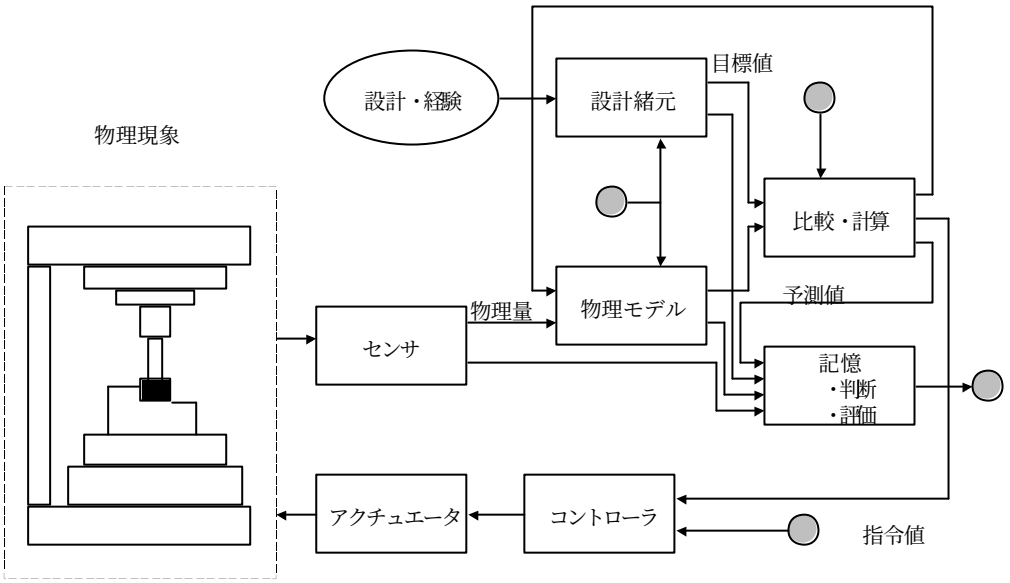


図２.２加工知能化の基本構成

2.2 マシニングセンタにおける熱変形能動補償の意義

今までに述べたように様々な物理現象が加工中に生じていますが、マシニングセンタの加工精度を一番低下させているのは加工をするときの加工反力による変形、熱による変形である。

加工反力による変形を補うために、工作機械本体の剛性を高めることで、変形を抑えるという方法がとられてきましたが、材料の剛性にも限界がありますし、コストがかかることから、解決には至っていないのが現状である。

もう一つの原因の熱による変形については、熱を発する部分を冷やしたり、発熱部からの伝熱を防ぐと言う対策が行われてきた。また、室温変化による変位を防止する為に、室温を一定に保つ（室温を一定に保つだけで、２０～３０ミクロンの変形を防ぐことができる。）試みがされてきた。しかし、熱を発する部分は、主軸モータの熱、加工中の工具と工作物の熱、工場の室温である。それら全ての熱を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な多くの部品と複雑な構造で構成される物は、加工反力による変形と熱変形を具体的に解析、モデル化することは非常に困難である。モデル化による熱変形補正は複雑な熱変形を十二分に補うことはできない。

そこで本研究では、「加工の知能化」を行うために変形を能動的に補正するための大切な、マシニングセンタの変形量を変形センサなど用いて測ろうとする研究である。

第３章

マシニングセンタの構成

3.1　システムの概略

研究に用いた高精密加工システムの概略を図 3.1 に示す。

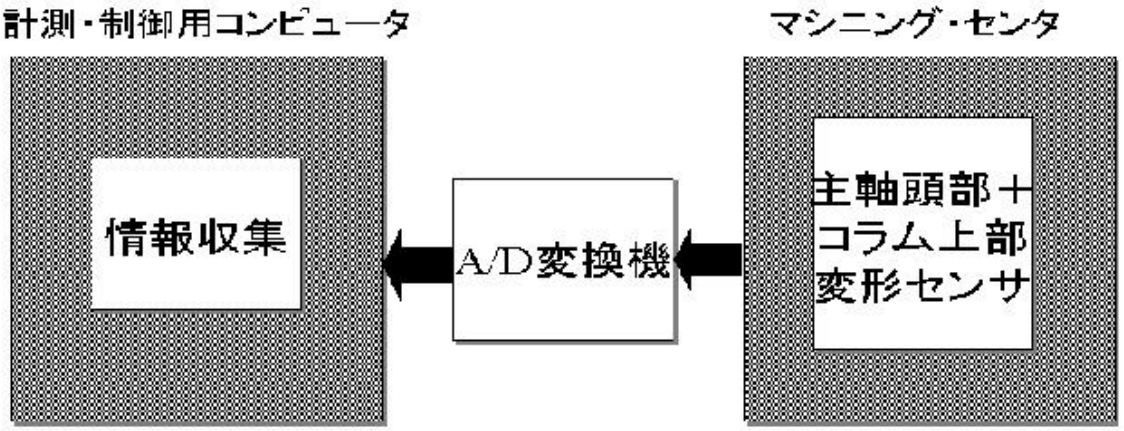


図3．1　システム概略

次節以降で、それぞれの機能について述べる。

3.2　マシニングセンタ

図 3.2、表 3.1 に示すのが、今回の研究に用いた高精密マシニングセンタの外観及びその仕様である。

図 3.2　高精密マシニングセンタ

表 3.1　熱変形能動補償マシニングセンタ(VM 4－Ⅱ)

項目	単位	仕様
作業面の大きさ	mm	800×410
最大移動距離 テーブル左右(X) サドル(前後) (Y) ヘッド(上下)(Z)	mm	 630 410 460
切削送り速度	mm/min	1～10000
早送り速度	m/min	XY：24　Z：16
主軸回転数 指令方式	Min^{-1}	25～6000 S5 桁
回転速度域変換数		2 段
加工物許容質量	kg	500
最大指令値	mm	±99999.999

このマシニングセンタは、大阪機工(株)製の VM 4－Ⅱという機種の小型汎用マシニングセンタである。これは加工の知能化を実現するための特殊な機構を装備したものである。

その機構を以下に示す。

コラム下部に熱アクチュエータ(ヒータ 16 点　ファン 14 点)を装備し、熱変形を能動的に補正ができるようになっている。

マシニングセンタには、合計 8 箇所に変形センサを取り付けることができるが、本研究ではその内の 5 個所に変形センサを取り付けた。そのことによりマシニングセンタの熱変形状態をコンピュータでリアルタイムに知ることができる。その情報を基に熱アクチュエータを用いて変形を補正することが出来る。

マシニングセンタをインターネットに接続することにより、遠隔操作を行うことができる。

フェイルセイフ機能とフェイルセイフテーブルを装備しており、X,Y 軸に過大な荷重が掛かった時、テーブルが逃げマシニングセンタを守ることができる。
外部制御装置からマシニングセンタの機能を使用できる。(ジョイスチックなど。)

3.3 変形検出機構

3.3.1 A/C ストレインアンプ

図に示すのは NEC 三栄（株）A/C ストレインアンプ AH1108/1116 である。本研究では、これを用いて５本の変形センサの歪ゲージ出力を増幅し、各センサの較正データからセンサ位置での実際の歪量を計算する。表 3.2 表 3.3 表 3.4 にその仕様を示す。

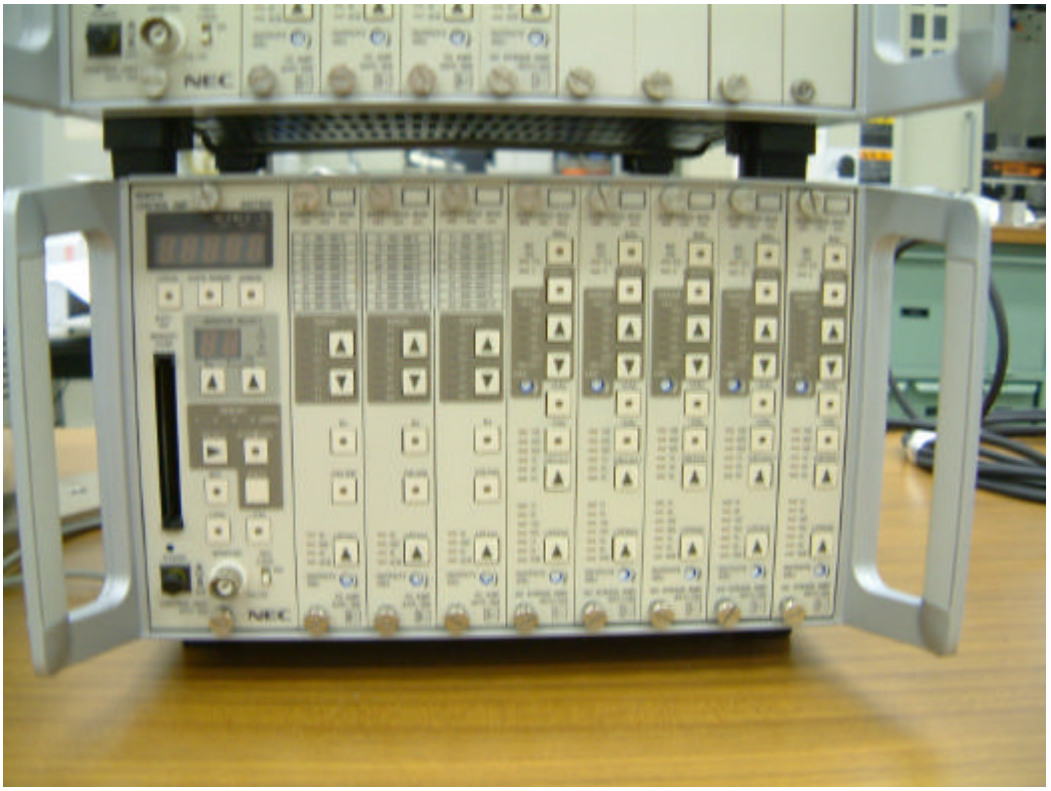


図 3.3 A/C ストレインアンプ

リモートコントロールアンプ AH1108 は、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がある。また計測の容易さを実現するオートレンジ、セルフチェックの搭載などでシステム計測し、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。

表 3.2　AC ストレインアンプユニット

型番	AH11-104/204
適用ゲージ抵抗	120Ω～1kΩ
ブリッジ電源	2V,0.5Vrms 内部切り換え AH11-104：正弦波 25kHz、AH11-204：正弦波 5kHz
平衡調整方式	抵抗分自動バランス、バックアップ約１ヶ月 容量分自動除去
平衡調整範囲	抵抗分±約 2%（±約 10000×10^{-6} ひずみ）
電圧感度	200×10^{-6} ひずみ入力にて 5V 以上（VAR 最大）
測定範囲	200、500、1k、2k、5k×10^{-6} ひずみ/FS、OFF (VAR 最大、BV≒2V)
非直線性	±0.2%/FS 以内
周波数特性	DC～10kHz±10%(AH11-104) DC～2kHz±10%

表 3.3　熱電対アンプユニット AH11-109

使用熱電対	T 形(CC)、E 形(CRC)、J 形(IC)、K 形(CA)
測定温度範囲	T 形　T1：－150～150℃　T2：－200～300℃ E 形　E1：－200～400℃　E2：－200～800℃ J 形　J1　：－200～400℃　J2：－200～800℃ K 形　K1：－200～600℃　K2：－200～1200℃
温度補償回路	誤差±2℃以内
リニアライザ回路	±0.5%FS 以内(0℃以上)　±1.0%/FS 以内(0℃以下)
雑音	7.5℃p-p(K 形－200～1200℃レンジ、フィルタ W/B の時)
周波数特性	DC～200kHz　＋1dB、－3dB

表 3.4　標準仕様(本体ケース　コントロールユニット)

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ルの設定数値の記憶 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、表示部に出力される。
外部インターフェイス	GP-IB、RS-232C 標準装備(同時使用不可)外部インターフェ イスを利用して、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：2Vrms 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
耐振性	29.4m/s²(3G)
耐電圧	AC1kV 1 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℃、20～85%RH 以内
電源	AC100±10%(AC120、220、240V 切り換え可) AH1108 約 80VA、AH1116 約 170VA DC10.5V～15V・・・AH1108 約 4A、AH1116 約 8A(12V 時)
質量	AH1108：約 9kg

3.3.2　A/D コンバータ

ストレインアンプの出力は電圧値なので、このままではコンピュータに取り込めません。そこでインターフェイス(株)16 ビット A/D コンバータ PCI－3155 図 3.4 によりデジタル量化した後、コンピュータに取り込むようになっています。仕様を表 3.5 に示します。

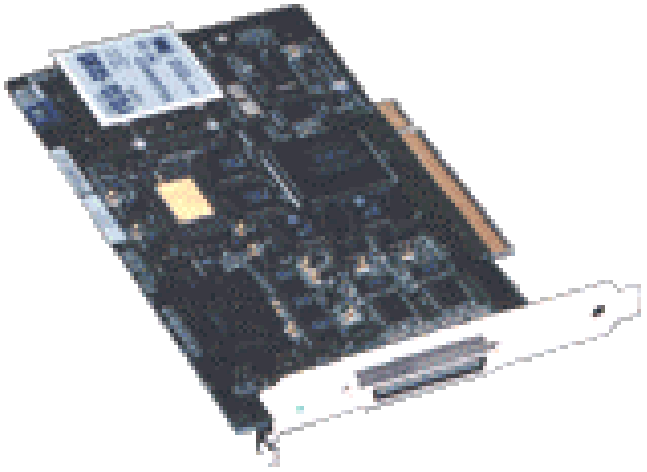


図 3.4　PCI-3155

表 3.5　PCI-3155 使用条件

入力チャンネル数	シグナルエンド入力 16 チャンネル/差動入力 8 チャンネル
入力制御方式	マルチプレクサ方式
入力アクセス方式	FIFO 方式
変換時間	10μs(チャンネル固定時) 10μs/チャンネル(チャンネル切替時)
入力分解能	16 bit
入力レンジ	バイポーラ：±1V,±2.5V,±5V,±10V
絶縁仕様	非絶縁
バス仕様	PCI ローカルバス(Rev.2.1 以上),32bit,33MHz
ボードサイズ	ショートサイズ[174.63(D)×106.68(H)]mm
消費電流	DC＋5V,1A(TYP)
占有 I/O ポート数	32 ポート＋4 ポート
占有スロット数	1 スロット

3.3.3 シリコンラバーヒーターとデジタルファインサーモ

検定実験では熱変形で歪みを検出するために、八光商事社製のシリコンラバーヒーターとデジタルファインサーモ DG2 を用いた。

シリコンラバーヒーターは、薄いシート状なので熱応答性なので優れているだけではなく、柔軟性にも優れているため被加熱物に完全にフィットさせることができるため使用した。

また、温度を安定してコントロールするためにデジタルファインサーモ DG2 を使用した。

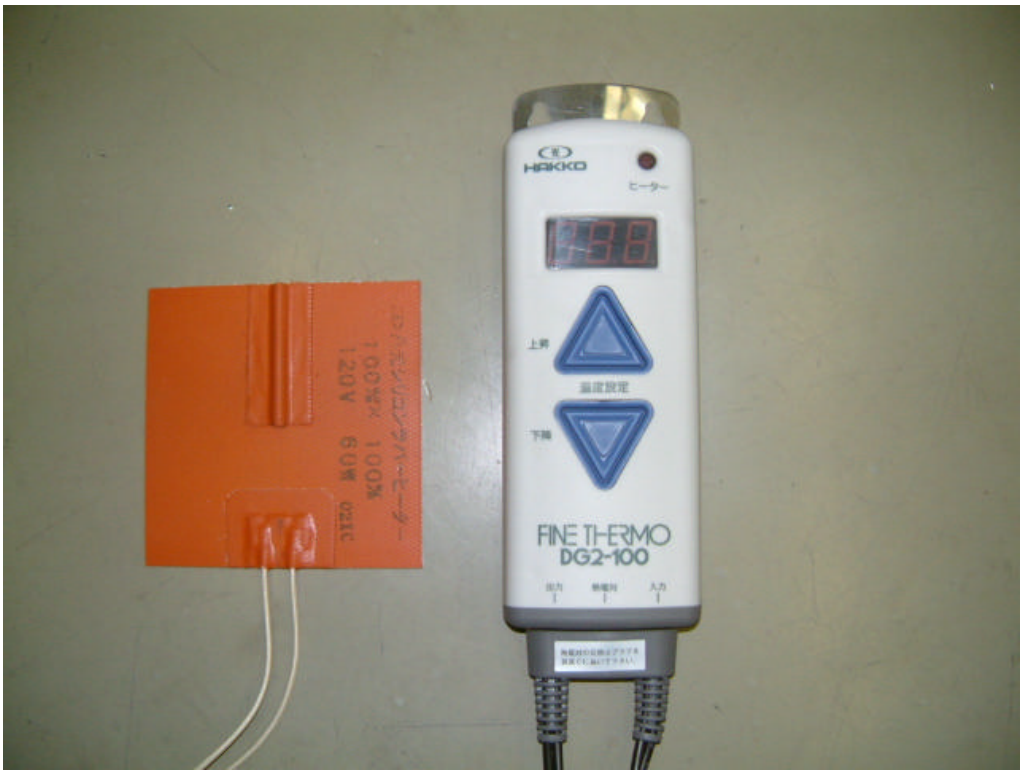


図 3.5　シリコンラバーヒーターとデジタルファインサーモ DG2

3.4 熱アクチュエータ

本研究に用いているような 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になる。主軸の傾きについては NC 機能では補正できないので、マシニングセンタのコラム下部に設置されている熱アクチュエータにより能動的にコラムを変形させ、主軸の傾きを制御している。

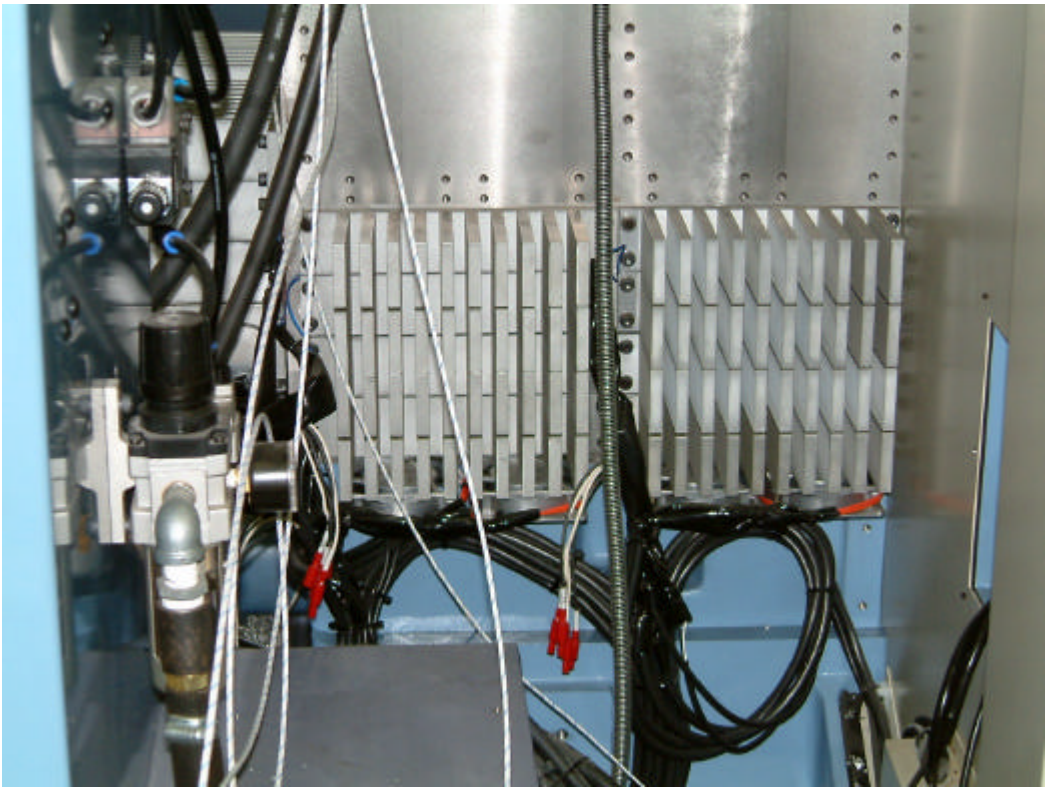


図 3.6　熱アクチュエータ

3.5 ワイヤーカット放電加工機

上記で紹介した機器以外に使用した機器を紹介する。
変形センサを製作する時に使用したのが図 3.5 に示すワイヤーカット放電加工機である。下記でワイヤーカット放電加工機の概要を述べる。

図 3.6　ワイヤーカット放電加工機

ワイヤーカット放電加工機の出現によって、抜き型の工作法が画期的な変化を遂げ、普通の抜き型はもとより、順送り型や極薄板の精密抜き型にいたるまで活用されている。さらに金型の設計製作だけでなく、試作品、部品加工、放電加工電極製作の分野にも広く利用され、試作期間の短縮、部品構造の一体化加工によって複雑な製品の無人化加工に革新を与えた。そのワイヤーカット放電加工の原理を図 3.6 に示す。

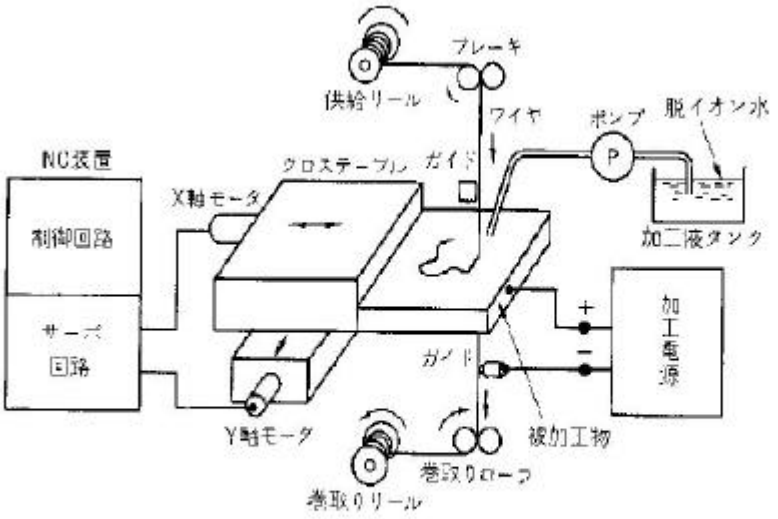


図 3.7　ワイヤーカット放電加工の原理図

被加工物を積載しているクロステーブルは、一般に NC 制御サーボモータ駆動によって X,Y 軸方向に送が与えられて、二次元形状の加工が行われる。

ワイヤは、供給リールから常に一定速度で送り出されており、放電によって生じる電極消耗を補正している。ワイヤ径は通常 0.05~0.25mmΦであるが、加工効率やワイヤ切れトラブルを考慮して、0.1mmΦ以上が多く利用されている。ワイヤは放電圧力などによる強制振動をできるだけ小さく押えるために常に変動しないテンションをかけておく必要があるだろう。

第４章

変形センサの製作

4.1 システムの概略

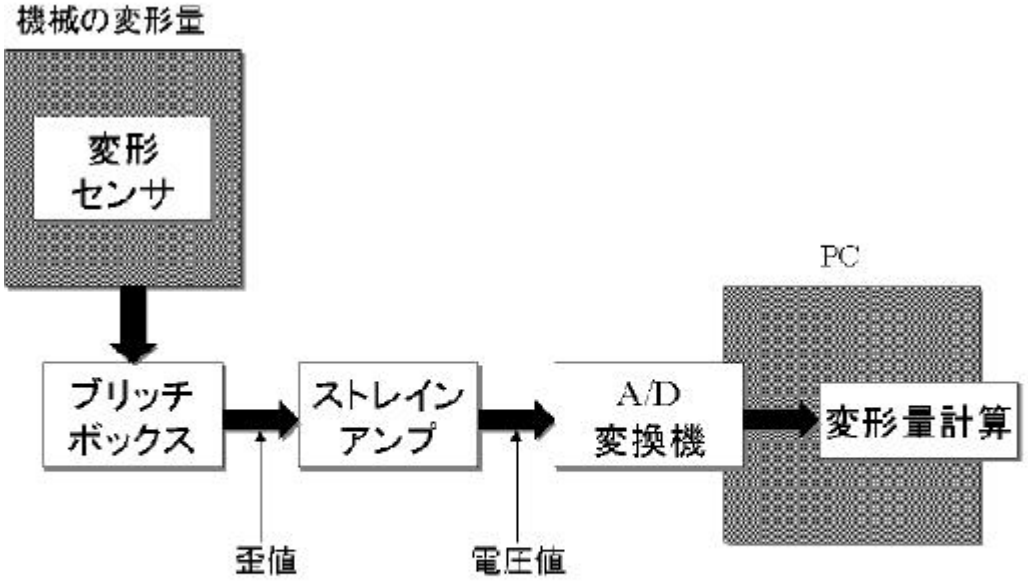


図4．1　変形量検出システム

4.2　変形センサの必要性

工作機械は、工作物を加工する機構とそれを支える機構からなりたっています。加工中に発生する加工反力、熱、音、振動は、機械全体に伝わりとくにコラム部分が変形させられる。しかし、その因果関係を知ることができれば、逆に工作機械に生じた変形量を測れば加工中の状態を知ることができる。

本研究では、変形量を測るために変形センサを用います。それは、２個からなる弾性ブロックと剛体連結棒からなります。片方のブロックには、平行平板構造が組み込まれていてそこに歪みゲージを貼り付けることによって工作機械のたわみを測定する。しかし、機械の設置環境によってそれぞれ変形量が異なるためすべてに適応するとは言いきれません。

変形センサの長所の１つは、構造体の形状や剛性を大きく変える必要がないことだ。また、変形センサのもう 2 つ目の長所は、機械本来の作業の邪魔にならない構造体に装着できることだ。本センサは、作業現場を乱すことなく、必要な運転情報を油出することができる。

短所は、構造体変形と運転状態との因果関係をえるために、変形センサを用いて両者を検定いなければならないことだ。たとえば、マシニングセンタでは設置されている場所によって環境がことなるために機械にかかる熱応力が異な

るためそれぞれ測定しなければなりません。

4.3　変形センサの設計

変形センサの機能は、構造体の２点の微小変形を変形センサ内の平行平板の変形に集中させ、任意の方向の変形を他の方向のそれと干渉させずに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して、その変形をひずみゲージで検出する。

なぜ、ひずみゲージを用いたかといいますと、レーザ測長器と比べるととても格安で時間ドリフトがとても小さいからである。普通、材料の強度を測ったり建設物の変形を知るためにひずみゲージを使いますが、本研究で使用した機械の構造体では、生ずるひずみが小さいので、直接ひずみゲージを貼ったとしても十分な出力が得られませんが、本研究で使用した変形センサは、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な変形でも、ブロックの低剛性部にひずみを集中させることで変形を効率よく検出することができるようになっている。

ひずみを検出するのに問題なのが干渉である。本研究では干渉防止のために、センサの弾性ブロックに平行平板構造を採用した。構造体ではｘ方向とｚ方向に変形が生じても、平行平板構造の容易変形方向のｘ方向だけに変形する。

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によってできる。変形センサの材質を構造体と同じにすると、変形センサにはセンサと構造体の温度が同時に熱膨張差が生じないので力による変形だけが検出できる。しかし、材質が異なると熱膨張差が生じるので、力変形だけではなく熱変形も検出できるようになる。

実際に設計・製作した変形センサを図 4.2 図 4.3 図 4.4 に示します。この変形センサは、平行平板構造を片方のブロックに配して、平板にひずみゲージを貼った物である。

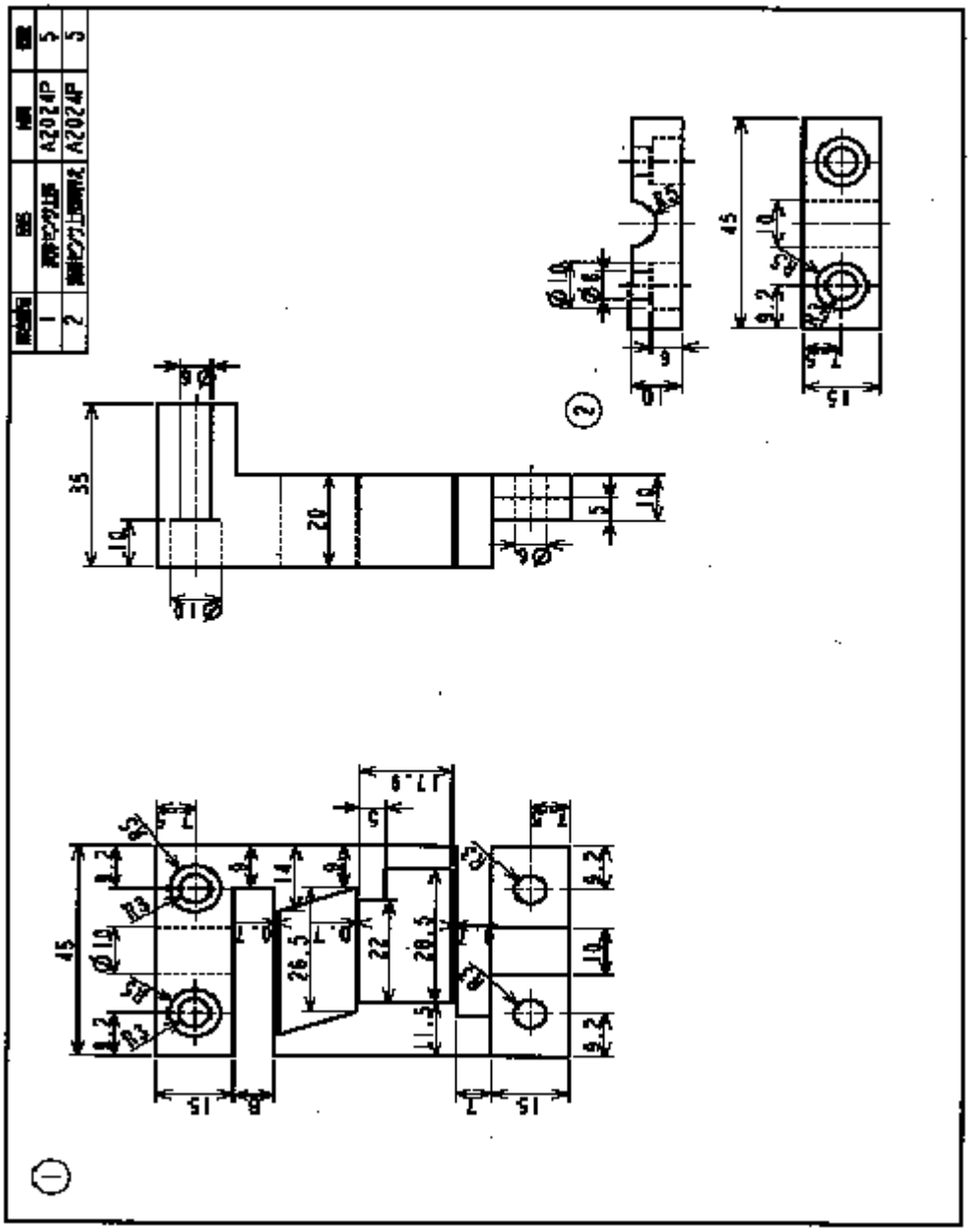


図 4.2 平行平板を使用した変形センサ頭部

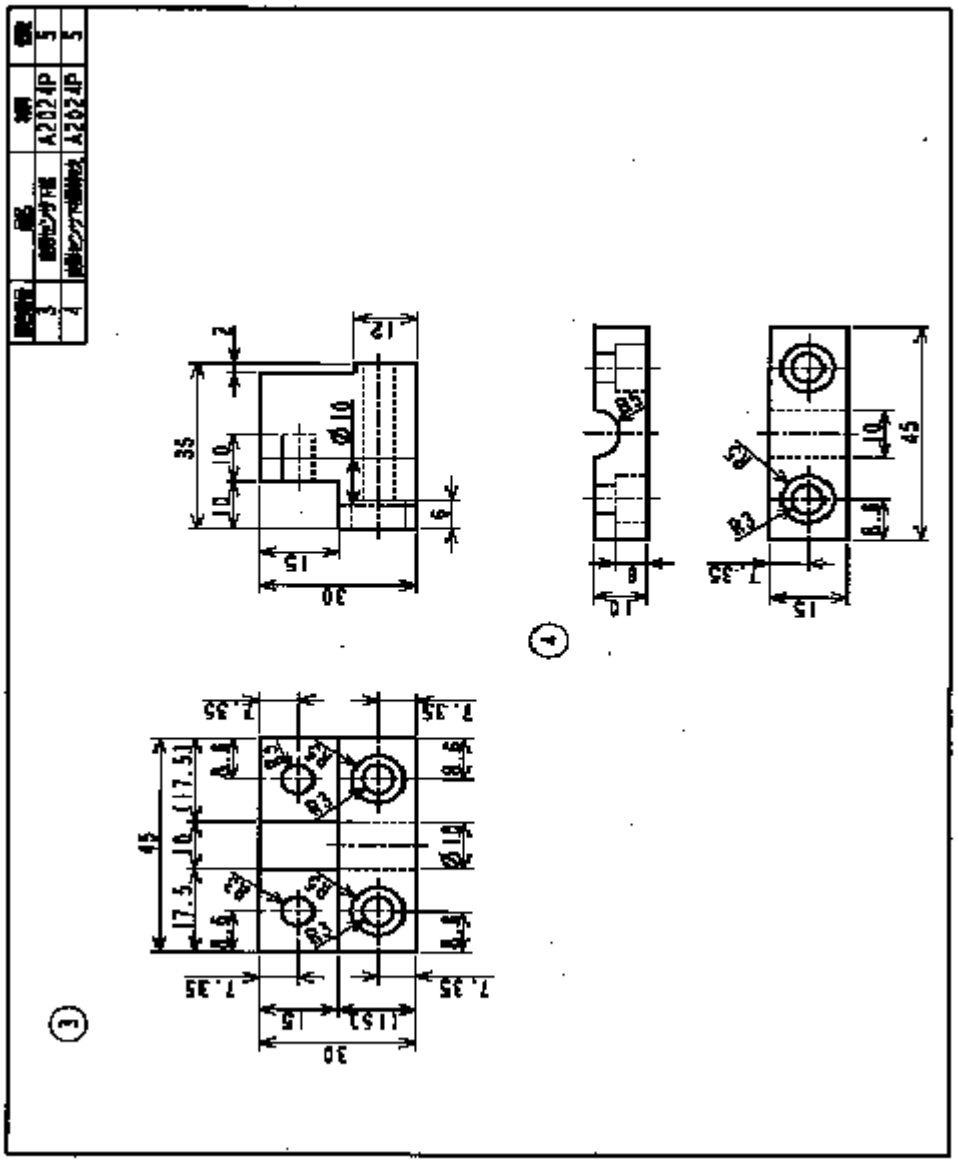
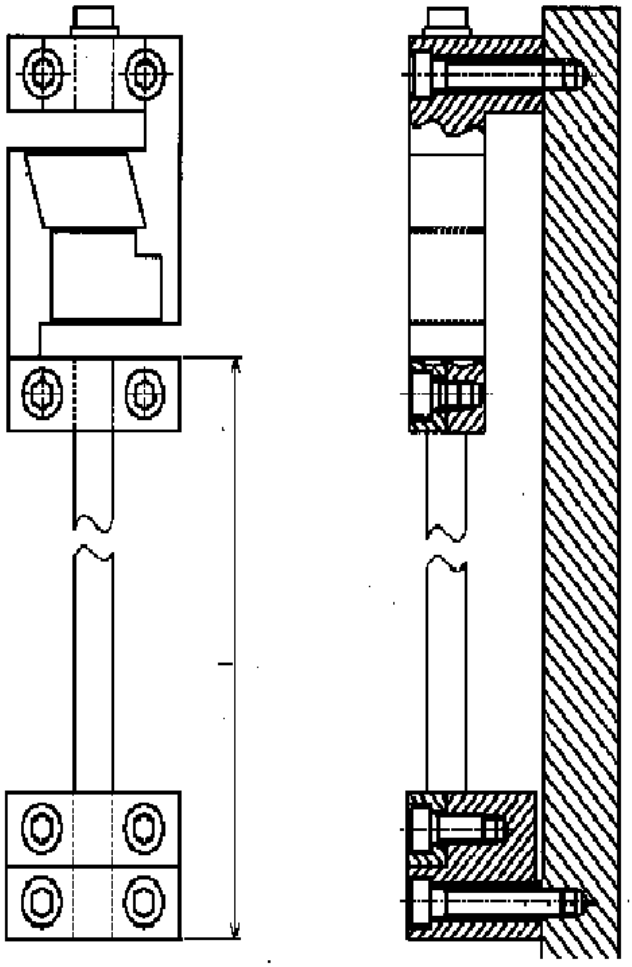


図 4.3　下部変形センサ

変形センサ完成写真を図 4.4 に示す。

図 4.4　変形センサ完成写真



1ch L=458mm
2ch L=218mm
3ch L=558mm
4,5ch L=1023mm

図 4.5 変形センサの構造

4.4 ANSYS による歪ゲージ装着場所の解析

変形センサに歪みゲージをはるため、引っ張り応力と圧縮応力が発生する時に、どの場所に最も歪みがかかるかを ANSYS で解析した。それを図 4.2 図 4.3 に示す。

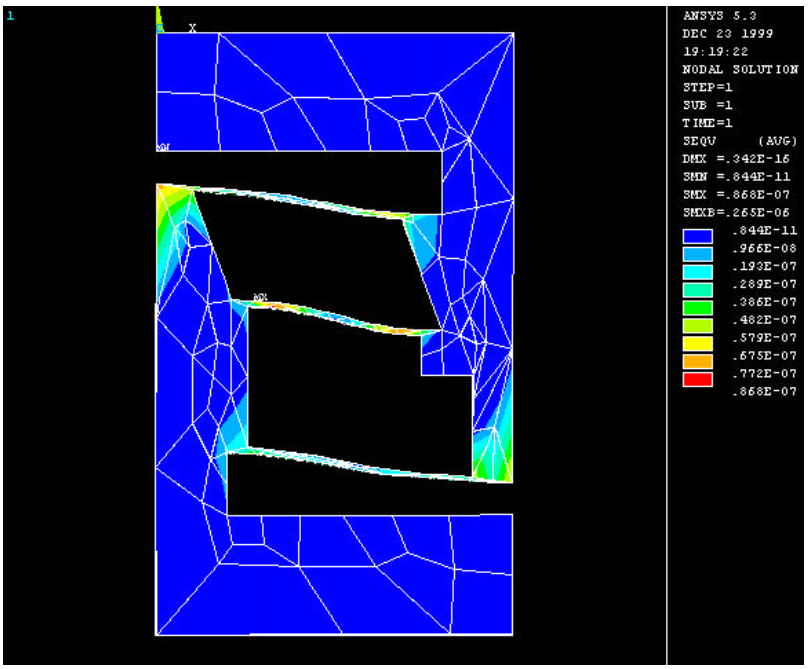


図 4.6 ANSYS による圧縮のシミュレーション

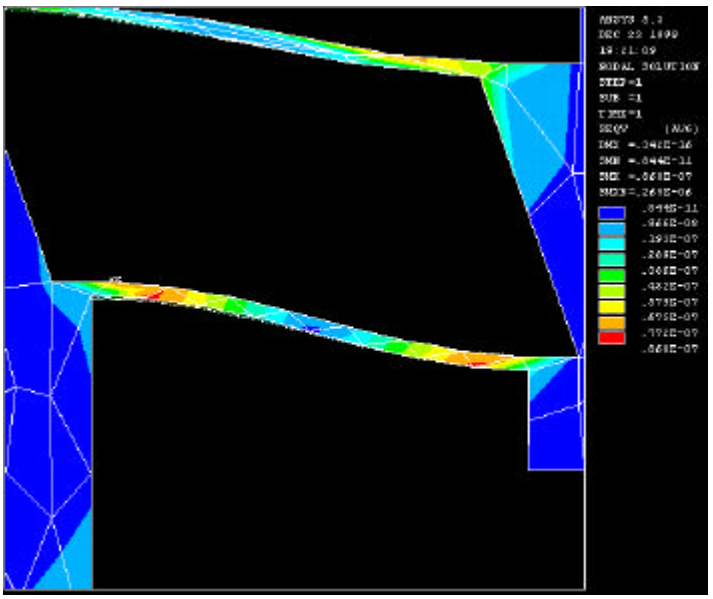


図 4.7 平行平板にかかる応力

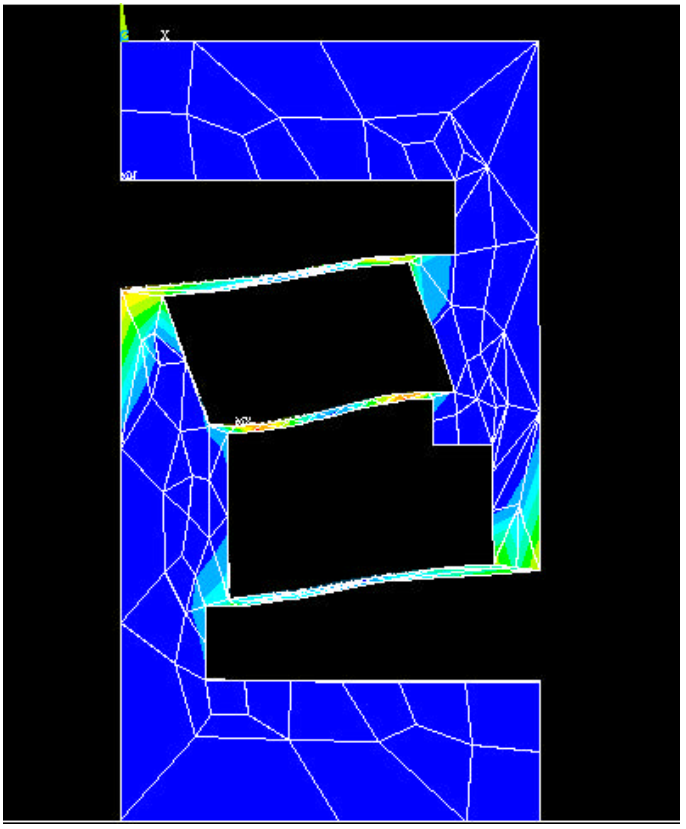


図 4.8　ANSYS による引張りのシミュレーション

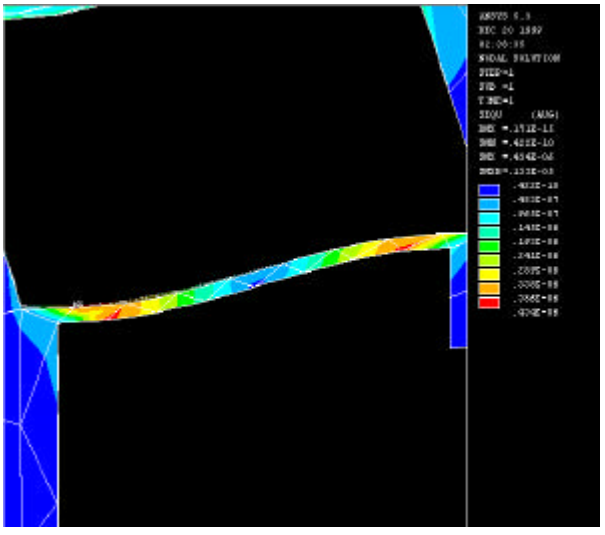


図 4.9　平行平板にかかる応力

4.5 ストレインゲージ

本研究では、KYOWA 製のアルミ用 KFG-2-120-C1-23 ストレインゲージを用いた。その仕様を図に示す。

GAGE LENGTH	2mm
GAGE RESISTANCE	119.8 ±0.2Ω
GAGE FACTOR	2.12 ±1.0%
ADOPTABLE THERMAL EXPANSION	23.4 PPM/℃
TRANSVERSE SENSITIVITY	0.70 %

図 4.1 ストレインゲージの仕様

ゲージ法

本研究では、2 ゲージ法を用いる。

2 ゲージ法は、曲げ歪みだけを分解、測定する時によく用いられる。また、2 枚の歪みゲージが互いに異符号等量の歪みをうけるため、1 ゲージ法の時の 2 倍の出力がとれ、しかも回路上で温度補償できるという特長がある。

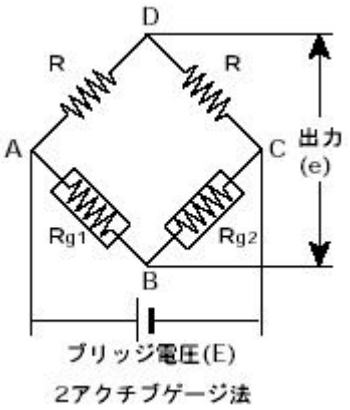


図 4.10 ブリッジ回路

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。

(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。

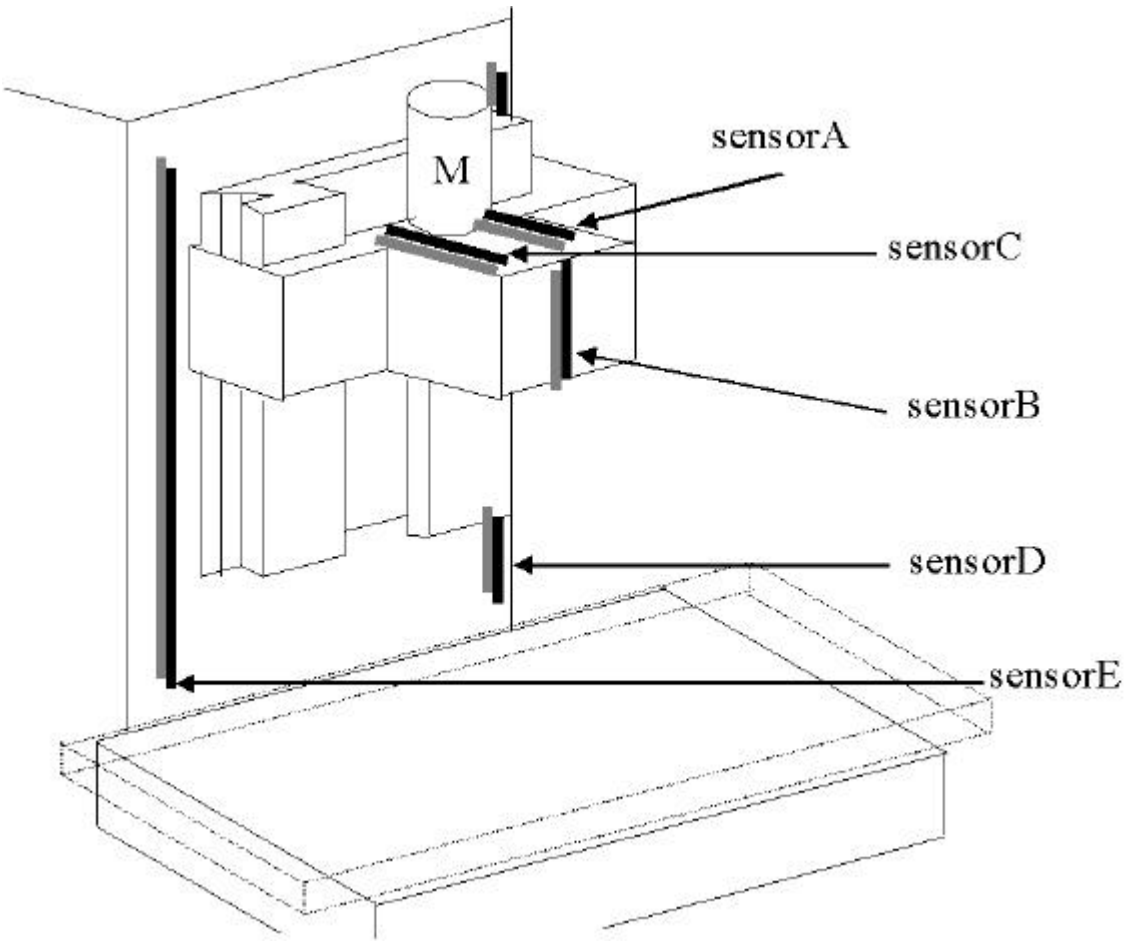


図 4.10 変形センサの位置

上記図 4.10 に変形センサの取り付け位置を示す。

変形量が最も検出されると考えられる主軸部にセンサを３個装着した。またコラム部には２個の変形センサを取り付けた。

第５章

実験

5.1 変形センサに用いる平行平板の 1 次変形の計算式

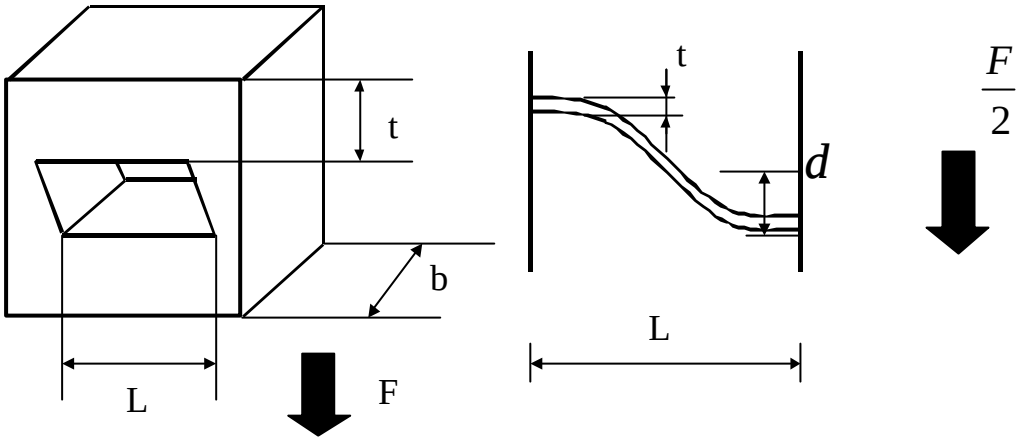


図 5.1 力を加えたときの変形図

平行平板の根元に発生する最大歪み：ε_{max}

$$\varepsilon_{max} = \frac{3FL}{2Ebt^2}$$

検出される歪み

$$\varepsilon_{max} = \frac{6FL}{Ebt^2}$$

したがって、このときの力 F は

$$F = \frac{3bt^2}{6L} \cdot \varepsilon_{max}$$ になる。

また、たわみ式は

$$d = \frac{FL^3}{2Ebt^3}$$ になる。

5.2　変形センサの検定実験

実験目的・実験条件

　この検定実験は、変形センサが熱変形での歪みを検出するかを確かめる実験である。図に示す実験装置の仕組みは、鉄板を熱して撓ませることによって変形センサに歪みを発生させるという仕組みである。

　またこの実験では、30 分加熱・30 冷却の方法をとる。それによって、変形センサが加熱した場合歪みの検出量が増大して行くか、冷却した場合検出量が縮小して行くかを調べる。ヒータは変形センサ間の中心に貼ることにする。

　実験で検出したセンサ A~E の歪みグラフを図 5.3 図 5.4 図 5.5 図 5.6 図 5.7 に示す。鉄板は、SS400 を使用した。

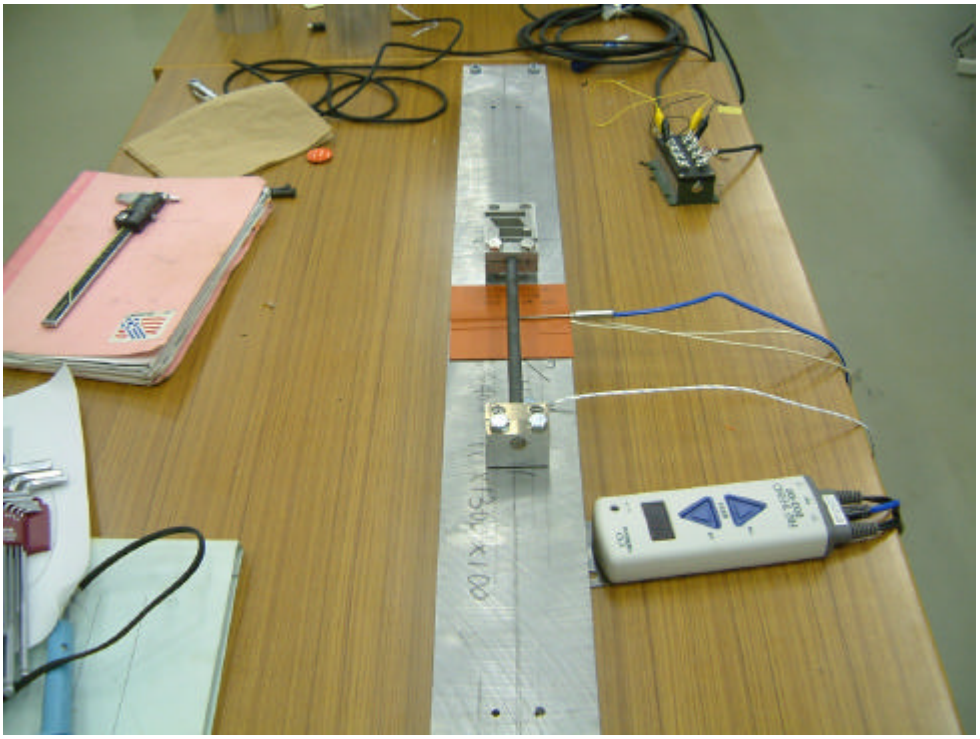


図 5.2　変形センサ検定実験装置

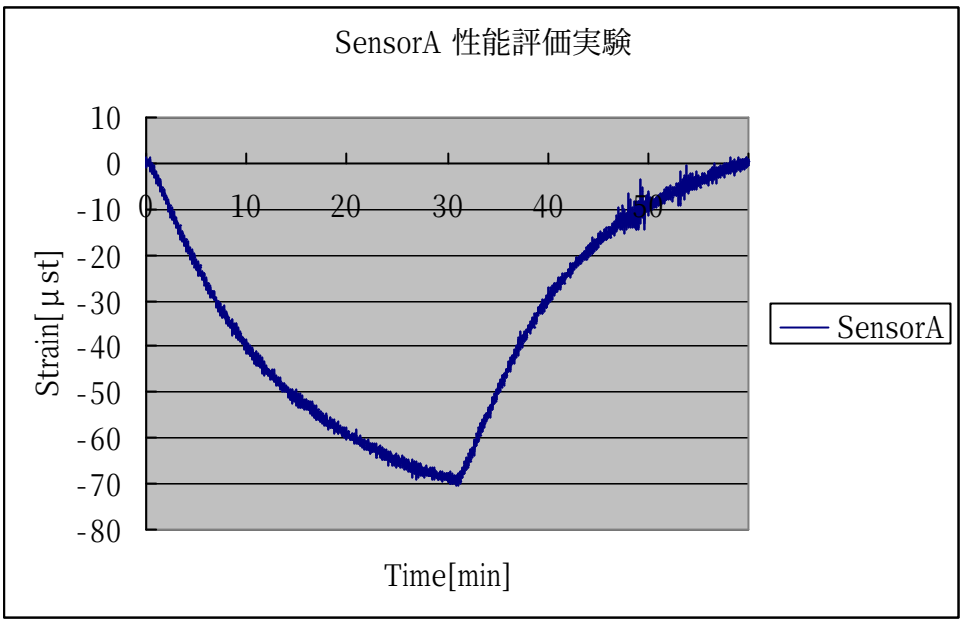


図 5.3　SensorA の性能評価実験

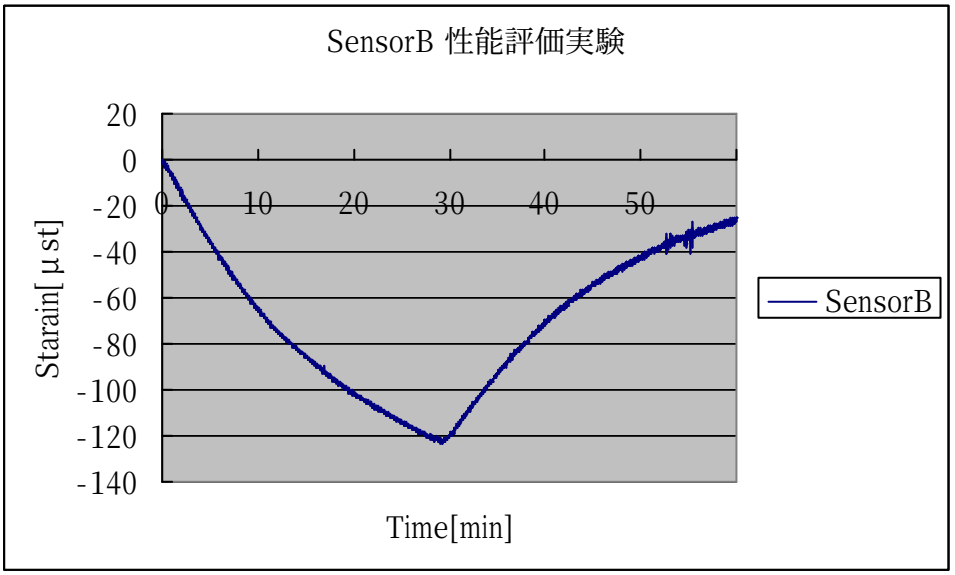


図 5.4　SensorB 性能評価実験

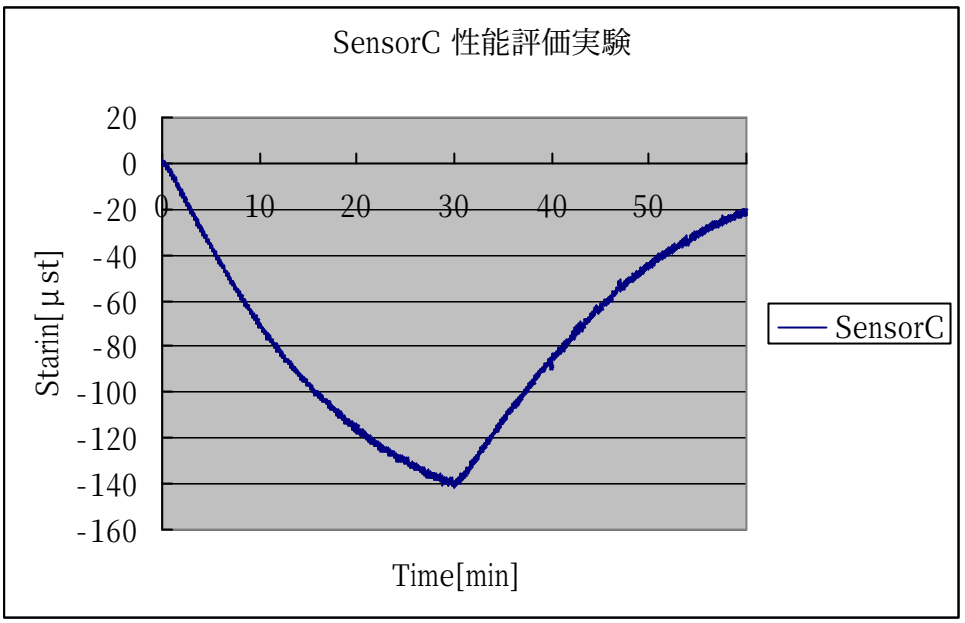


図 5.5 SensorC 性能評価実験

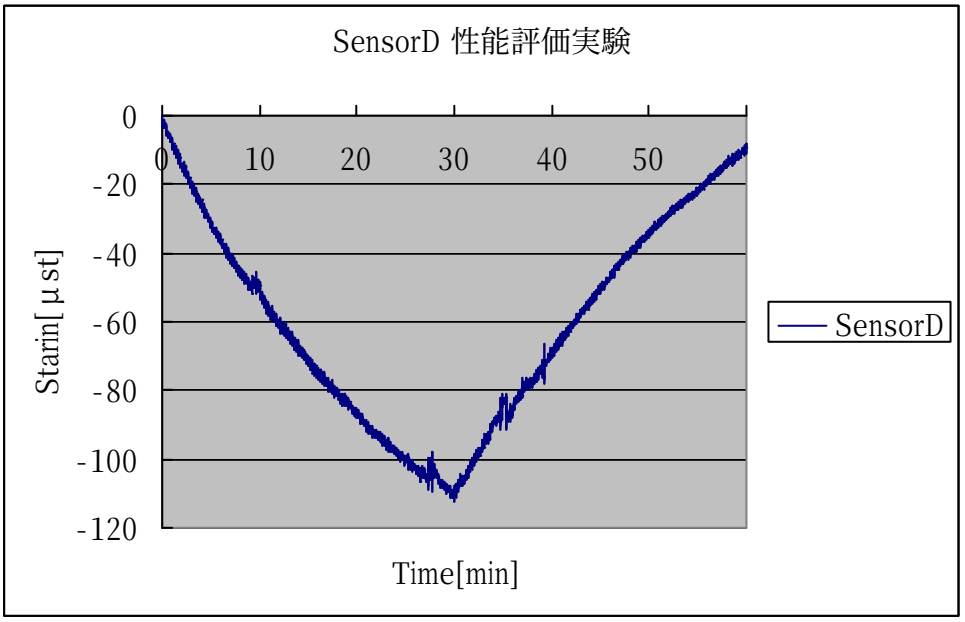


図 5.6 SensorD 性能評価実験

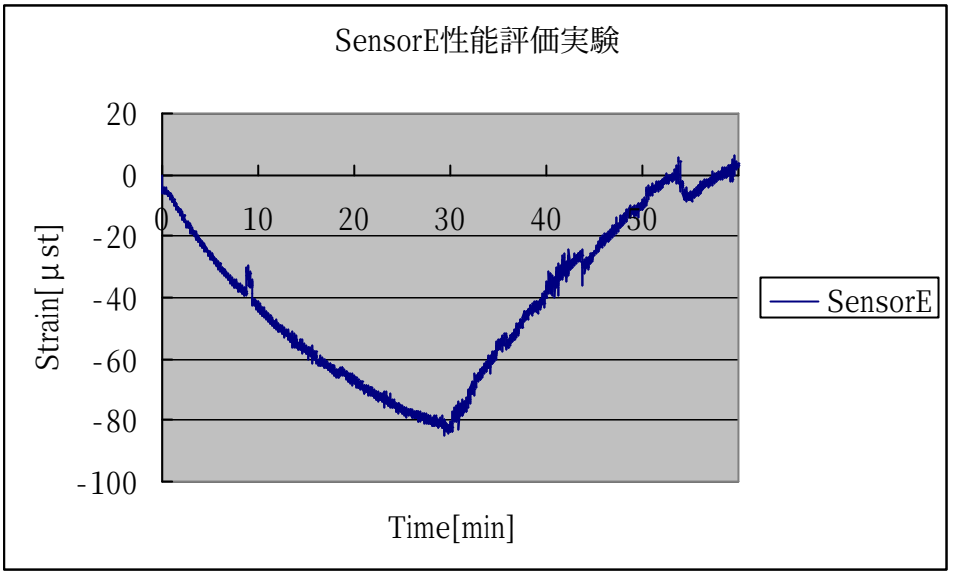


図 5.7　SensorE 性能評価実験

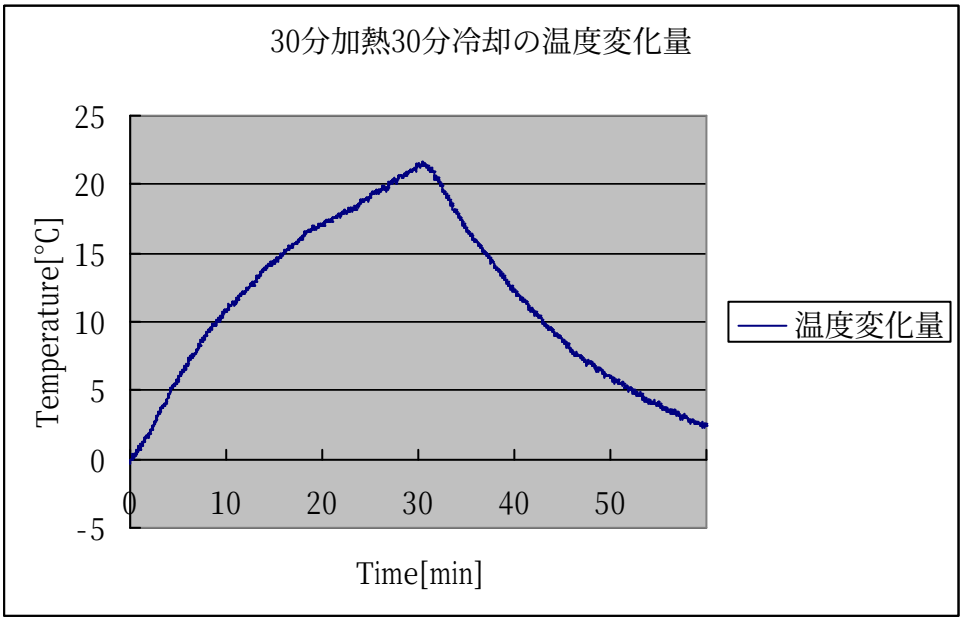


図 5.8　温度変化量

実験結果・考察

① 上記に示したグラフを見ると加熱をした場合には引っ張り力が冷却をした場合には圧縮力が働いていることがわかる。

② この実験では、A~E までセンサを連続的に測ったので同じ温度で測っていない。それを示しているのは、図 5.6 図 5.7 である。センサ D と E は同じ長さのインバーを使用しているが最大歪みは大幅に違うからである。しかし、この実験では、変形センサの性能を調べるためのものなのでそれは、無視をする。

③ センサ 5 つの十分間での歪み量を調べてみると最大で 5μst の誤差が生じることが分かった。

5.3 ANSYS と実験結果との比較

実験目的

5.2 の実験で検出したデータと ANSYS のデータの歪み量との誤差を調べることにより変形センサの信頼性を高める。また、検出実験では、歪みを検出できるかを実験しただけで精度を調べることはできなかったのでこの実験でセンサ一つ一つの精度を調べる。

実験方法

連結棒を着けたままで実験を行ったら変形センサ自体から検出される歪みがでないので外して実験を行った。

実験条件

ヤング率：69Gpa
ポアソン比：0.33
密度：2710[kg/m^3]

重さ(kg)	力(N)	実験歪み(μst)	ANSYS ひずみ(μst)	誤差(μst)
0.13	1.274	3.32601	3.20615	-0.11986
0.26	2.548	6.24022	6.4124	0.17218
0.39	3.822	9.45229	9.6186	0.16631
0.52	5.096	12.815	12.8244	0.0094
0.7	6.86	17.5821	17.2338	-0.3483
0.87	8.526	21.9767	21.4563	-0.5204
1	9.8	23.2349	24.6632	1.4283

表 5.1　センサ A の誤差

重さ(kg)	力(N)	実験歪み(μst)	ANSYS ひずみ(μst)	誤差(μst)
0.13	1.274	2.91599	3.20615	0.29016
0.26	2.548	5.81558	6.4124	0.59682
0.39	3.822	8.92667	9.6186	0.69193
0.52	5.096	11.2975	12.8244	1.5269
0.7	6.86	14.8468	17.2338	2.387
0.87	8.526	18.0784	21.4563	3.3779
1	9.8	21.1827	24.6632	3.4805

表 5.2　センサ B の誤差

重さ(kg)	力(N)	実験歪み(μst)	ANSYS ひずみ(μst)	誤差(μst)
0.13	1.274	3.37248	3.20615	-0.16633
0.26	2.548	6.40578	6.4124	0.00662
0.39	3.822	9.58109	9.6186	0.03751
0.52	5.096	13.0175	12.8244	-0.1931
0.7	6.86	16.8485	17.2338	0.3853
0.87	8.526	21.9694	21.4563	-0.5131
1	9.8	23.7365	24.6632	0.9267

表 5.3　センサ C の誤差

重さ(kg)	力(N)	実験歪み(μst)	ANSYS ひずみ(μst)	誤差(μst)
0.13	1.274	3.45147	3.20615	-0.24532
0.26	2.548	7.19627	6.4124	-0.78387
0.39	3.822	9.31095	9.6186	0.30765
0.52	5.096	12.1279	12.8244	0.6965
0.7	6.86	16.8582	17.2338	0.3756
0.87	8.526	20.6836	21.4563	0.7727
1	9.8	24.2514	24.6632	0.4118

表 5.4　センサ D の誤差

重さ(kg)	力(N)	実験歪み(μst)	ANSYS ひずみ(μst)	誤差(μst)
0.13	1.274	2.65542	3.20615	0.55073
0.26	2.548	5.68905	6.4124	0.72335
0.39	3.822	8.31282	9.6186	1.30578
0.52	5.096	11.0634	12.8244	1.761
0.7	6.86	14.6175	17.2338	2.6163
0.87	8.526	18.2064	21.4563	3.2499
1	9.8	20.2249	24.6632	4.4383

表 5.5　センサ E の誤差

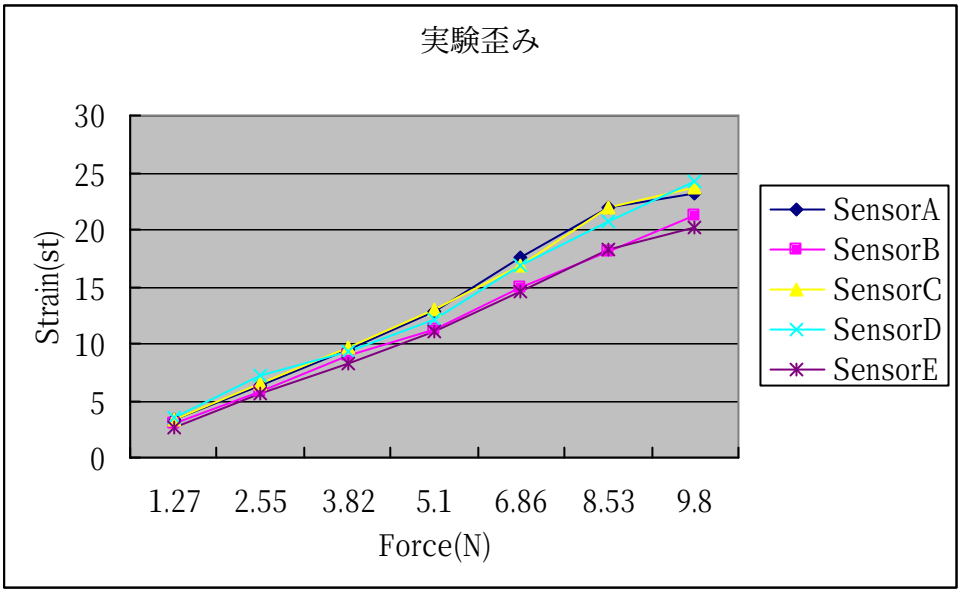
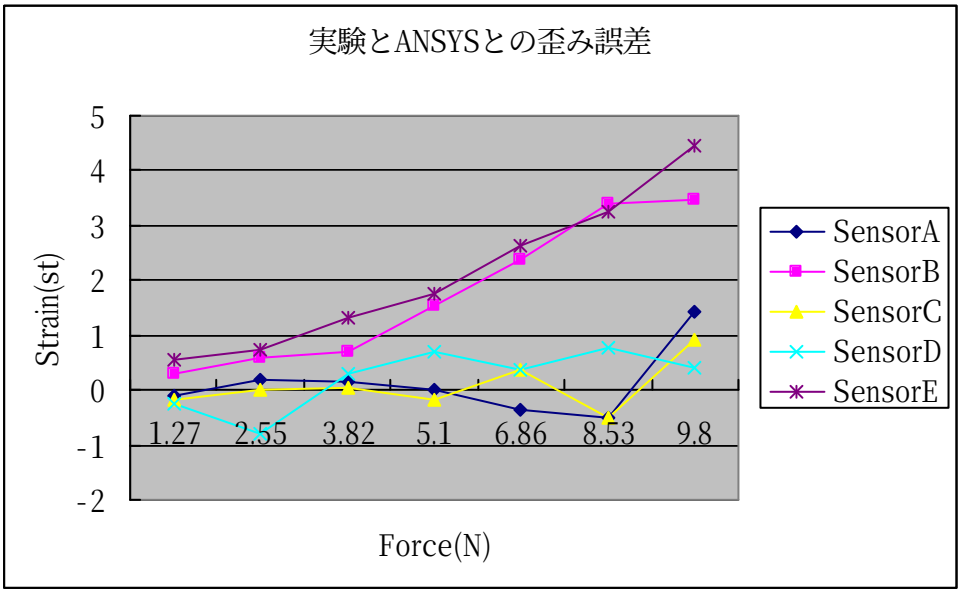


図 5.9　実験での歪み



5.10　実験と ANSYS との歪み誤差

結果の比較・検討

① **センサ** B **と** E **は特に誤差が多いためあまり信用できない。しかし、センサ** A,C,D **は、誤差が少ないので信用して良い。**

② **力と歪みの計算結果と実験結果を表** 5.1~5.5 **および図** 5.9 **図** 5.10 **にそれぞれ示す。**ANSYS **による計算結果と実験結果は、今回実験したすべての荷重に対してそれなりに一致していることがわかる。しかし、荷重が大きくなると、実験結果と計算結果のずれが大きく傾向が見られる。これは、固定端の条件が現実的には実現されにくい条件であることに起因すると考えられる。また、実験装置がかなり原始的なため正確な歪みを検出できなかったとも考えられる。**

5.4　マシニングセンタの測定実験

実験目的

　実際に変形センサをマシニングセンタに装着させ主軸周りがどのくらい歪んでいるのかを知る。

実験条件

　熱アクチュエータを 30 分加熱 30 分冷却させ強制的に歪みを発生させる。

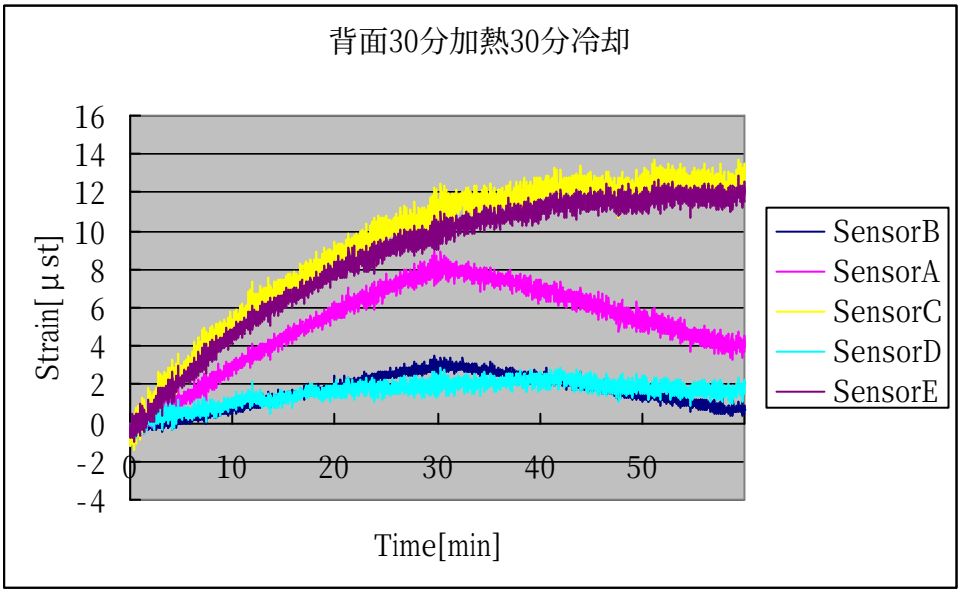


図 5.11　背面 30 分加熱 30 分冷却

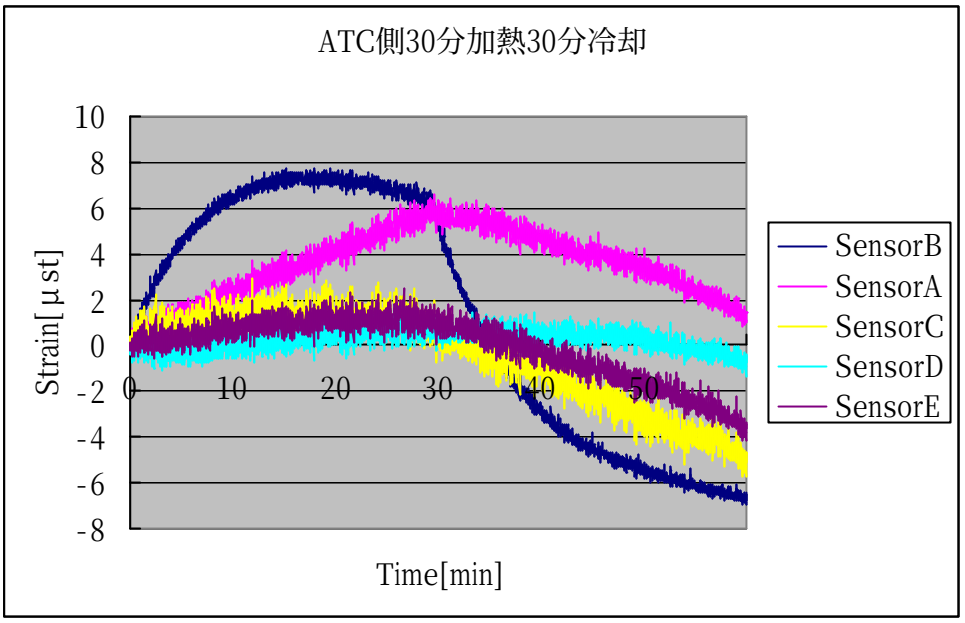


図 5.12　ATC 側 30 分加熱 30 分冷却

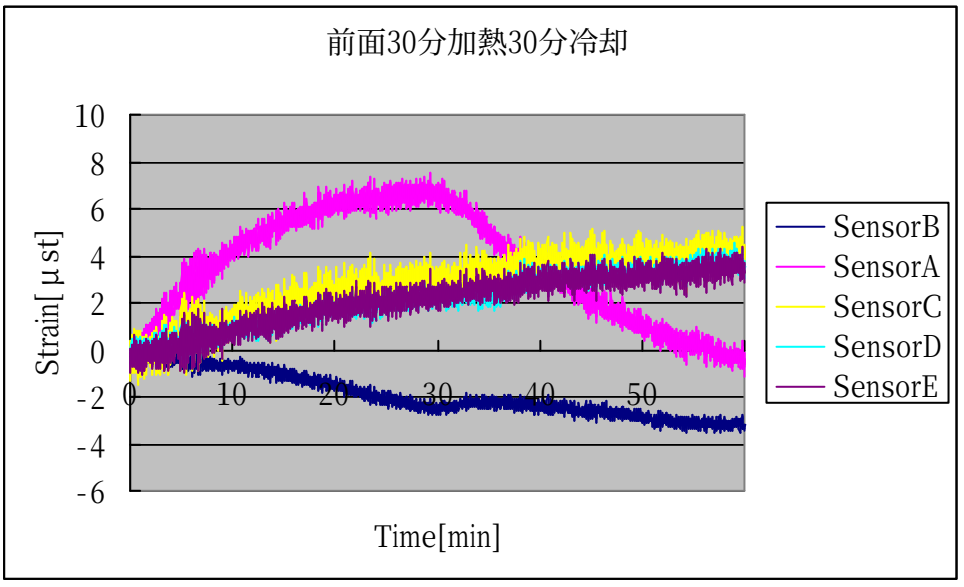


図 5.13　前面 30 分加熱 30 分冷却

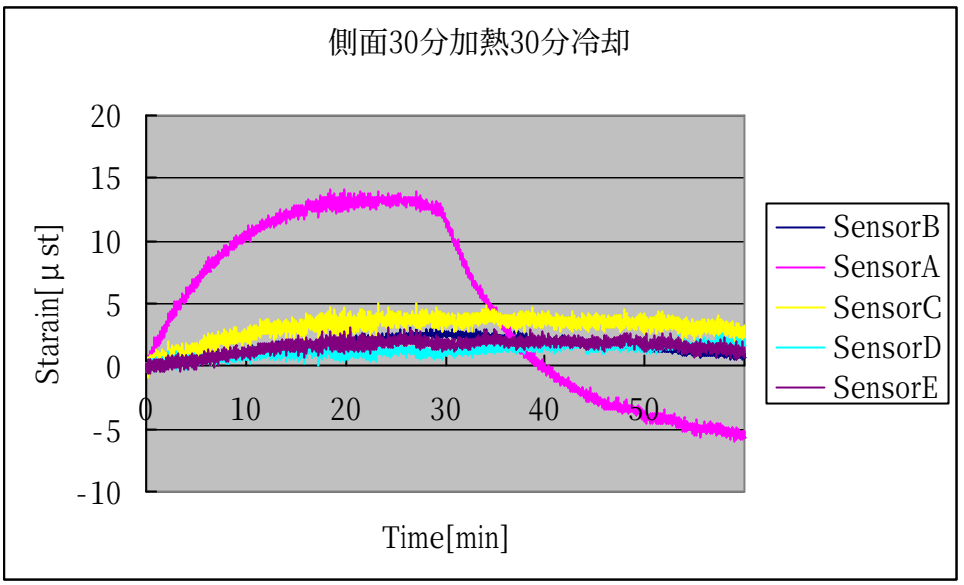


図 5.14　側面 30 分加熱 30 分冷却

実験結果・考察

① 熱アクチュエータは加熱、冷却することで十分変位(歪み)を修正できることが分かった。

② 後から分かったことだがマシニングセンタ自体が漏電していた。それによりノイズが発生したがあまり影響はないと判断した。

③ 図と図を見たら SensorC と SensorE はマシニングセンタを冷却しているにもかかわらず歪みが減少していない。そのことから、SensorC,E は X 軸以外の歪みを検出している可能性がある。

以上のことからニューラルネットワークを用いれば変位(歪み)が修正できるといえる。

5.5 変形センサとレーザ干渉測長機を使用しての測定実験

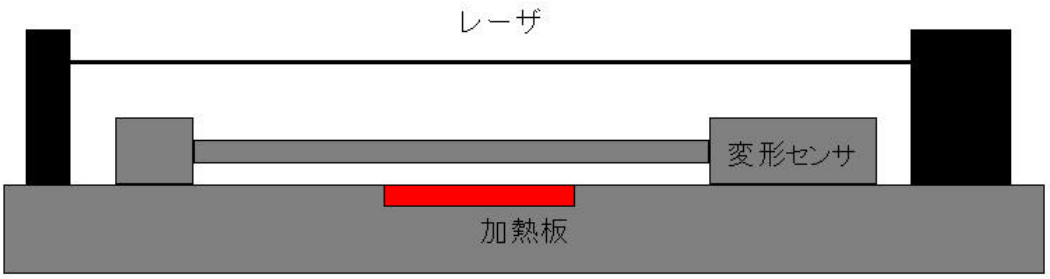


図 5.15　実験装置概略図

実験目的

鉄板を加熱した時の変形センサとの関連を調べるための実験である。

センサ A

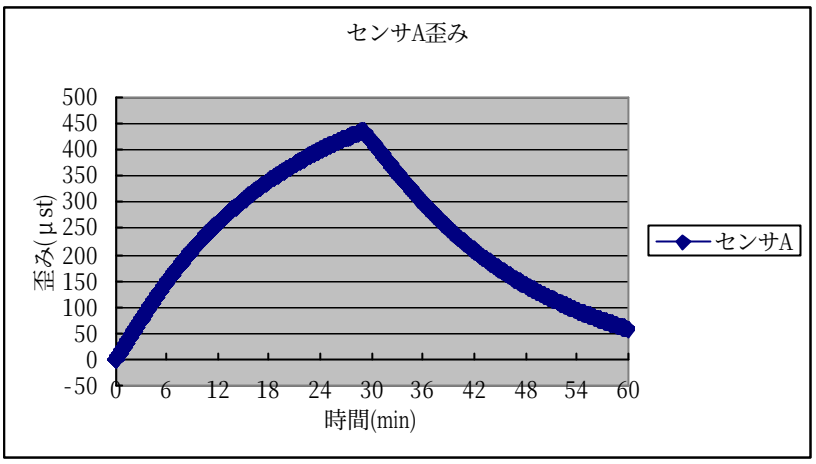


図 5.16　センサ A 歪み

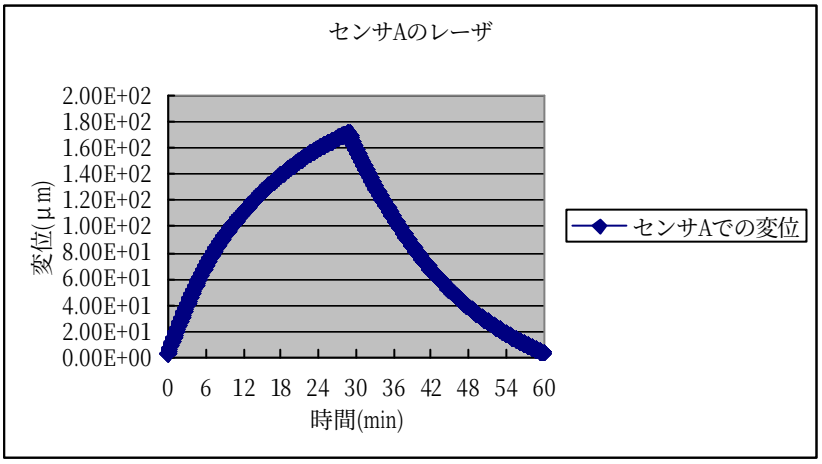


図 5.17　鉄板変位

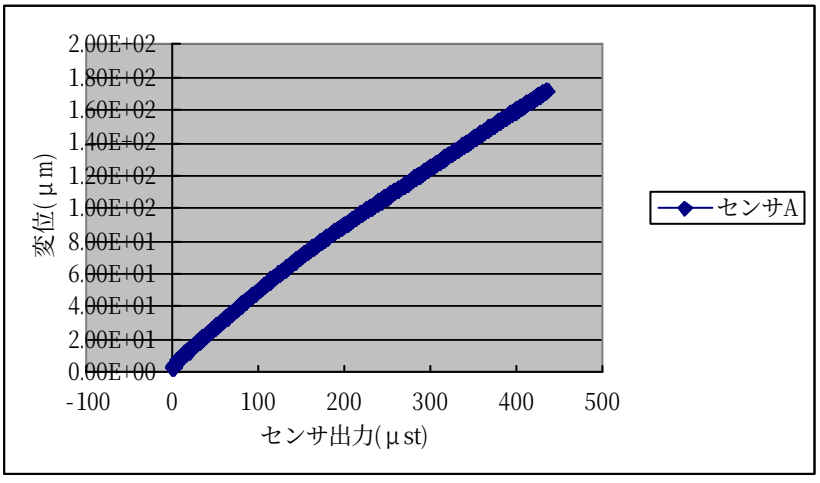


図 5.18　変形センサ A の力特性

センサ B

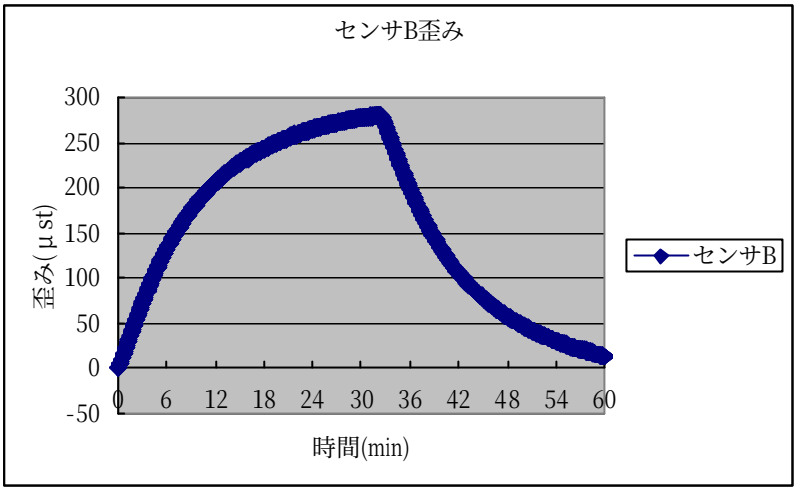


図 5.19　センサ B 歪み

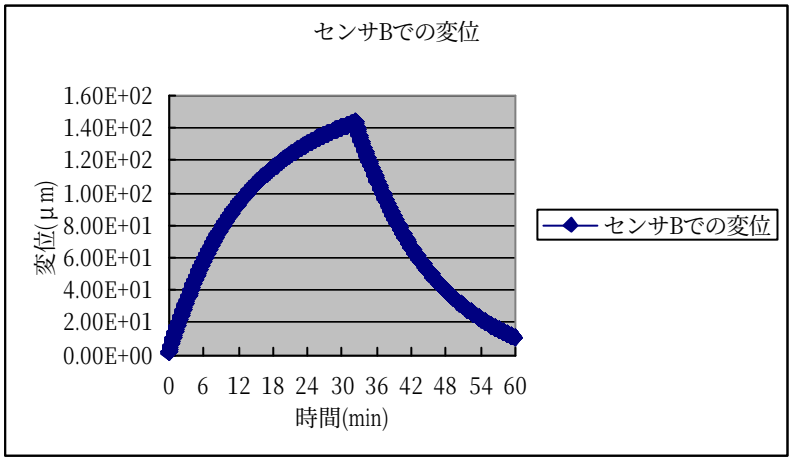


図 5.20　鉄板変位

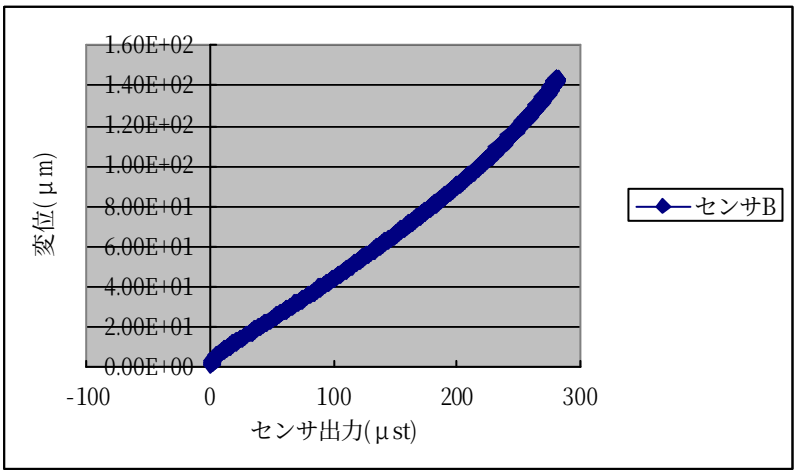


図 5.21　変形センサ B の力特性

センサ C

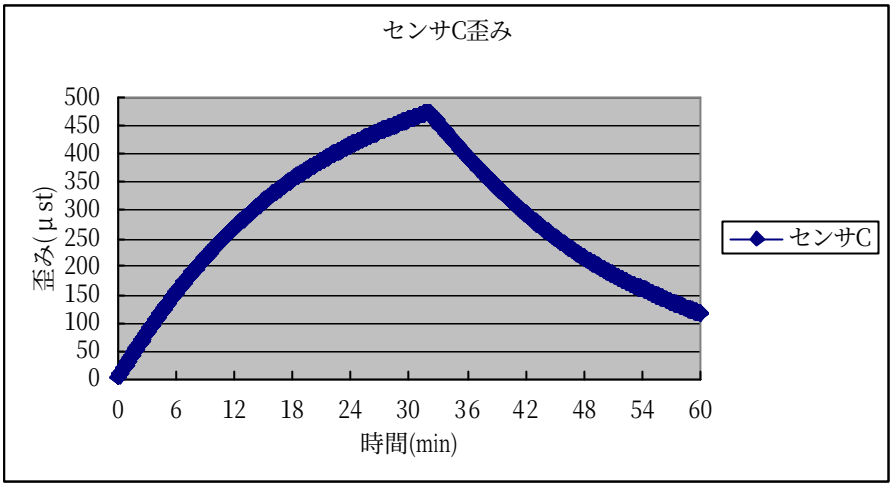


図 5.22　センサ C 歪み

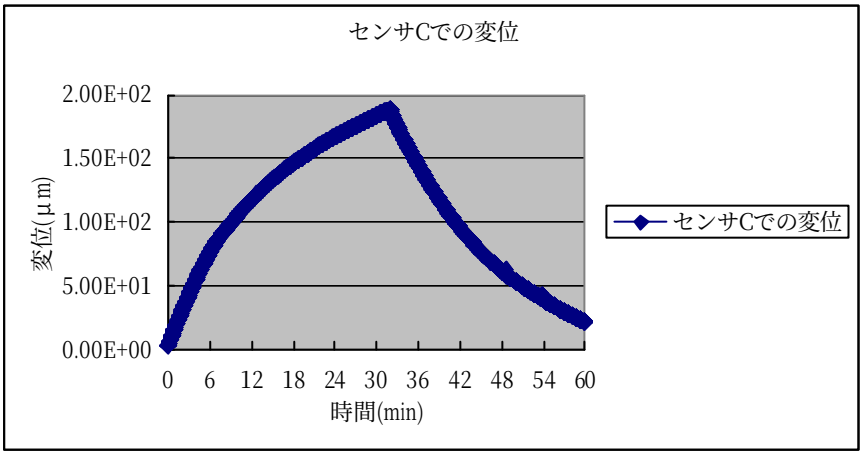


図 5.23　鉄板変位

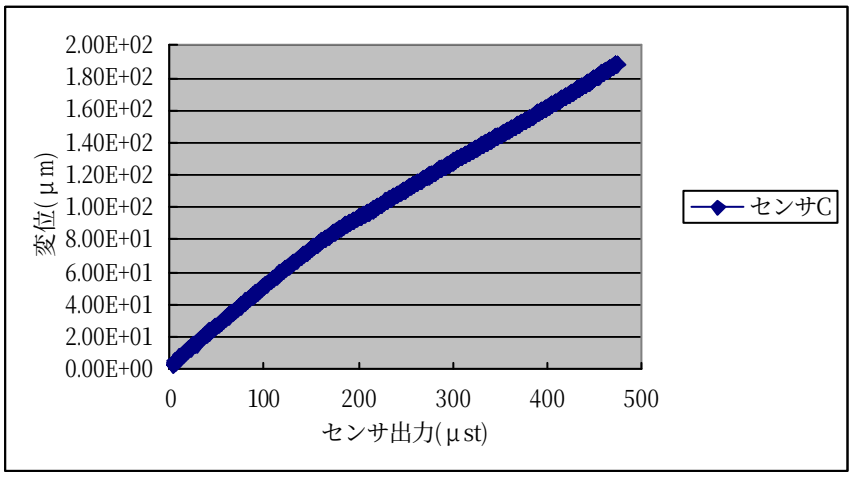


図 5.24　変形センサ C の力特性

センサ D

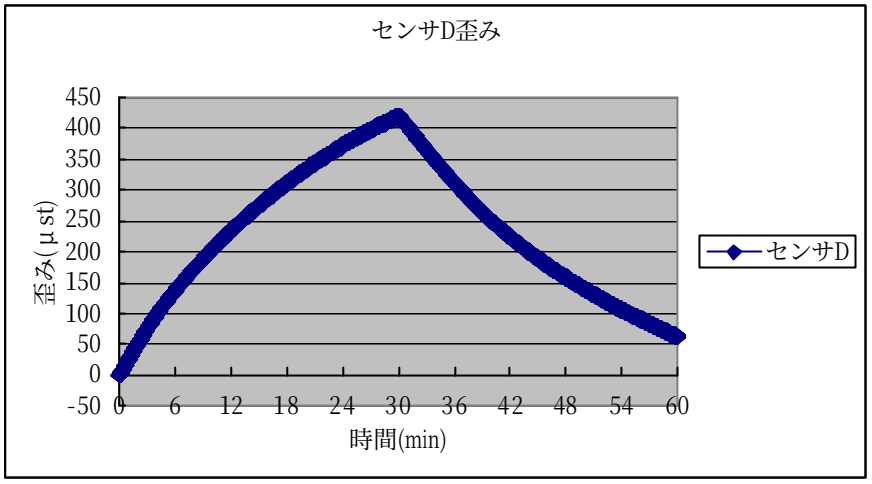


図 5.25　センサ D 歪み

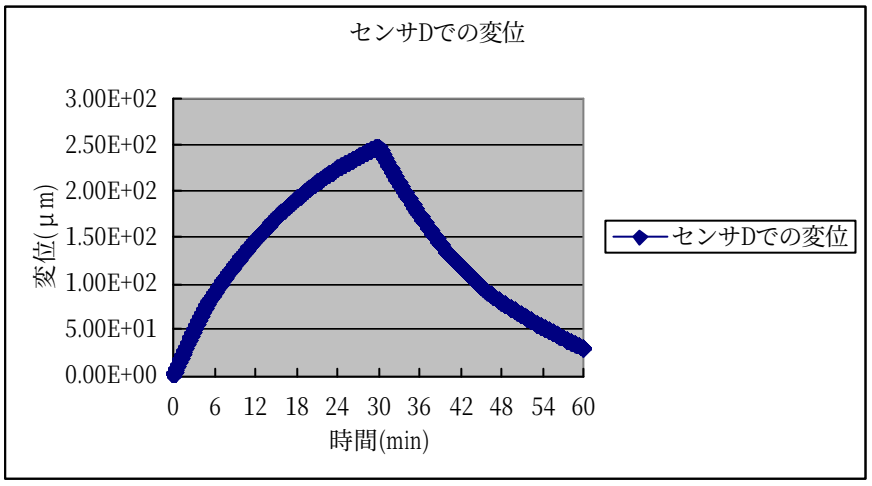


図 5.26　鉄板変位

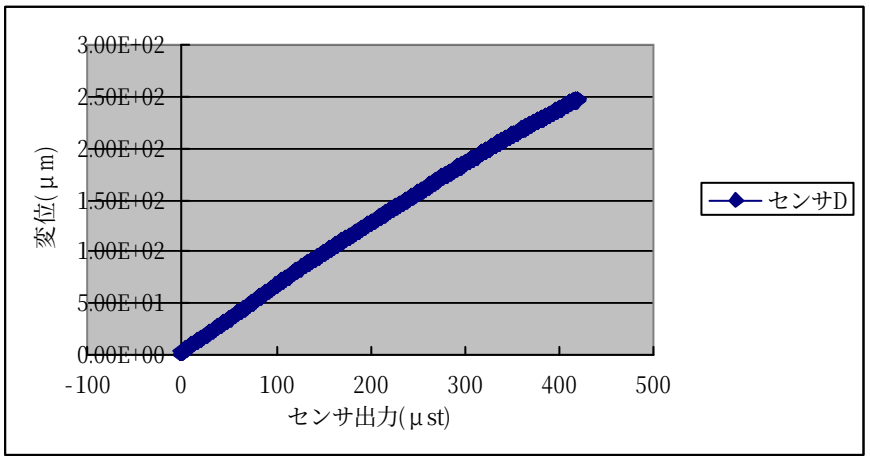


図 5.27　変形センサ D の力特性

センサ E

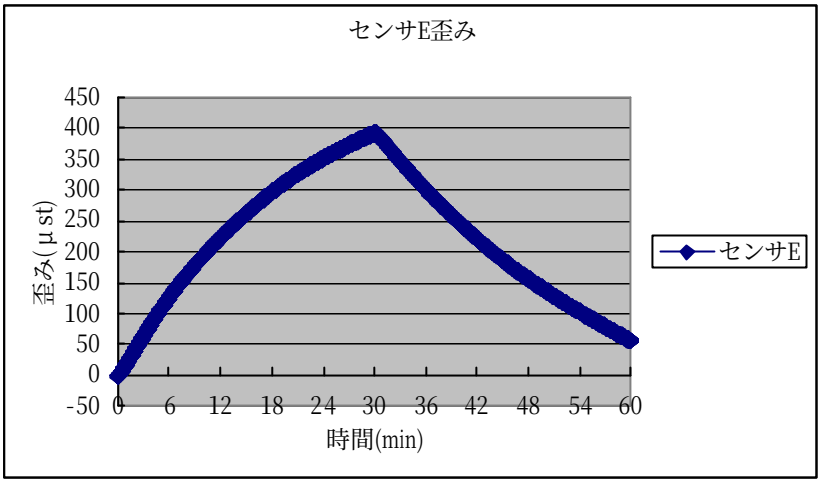


図 5.28　センサ E 歪み

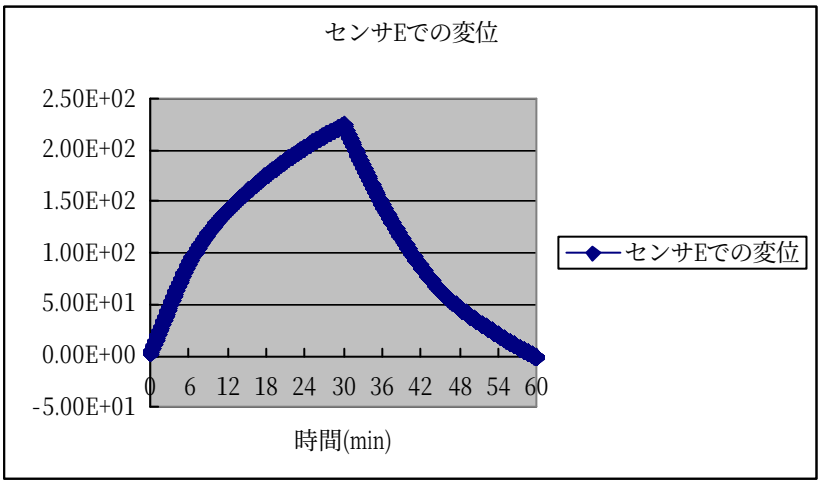


図 5.29　鉄板変位

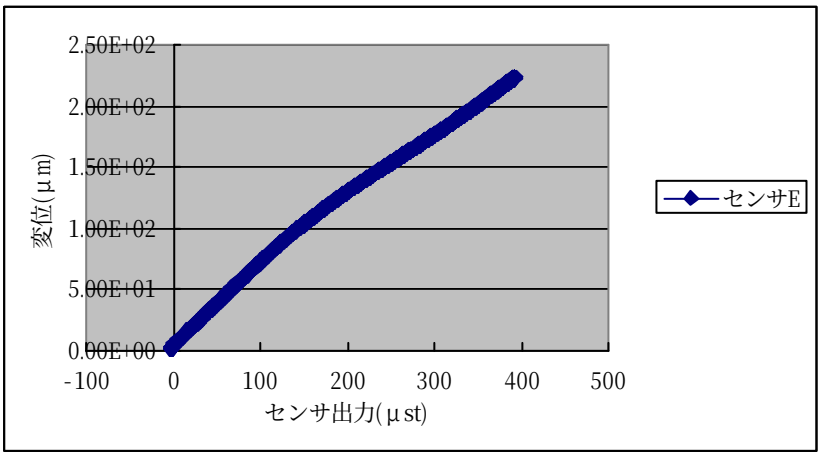


図 5.30　変形センサ E の力特性

結果・考察

① グラフを見るかぎり、鉄板変位グラフと変形センサ歪みグラフは、同じような変化をしていることがわかる。

② また、力特性グラフを見るかぎり線形的なグラフになっていることがわかる。

③ 鉄板の材質・センサの材質・連結棒の材質がそれぞれ違うために、熱膨張差が生じるため、力による変形だけでなく熱変形も検出できたと考えられる。

第6章

展望

展望

① 変形センサと並行してレーザ測長器を使い、歪み・変位を検出し補償して行く必要がある。

② 現在変形センサを取り付けているところ以外に(コラム部)変形センサを取り付けるべきである。

③ 熱アクチュエータは、アクチュエータの操作を行ってから実際にマシニングセンタが補正変形するまでに時間がかかるので迅速かつ正確に補正が行われるシステムを開発する必要がある。

④ 今回は高額なためスーパーインバーで変形センサを作成しなかったが、後輩には作ってもらい、アルミで作成した変形センサとを比較してもらいたい。

⑤ 実験結果を見るとX方向以外の変形を検出していた恐れがあったので、連結棒のX方向と垂直のYとZにノッチをそれぞれ設けて、X方向以外の変形を吸収できるようにしてもらいたい。

第7章

結論

7.1 結論

① **平行平板厚さ** 0.7mm **で、測定に十分な精度の持つものを製作することができた。**

② **今回の研究では、変形センサを製作し歪みを検出することにより、マシニングセンタでどのような変形がおこっているかを知ることができた。**

③ **センサ** B **とセンサ** E **は多少ズレがあるが、グラフを見る限り線形的な特性を持っているので考慮する必要がないと考えた。**

7.2 謝辞

変形センサを作成する過程で、多くの方々にお世話になりました。ここでお礼を申し上げます。

指導教官の長尾先生には、学部 3 年の時からご指導頂きました。どうもありがとうございました。

楠川先生、藤岡さんには、ひずみゲージのことで大変お世話になりました。

小林研究室の橋本さん、小山君には変形センサを製作するにあたり、マシニングセンタでは加工できない箇所加工して頂きました。ありがとうございます。

小山君には、大学院に進まれても研究に励まれることを期待します。

院生の上條さんには研究を終えるまで親身になって相談に乗って頂きました。上條さんがいらっしゃらなければ本研究を終えることはできませんでした。どうもありがとうございました。

共同研究でお世話になった院生の更谷さん学部の浅田君には、苦しみも喜びも分かち合うことができました。一緒に研究ができたことをうれしく思います。

ここに挙げた方々のみならず、たくさんの方々に協力頂きました。本当にありがとうございました。

7.3　参考文献

[1]　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

[2]　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

[3]　長尾高明,畑村洋太郎,光石衛,中尾政之，知能化生産システム,pp42-54,

[4]　畑村洋太郎,小野耕三,中尾政之，機械創造学,pp58-62,

[5]　高橋賞,河井正安　大成社　　ひずみゲージによるひずみ測定入門　歴史から測定まで

[6]　OKK　大阪機工(株)　　VM4-Ⅱ　仕様説明書

7.4　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
%
```

歪み検出 C 言語

```
// 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,yol
d3,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]/65536-5.
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";

          }
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"); */
}
```