正　本

令和元年（ワ）第１０９４０号　損害賠償請求事件

原告　森　次　茂　廣

被告　株式会社[redacted]

第７準備書面

令和３年４月２０日

大阪地方裁判所第２６民事部合議係　御中

原告訴訟代理人弁護士　[redacted]

同　[redacted]

同　[redacted]

（担当）同　[redacted]

第１　被告準備書面５の２（本件プログラム３の著作物性）に対する反論

１　被告は，本件プログラム３について，Timer コンポーネントを利用して，指定した時間間隔でイベントを発生させ，これに対応する処理（イベントハンドラー）を実行することはありきたりな処理方法であると主張する。

この点，データ取得処理に関するソースコードの組み合わせは何を重視するかという点やプログラマーのスタイルによって多様であるところ，一般的な方法としては変換終了イベントでの取得処理の方法が

考えられる。なお，コンテック社のサンプルプログラムでもこの処理の方法が紹介されている。

もっとも，原告はデータと時刻を結びつける際の誤差が生じることを重視して，一般的な方法とは違うタイマーイベントを用いているのであり，それによって作成された第４準備書面１０頁のソースコードの組み合わせは，ありきたりなものではなく，原告の個性が表れていることは明らかである。

２　また，被告は，本件プログラム３について，変数名の記載を漢字にすること，観測開始時間に係る変数を「年」「月」「日」「時」「分」「秒」に分けて管理していること等について思想又は感情を表現したものではない等と主張する。

（１）　この点，原告は本件プログラム３のソースコードについて，ソースコード上で設計書の役割を果たさせるという意味を持たせるために漢字を変数名等に使用しているのであり，原告の思想又は感情がソースコードに表現されているといえる。また，被告が提出している乙２１号証でも「変数名を付けるときは半角英数文字を使うようにしましょう。」と記載されていることから，漢字を用いた変数名をソースコードに記載することはありきたりとはいえず，漢字を用いた変数名を記載することを可能とする原告のソースコードには，原告の個性が表れている。

（２）　また，一般的に時刻を記憶する変数は，日付型でまとめて作成されるため，「年」「月」「日」「時」「分」「秒」に分けて記憶を行うよう設定することは稀である。原告が「年」「月」「日」「時」「分」「秒」に分けて設定しているのは，これまでのプログラマーとしての経験に基づき，日付型が誕生する前のＭＳ-ＤＯ

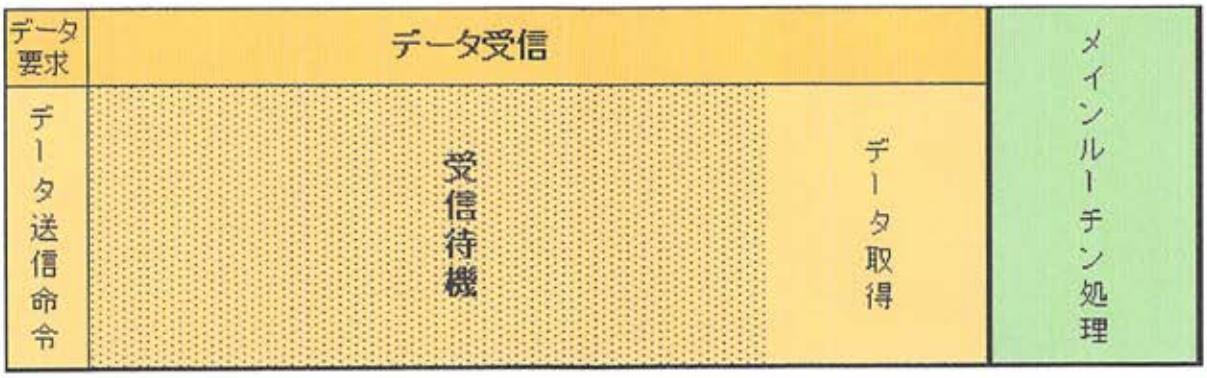
S時代からのファイルフォーマットをソースコードに組み込んでいるからである。したがって，上記のような設定を行うことは，他のプログラマーが作成しても同様になるものではなく，原告の個性が表現されている。

3　さらに，被告は，個々の処理を細分化し同時実行にみせかけることは，オブジェクト指向プログラミングにおいてありきたりな手法であると主張する。

そもそも，個々の処理を細分化し同時実行にみせかけることと，オブジェクト指向プログラミングとは関係ない。なぜなら，個々の処理を細分化し同時に実現することは，後述のとおり効率的な処理を実現できるよう経験に基づきソースコードの構成を工夫することを指しており，一方でオブジェクト指向は，抽象的なプログラミングの考え方を指しており，具体的な命令の組み合わせとは関係しないからである。そして，個々の処理を細分化し同時実行にみせかけることはありきたりな手法ではない。以下，周辺装置からデータを取得する処理を例として説明する。

一般的なデータ取得処理のプログラムの処理の流れとしては以下のようになる。

＜一般的なデータ取得処理の流れ＞



上記図のように，データ要求からデータ受信まで処理を占有してしまい，他の処理ができなくなってしまう。また，データ受信の「受信待機」の部分は何もしない待ち時間になってしまうため，非効率的である。

そのため，「受信待機」の部分を，処理を占有している部分から離すことで効率的な処理をすることができると考えられるところ，その処理を実現する方法としてタイムシェアリングがある。タイムシェアリングを行うには処理を細分化する必要がある。

＜細分化したデータ受信処理の流れ＞

上記の図のように受信待機の時間をメインルーチンに返すことで，データ取得処理とメインルーチンの処理が同時実行されているように見せかけることができる。

原告がこのような処理を行ったのが甲８－１の 58 頁 20 行目～28 行目及び 59 頁 32 行目～60 頁 33 行目のソースコードである。細分化した処理をすることにより、リアルタイムなデータ表示がおこなえており，個々のプログラマーによってどのような構成にするかは異なる。

したがって，個々の処理を細分化し同時実行に見せかけることは一般的なソースコードの構成とは異なり，かつ原告が作成したソースコードには，原告の経験に基づく工夫が凝らされているのだからありき

たりなものではない。

4　また，被告は実数値を整数値に変換して演算することはプログラミングにおいてありきたりに行われる手法と主張しているが，データを整数型で処理するか，実数型で処理するか，また演算をどのタイミングで行うかは，重視する目的やプログラマーの考え方によって選択肢は多岐にわたる。そのため，実数値を整数値に変換するソースコード一つとってもありきたりなものにはならず，創作性が認められる。

5　小括

被告が反論で指摘しているのは，どれも手法について創作性はないというものであるが，命令を組み合わせたソースコードを指摘しなければ反論にならず，失当である。

第２　被告第６準備書面に対する反論

１　本件プログラム４について

（１）　被告は，本件プログラム４について，気象庁の気象観測統計指針に基づき被告が発注した仕様に従ったアルゴリズムであるため，著作物性が認められないという旨主張する。

しかしながら，コンピュータプログラムはそもそもアルゴリズムをコンピュータによって自動化及び高速化することを目指し，命令を組み合わせて作成されるもの（ソースコード）である。そして，アルゴリズムを実現するための命令の組み合わせには，プログラマー毎の知識・経験や何を重視するかという点によって選択肢に幅が生じ，ありきたりなものにはならない。

したがって，被告が発注した仕様に従ったアルゴリズムであるから著作物性が認められないということにはなりえない。

（２）　また，被告は，Ｗｉｔｈ文やグリッドコントロールを使用することは，ありふれた手法であり，創作性が認められないと主張する。

しかし，そもそもＷｉｔｈ文やグリッドコントロールを使って命令の組み合わせを作るか，これらを使わないで命令の組み合わせを作るかという点もプログラマー毎の選択によるものであり，これらを使用する選択をしたとしても，どのような命令の組み合わせとするかという点にはプログラマー毎の個性が表れるため，ありきたりなものにはならない。

なお，原告は，本件プログラム４において，グリッドコントロールを２次元配列の代用として用いている（甲１１号証－１　１６頁９行目乃至３８行目参照）。２次元配列の代用としてグリッドコントロールを用いることは一般的な手法ではなく，独特のアイデアと考えられている。そして，原告は，グリッドコントロールを用いることで本件プログラム４について機能の実現とデータの見える化を実現している。

したがって，この点からもグリッドコントロールを用いた本件プログラム４のソースコードは，ありきたりなものではなく，創作性が認められる。

（３）　また，メモリー機能，高速サンプリング，連続サンプリング，トリガ収録，プリトリガ収録等を実現しているソースコードについては以下のとおりである。

ア　メモリー機能，トリガ収録，プロトリガ収録について

上記の機能について記載しているソースコードは，甲１１－１号証の１５頁１６行目乃至１６頁４０行目の箇所である。これら

のソースコードにより，ＡＤＣから電圧値のデータを取得し，物理的に係数計算した上で，上述のグリッドコントロールに数値を代入している。その際，数値の代入にポインターを使い，グリッド内にリングメモリ構造を作成することで，トリガ収録及びプリトリガ収録が可能となっている。

イ　高速サンプリング，連続サンプリングについて

上記について，厳密に特定するとソースコードは以下のとおりである。

㋐　高速サンプリング及び連続サンプリングの設定

甲１１－１号証１３頁８行目乃至４２行目

㋑　連続サンプリング処理に必要な，ＡＤＣの状態の取得

甲１１－１号証１４頁２５行目乃至１５頁５行目

㋒　高速サンプリングを実現するための状態にあったデータ数の取得（変換データの取得）

甲１１－１号証１５頁６行目乃至１５行目

㋓　連続サンプリングを行うためのイベント処理

甲１１－１号証９１頁２行目乃至９２頁１４行目（グリッドことロールに代入されたデータから最新のデータを取得し，画面にリアルタイムのデータとして表示する等の処理を行う。）

２　被告第６準備書面　第２及び第３の部分について

追って反論する。

第３　原告の主張

１　本件プログラム５の著作物性

（１）　本件プログラム５の概要

本件プログラム５は，アナログ入力ボード（以下，「ＡＤＣ」と言います。）の機能のテストを行うためのプログラムであり，開発プログラムが正常にデータを取り込んでいるかを確認する機能を有している。また，ＡＤＣを最大４枚までコントロールすることができることから，チャンネル数として最大１２８ｃｈ（３２ｃｈ×４枚）の動作確認を行うことが可能となり，スムーズな開発を支援している。

（２）　ソースコードの構造について

本件プログラム５の主な特徴として，アナログ信号入力処理，スケジュール測定，動的測定及び静的測定のデータ確認等が挙げられる。そして，本件プログラム５のソースコードの構造については，後述するが別紙のとおりとなる。

なお，本件プログラム５のソースコードについて，第５準備書面第１の１（３）で述べたとおり，甲１３－１号証について修正したものを甲３０号証として提出する。

（３）　制御するＡＤＣの個数について（別紙①）

ア　選択肢に幅があること

本件プログラム５を作成するにあたって，まず制御するＡＤＣの個数をいくつにするかという問題があるところ，別紙①のとおり，選択肢としてシングルボード制御（単体），マルチ（２枚）ボード制御，マルチ（４枚）ボード制御が考えられた。

シングルボード（単体）のＡＤＣでも，１６チャンネルの入力が可能であり，シングルボードを選択しても問題はない状況ではあったが，原告は将来的な面や自身のスキルアップも兼ねてマルチ（４枚）ボード制御を選択し，「For i = AD_LBound To AD_UBound」

（甲３０号証　７８頁（３８行，５１行，６５行），７９頁（１０行），８０頁（３８行，５８行），８１頁（２６行，５５行，６７行））のコードのループ文を作成した。この「For i = AD_LBound To AD_UBound」のループ文により，４枚のＡＤＣから制御するＡＤＣを切り替えることが可能となっている。したがって，シングルボード制御を選択した場合は，このようなループ文を用いたソースコードを作成する必要はなく，マルチ（２枚）ボード制御を選択した場合も，ソースコードは甲３０号証７８頁～８１頁のような複雑なループ文とはならない。

また，原告は，以下のソースコードも作成している（甲３０号証　７８頁１０行目乃至２６行目）。

＜原告が作成したソースコード＞

```
Public Sub AD_ControlCount()
Dim i As Integer, dmyBoardName As String, NotCtrl As Boolean
On Error GoTo OcxError
AD_LBound = 16: AD_UBound = -1
For i = 0 To 15
dmyBoardName = fMainForm.ctrlAD(i).BoardName
If AD_LBound > i Then AD_LBound = i
If AD_UBound < i Then AD_UBound = i
NextLine:
If NotCtrl Then Exit For
Next
On Error GoTo 0
Exit Sub
```

```
OcxError:
NotCtrl = True
Resume NextLine
End Sub
```

このソースコードは，制御するＡＤＣの数を検出するためのものであり，シングルボードやマルチ（２枚）ボード制御を選択した場合には，ＡＤＣの個数が異なることから，ソースコードの表現も上記から変化する。

よって，制御するＡＤＣの個数を選択することについて，ソースコードに選択肢の幅がある。

イ　原告の個性が表れていること

上記アで述べたループ処理は，活性化の判定を行いながらボードの同期を行うものであるところ，活性化の判定は「AD_Enabled」の配列に記録された状態で，「AD_BoardFlag」とボードステータスを表示するランプの２つによって行っている。

「AD_Enabled」による判定のソースコードは以下のとおりである（甲３０号証　７９頁　１１行目乃至２３行目）。

＜「AD_BoardFlag」判定について原告が作成したソースコード＞

```
If AD_BoardFlag(i) And (.pbtmAD(i).Button = Button_Green Or .pbtmAD(I).Button =
Button_Yellow) Then
AD_Enabled(I) = True
.txtChannelString(I) = .ctrlAD(i).ChannelString
.lblChannelNumber(i) = .ctrlAD(i).ChannelNumber
.txtChannelString(i).Enabled = True
AD_ChN(i) = .lblChannelNumber(i)
```

```
Else
AD_Enabled(i) = False
.txtChannelString(I) = ""
.lblChannelNumber(i) = 0
.txtChannelString(i).Enabled = False
AD_ChN(i) = 0
End If
```

また，ボードステータスを表示するランプでの判定のソースコードは以下のとおりである（８１頁　２４行目乃至４５行目）。

＜ボードステータスを表示するランプでの判定について原告が作成したソースコード＞

```
If AD_BoardFlag(i) And (.pbtmAD(i).Button = Button_Green Or .pbtmAD(i).Button =
Button_Yellow) Then
AD_Enabled(i) = True
.txtChannelString(i) = .ctrlAD(i).ChannelString
.lblChannelNumber(i) = .ctrlAD(i).ChannelNumber
.txtChannelString(i).Enabled = True
AD_ChN(I) = .lblChannelNumber(i)
Else
AD_Enabled(I) = False
.txtChannelString(i) = ""
.lblChannelNumber(i) = 0
.txtChannelString(i).Enabled = False
AD_ChN(i) = 0
End If
```

これらのソースコードにより，本件プログラム５は，「AD_BoardFlag」及びボードステータスを表示するランプによる判定という複雑な機能を実現しており，他のプログラマーが作成しても同様のソースコードにならないと考えられる。

また，上記の「For i = AD_LBound To AD_UBound」（甲３０号証　７８頁～８１頁）のループ文を用いたソースコードは，単純に４つのＡＤＣのループを行うものではなく，ＡＤＣの装備状態やスペックを判断し，可変的な装備に対応した命令の組み合わせとなっている（ソースコード）。通常の計測業務のプログラムでは，装備されるＡＤＣが予め固定されていることから，可変的な対応をすることはないが，本件プログラム５は開発環境で複数のＡＤＣを用いることも考えられることから可変的な対応をする必要があり，原告は独自の経験から上記の複雑なソースコードを作成している。そのため，「For i = AD_LBound To AD_UBound」（甲３０号証　７８頁～８１頁）のループ文を用いたソースコードも，他のプログラマーが作成しても同様のものにはならないと考えられる。

以上より，本件プログラム５には原告の個性が表れているといえる。

（４）　ＡＤＣの制御の方法について（別紙②）

ア　ソースコードの選択肢に幅があること

ＡＤＣを制御するソースコードの構成として，別紙②のとおり，「ＡＰＩ（アプリケーション・プログラミング・インターフェイス）によるボード制御」，「ＡｃｔｉｖＸによるボード制御」の選択肢が考えられる。ＡＰＩは，制御命令が個別の関数となっていることから，精細な制御に向いているという特徴があり，一方，

ＡｃｔｉｖＸは，制御命令がパック化されていることから，基本的な制御であれば簡単に実現できるようになっているという特徴がある。

ＡＰＩを選択する場合，ソースコードは，以下のような内容となる。

＜ＡＰＩを用いる場合のソースコード＞

```
'初期化を行いデバイスハンドルを取得します
Ret = AioOpenEx(TxtDevice, hDrv)
'A/D 変換を行います
Ret = AioInpBdMem(hDrv, InpMode, LpAInpMd4, MsgOK.hWnd,
MsgOK.Message, MsgERR.hWnd, MsgERR.Message)
'データの取得を行います
Ret = AioReadBuf(hDrv, LpARead)
```

また，ＡｃｔｉｖＸを用いる場合のサンプルのソースコードは以下のものとなる。

＜ＡｃｔｉｖＸを用いる場合のソースコード＞

```
'デバイスの初期化
Ret = AcxAiol.Open
'A/D 変換開始
Ret=AcxAiol.StartAI()
'--- データ取得 ---
AcxAiol.GetAIData ScanNumber, Buffer
```

原告は，本件プログラム５は，機能のテスト目的で作成される

ものであることから，詳細な制御は不要と判断し，ＡｃｔｉｖＸを採用した。その上で，目的にあったカスタマイズを行い，甲３０号証の７８頁乃至８３頁のソースコードを作成した。

以上より，ＡＤＣの制御の方法についても，選択肢が複数存在し，かつ目的に応じてカスタマイズされるソースコードも変化することから，ソースコードの組み合わせに選択肢の幅があることは明らかである。

イ　原告の個性が表れていること

ＡＤＣの制御に関するソースコード（甲３０号証　７８頁乃至８３頁）については，上記（３）で述べたソースコードと重なっており，原告の個性が表れていることは，上記（３）イのとおりである。

（５）　計測プロセス制御について（別紙③，④）

ア　ソースコードの選択肢に幅があること

本件プログラム５を開発プログラムのテスターとして利用するにあたって，開発プログラムのデータの確認作業を行う必要があるところ，データを計測する方法として「動的計測」，「静的計測」の方法が考えられる（別紙③）。

動的計測とは，振動などの変化の速い現象について，１秒間に多くのデータを測るものであり，静的計測とは，気温などの変化の遅い現象について，数分に１つのデータを測るものである。

本件プログラム５は，上述のとおり，様々な開発プログラムのテスターとして使用されることが予定されていたことから，原告は，動的計測と静的計測の両方を同時に実現できるよう甲３０号証の５乃至６頁，９乃至１０頁，６０頁５８行目乃至６１頁２３行目のソ

ースコードを作成した。

このうち甲３０号証の５頁乃至６頁のソースコードは，動的計測の測定データをファイルに出力するコードであり，８頁乃至１０頁目のソースコードは静的計測の測定スケジュールを管理するコードである。そして６０頁５８行目乃至６１頁２３行目のソースコードは，プロセスを制御するコードである。

したがって，原告が動的計測と静的計測の両方を同時に実現するソースコードを作成するという選択肢を採用しなければ，本件プログラム５は上記（甲３０号証の５乃至６頁，８乃至１０頁，６０頁５８行目乃至６１頁２３行目）のような組み合わせのソースコードにはならず，計測プロセス制御のソースコードの選択肢に幅があるといえる。

イ　原告の個性が表れていること

まず，上述した甲３０号証５乃至６頁のソースコードについては，出力処理を「First」と「Append」に分けて処理できるよう構成している。具体的には，「Public Function DynamicDataSave_First() As Boolean」のソースコード（甲３０号証　５頁２２行目）や「Public Function DynamicDataSave_Append(rData() As Integer) As Boolean」（甲３０号証　５頁５０行目）のソースコード等が挙げられる。

データ出力処理は，処理速度が速く簡単という理由から，測定が完了した時点で，一括で出力できるようにソースコードを組み合わせることが一般的であるが，本件プログラム５は，テスターであることから，様々なデータ量の測定に対応しなければならなかった。したがって，原告は，どのようなデータ量にも対応できるよう上記のように「First」と「Append」を組み合わせたソースコードを独自

に作成した。

したがって，甲３０号証５頁乃至６頁に記載されたソースコードは，他のプログラマーが作成しても同様のものになるとは考えれられず，原告の個性が表れている。

次に，甲３０号証の８乃至１０頁のソースコードについて，当該ソースコードは，上記のとおり静的計測の測定スケジュールを管理するものである。そして，原告は，当該ソースコードを「Cschedule クラス」と「Csetting クラス」のコードを用いて作成している。具体的には，「Private mvarSchedule(1 To 20) As New CSchedule」（甲３０号証９頁１８行目），「Global gSetInfo As New CSetting」（甲３０号証１頁７行目）等のソースコードが挙げられる。

一般的な静的計測では，１時間間隔での測定や１０分間隔での測定等一定の間隔で測定を行うように構成するため，ソースコードも時刻を対象に○○分経過すれば測定，分を１０で割り余りが０になれば測定するというような単純な判定で測定を開始できるような組み合わせとなる。

一方，本件プログラム５については，テスターであることから，例えば５分間隔で１２回測定した後に，１０分間隔で６回測定する等様々な測定スケジュールに対応して管理することが必要と考えられることから，原告は上記「Cschedule クラス」，「Csetting クラス」のソースコードを用いて，様々な測定スケジュールに対応できるソースコードを作成した。なお，原告は，最適化をポリシーとしていることから，判定をクラス内で行うようソースコードを構成している。

したがって，甲３０号証８頁乃至１０頁のソースコードよって，

様々な測定スケジュールの管理という複雑な処理を行うことが可能となっており，かつ原告は最適化の観点を重視して作成しているため，他のプログラマーが作成しても重視する点や経験等が異なり，同様の組み合わせになるとは考えられない。よって，原告の個性が表れている。

さらに，甲３０号証の６０頁５８行目乃至６１頁２３行目のソースコードは，計測プロセスを制御するコードである。具体的には，設定したスケジュールに該当するか，現在の測定モードは何なのか，変換データを転送する必要があるか等についての命令及びこれらの判断後の処理プロセスの呼出も記載されている。これらの判断は「g測定mode」の内容で判定されるため，Ｃａｓｅ文を用いることも考えられたが，原告は拡張時に「g測定Mode」要素以外を判断材料とする可能性があるため，ＩＦ　Ｅｌｓｅｌｆ文での分岐判断を使用し，「Elself g測定 Mode = 測定中 Then」（甲３０号証６１頁１５行目）等のソースコードを作成している。

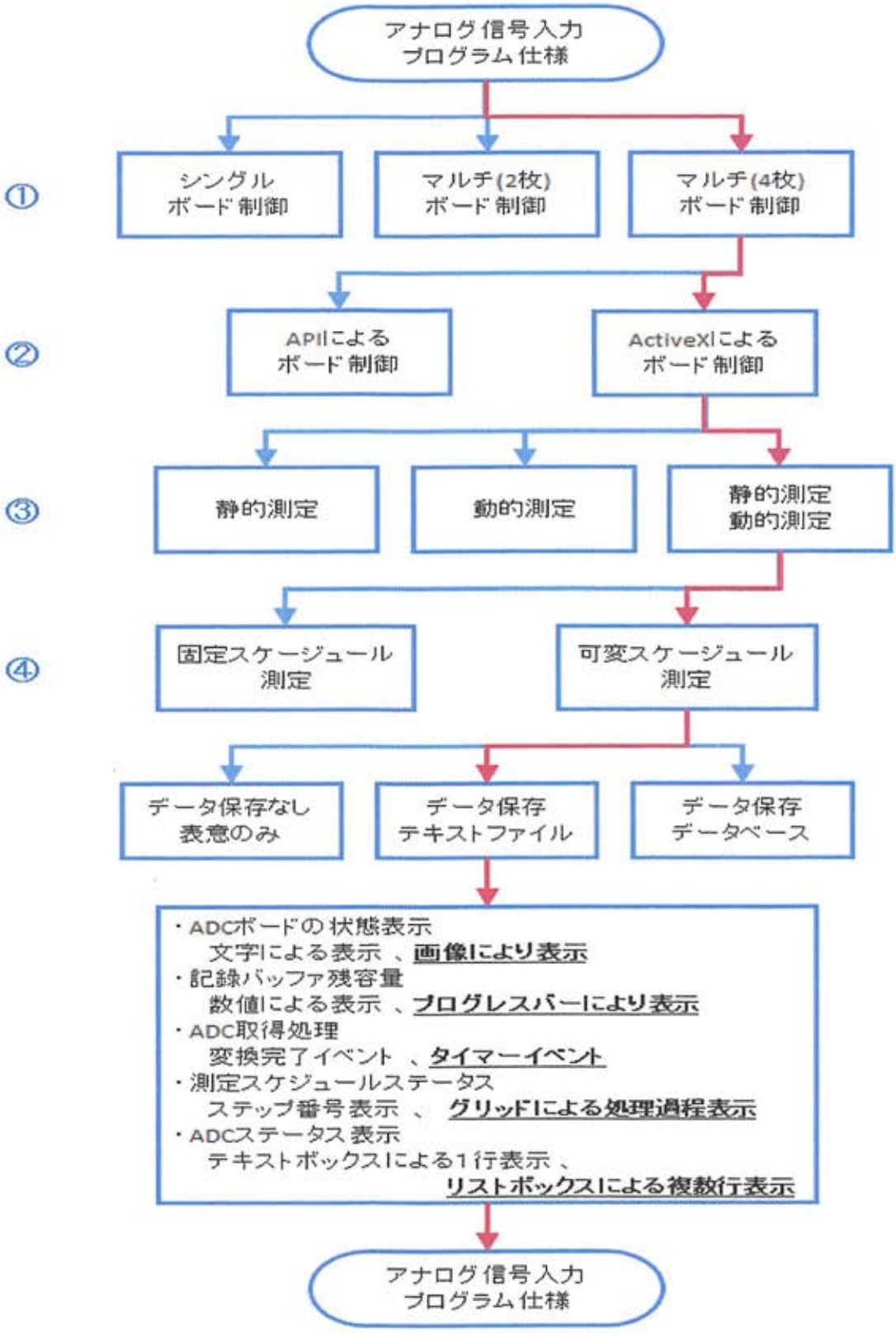
このような判断もプログラマー個人の経験や重視する点によってソースコードの記載が異なるところであり，他のプログラマーが作成した場合にも同様になるとは考えられず，原告の個性が表れている。

よって，計測プロセス制御のソースコードについても，原告の個性が表れている。

（６）　以上より，本件プログラム５の著作物性が認められる。

以　上

別紙


アナログ信号入力
プログラム仕様
①
シングル
ボード制御
マルチ(2枚)
ボード制御
マルチ(4枚)
ボード制御
②
APIによる
ボード制御
ActiveXによる
ボード制御
③
静的測定
動的測定
静的測定
動的測定
④
固定スケージュール
測定
可変スケージュール
測定
データ保存なし
表意のみ
データ保存
テキストファイル
データ保存
データベース
・ADCボードの状態表示
文字による表示 、画像により表示
・記録バッファ残容量
数値による表示 、プログレスバーにより表示
・ADC取得処理
変換完了イベント 、タイマーイベント
・測定スケジュールステータス
ステップ番号表示 、 グリッドによる処理過程表示
・ADCステータス表示
テキストボックスによる1行表示 、
リストボックスによる複数行表示
アナログ信号入力
プログラム仕様