平成１８年１２月１３日判決言渡　同日原本領収　裁判所書記官

平成１７年(ワ)第１２９３８号　不正競争行為差止等請求事件

口頭弁論終結日　平成１８年１０月１１日

判　　決

栃木県足利市（以下略）

原　　告　　栃木県トーションレース協同組合

訴訟代理人弁護士　　塚越敏夫

訴訟代理人弁理士　　工藤一郎

補佐人弁理士　　渡邉直幸

群馬県桐生市（以下略）

被　　告　　A

訴訟代理人弁護士　　白田佳充

同　　久保田寿栄

主　　文

１　請求第２項記載の訴えをいずれも却下する。

２　原告のその余の請求をいずれも棄却する。

３　訴訟費用は原告の負担とする。

事 実 及 び 理 由

第１　請求

１　被告は，別紙物件目録記載のソフトウェアを製造し，使用し，譲渡し(電気通信回線を通じた提供を含む。)，貸し渡し，又は譲渡若しくは貸渡しのために展示してはならない。

２(1)　被告は，上記１の侵害行為を組成した物(侵害の行為により生じた物を含む。)を廃棄し，かつ，侵害の行為に供した設備を除却せよ。

(2)　被告は，上記１の侵害の停止又は予防に必要な行為をせよ。

３　被告は，原告に対し，次の金員を支払え。

(1)　１３４９万３０００円及びこれに対する平成１５年１１月２０日から支払済みまで年５％の割合による金員，

(2)　１３０万円及びこれに対する平成１７年７月７日から支払済みまで年５％の割合による金員

第２　事案の概要

本件は，トーションレース(ゆるく縒った麻糸・綿糸で編んだ目の粗いレース)の編み機用ソフトウェアの著作権等を有する原告が，ソフトウェア開発会社の社員として原告のソフトウェアを開発した後，個人で第三者用に同様のソフトウェアを開発して譲渡した被告に対し，①原告の営業秘密であるアルゴリズムを使用したものであり，不正競争防止法２条１項７号の不正競争行為に当たる，②被告が勤務していたソフトウェア開発会社との信義則上の秘密保持義務に違反している，③原告の有するプログラム著作権(複製権又は翻案権)を侵害している，④原告が有する画面表示の著作権(複製権又は翻案権)を侵害していると主張して，不正競争防止法３条(上記①)，原告とソフトウェア開発会社との開発委託契約に基づく請求権を代位債権とする債権者代位権の行使(上記②)又は著作権法１１２条(上記③及び④)に基づき，ソフトウェアの製造差止め等を求めるとともに，不正競争防止法４条(上記①)又は著作権侵害の不法行為(上記③及び④)に基づく損害賠償を請求した事案である。

１　前提事実

(1)　当事者

ア　原告は，トーションレースの製造販売業者が設立した事業協同組合である。

イ　被告は，株式会社両毛システムズ(以下「両毛システムズ」という。)に勤務し，ソフトウェア開発を担当していたが，平成９年６月３０日，同社を退社した。(以上，争いのない事実，弁論の全趣旨)

(2)　原告ソフト開発の経緯

ア　原告は，平成８年８月，両毛システムズに対し，トーションレースの編み機を制御するためのソフトウェアの開発を委託した(以下，この契約を「本件開発

委託契約」という。)。

(争いのない事実)

イ　本件開発委託契約には，以下の定めがある。

「(総則)

第１条　甲(注：原告)は，トーションレースソフトウェア開発(以下対象物件といい，対象物件作成の業務を対象業務という)を乙(注：両毛システムズ)に委託し，乙は，これを受託し付則に定めた委託業務を履行するものとする。

(秘密の保持)

第９条　乙は，この契約の履行により知り得た委託業務の内容を第三者に漏らしてはならない。

(目的外使用の禁止)

第１０条　乙は，この契約の履行に必要な委託業務の内容を他の用途に使用してはならない。

(著作権の譲渡等)

第１１条　この契約により作成される対象物件(「多少物件」は，誤記と認める。)の著作権等の取り扱いは，次の各号に定めるところによる。

(1)　乙は，著作権法に基づく複製権，貸与権，翻訳権・翻案権等及び二次的著作物の利用に関する原著作者(「源著作者」は，誤記と認める。)の権利を，甲に無償で譲渡するものとする。

(2)　(略)」

(甲１)

ウ　被告は，両毛システムズにおいて，上記ソフトウェア開発に責任者として関与し，平成９年３月ころ，「トーションレースシステムVersion 2.0」(以下「原告ソフト」という。)の開発を完了した。

(争いのない事実)

(3)ア　原告ソフトは，プログラムの著作物(著作権法１０条１項９号)である。

イ　原告は，本件開発委託契約１１条(1)に基づき，原告ソフトの著作権を有する。

(以上，争いのない事実)

(4)　被告は，両毛システムズを退職後の平成１１年，トーションレースの編み機を製造販売している株式会社市川鉄工(以下「市川鉄工」という。)の委託により，その制御用ソフトウェアの開発に着手し，平成１２年，完成の上納品した(以下，そのソフトウェアを「市川ソフト」という。)。

(乙６，７，弁論の全趣旨)。

２　争点

(1)　営業秘密としてのアルゴリズムの不正使用

(2)　債権者代位権に基づく差止請求

(3)　プログラム著作権(複製権又は翻案権)の侵害

(4)　画面表示の著作権(複製権又は翻案権)の侵害

(5)　原告の損害

３　争点に関する当事者の主張

(1)　営業秘密としてのアルゴリズムの不正使用

ア　原告の主張

(ｱ)　原告アルゴリズムの内容

原告ソフトには，原告ソフトの主要機能である①糸の交差位置におけるテンション量に基づいて糸の移動軌跡を計算する処理，②計算した移動軌跡に基づいてトーションレースの図柄を表示する処理，③トーションレースの図柄のデータを編機用のデータに変換する処理などを具現化するために，別紙「アルゴリズムに関する当事者の主張」原告の主張のとおりのアルゴリズム(以下「原告アルゴリズム」という。)が使用されている。

(ｲ)　営業秘密該当性

ａ　秘密管理性

原告アルゴリズムに基づいて作成された原告ソフトのソースコード(以下「原告ソースコード」という。)は，原告の代表理事のみが管理し，利用希望の組合員に対しては，ソースコードではなく，原告ソフトのインストール用プログラムファイルが記録されているフロッピーディスクを貸与してインストールさせていた。

ｂ　有用性

原告アルゴリズムを具現化した原告ソフトは，糸の張力を考慮した実際の編み柄図を画面上に表示するシミュレーション機能により，試し編みという余分な工程を省略したり，組織図データの誤りなどをコンピュータ上で修正したりすることが可能となり，原告ないしその組合員の事業活動において，費用の節約，経営効率の改善等に役立つものである。

ｃ　非公知性

原告ソフトは，上記ａのとおり，原告の代表理事が管理し，一般に入手できないから，原告アルゴリズムも一般に知られていない。

(ｳ)　示された点

ａ　原告は，原告ソフトの開発に当たり，被告と綿密な打合せを行い，糸の太さや張力によってどのような図柄が実際に編み上がるかなど，トーションレースのシミュレーションを行う際の注意点を具体的に教示した(甲２１の１～２０)。

よって，原告は，被告に対し，原告アルゴリズムの基礎となる内容を示しており，営業秘密たる原告アルゴリズムを示したといえる。

ｂ　仮に，被告が独力で原告アルゴリズムを創作したとしても，その創作は職務としてされたものであり，原告アルゴリズムは本件開発委託契約９条及び１０条にいう委託業務の内容に含まれるから，被告は，営業秘密たる原告アルゴリズムを示されたといえる。

(ｴ)　原告アルゴリズムの使用

被告は，別紙「アルゴリズムに関する当事者の主張」原告の主張のとおり，原告アルゴリズムを使用して，市川ソフトのソースコード（以下「市川ソースコード」

という。）の作成のためにアルゴリズム(以下「市川アルゴリズム」という。)を作成し，それに基づき市川ソースコードを作成し，市川鉄工に譲渡した。

(ｵ)　図利加害目的

上記(ｴ)の使用は，原告ソフトと同一の機能を有する市川ソフトを利用して不正の利益を得る目的でされた。

(ｶ)　営業上の利益の侵害

上記(ｲ)ａのとおり，原告は，原告ソフトを組合員にのみ貸与していたところ，被告が同一の機能を有する市川ソフトを作成し，これが流通したことにより，原告の営業上の利益が侵害され，又は侵害されるおそれがある。

ｲ　被告の主張

(ｱ)　原告アルゴリズムの内容

被告の主張は，別紙「アルゴリズムに関する当事者の主張」被告の主張のとおりである。

(ｲ)　営業秘密該当性

原告の主張(ｲ)ａは不知，ｂは認め，ｃは否認する。

(ｳ)　示された点

ａ　原告の主張(ｳ)ａのうち，原告が原告ソフトの開発に当たり，被告と打合せを行ったことは認め，その余は否認する。

原告アルゴリズムは，被告が作成したものであり，原告から示されたものではない。原告が被告と行ったのは，原告ソフトの機能の打合せにすぎない。

ｂ　同ｂは否認する。

(ｴ)　原告アルゴリズムの使用

被告の主張は，別紙「アルゴリズムに関する当事者の主張」被告の主張のとおりである。

(ｵ)　図利加害目的

原告の主張(ｵ)は否認する。

(ｶ)　営業上の利益の侵害

原告の主張(ｶ)は否認する。

(2)　契約に基づく差止請求権の代位行使の可否

ア　原告の主張

(ｱ)　被告は，両毛システムズに対し，信義則上，同社に在職中知り得た秘密を保持する義務を負っている。

(ｲ)　原告は，両毛システムズに対し，本件開発委託契約に基づき，秘密保持ないし目的外使用の禁止請求権を有している。

(ｳ)　ソフトウェアの開発は，アルゴリズムの作成及びアルゴリズムに基づくソースコードの作成を含むから，本件開発委託契約９条の「知り得た委託業務の内容」には，原告アルゴリズムが含まれる。

(ｴ)　原告は，上記(ｲ)の請求権に基づき，債権者代位権の行使として，被告に対し，請求第１項及び第２項の請求をする。

イ　被告の主張

(ｱ)　原告の主張(ｱ)は否認する。

(ｲ)　原告の主張(ｲ)は否認する。

(ｳ)　原告の主張(ｳ)は否認する。

プログラムの開発業者が想到したプログラミングの手法や創作したアルゴリズムは，ソフトウェア開発における直接的な業務内容ではなく，本件開発委託契約９条の「知り得た委託業務の内容」に含まれない。

(ｴ)　原告の主張(ｴ)は争う。

(3)　プログラム著作権の侵害の成否

ア　原告の主張

(ｱ)　類似性

以下のとおり，市川ソースコードは，原告ソースコードに類似している。

ａ　データ構造の類似

(a)　原告ソフトにおいては，インスタンス変数(以下，単に「変数」ということがある。)であるm_ViewHGが，別紙「アルゴリズムに関する当事者の主張」図１の組織図を表わす１次元の配列を指している。この１次元の配列は，m_Spin*4*m_Maisu*2のバイト数の長さを有する(ここでは，長さなどの管理データの部分はないものとする。)。これにより，同図１０のように，各行に４バイトの領域がm_Spin個並び，そのような行がm_Maisu*2個からなる２次元の配列が表現される。

(b)　また，原告ソフトにおいては，変数であるm_TenshonHGが，糸の交点の座標を格納するための１次元の配列を指している。この１次元の配列は，m_ViewHGの指す１次元の配列と同じく，m_Spin*4*m_Maisu*2のバイト数の長さを有し，各行に４バイトの領域がm_Spin個並び，そのような行がm_Maisu*2個からなる２次元の配列が表現される。ただし，各行の４バイトの領域は，２バイトずつのshort型の２つの小領域から構成され，糸の交点のＸ座標，Ｙ座標の値がそれぞれの小領域に格納される。

(c)　一方，市川ソフトにおいては，SampleCreate()の記述から，変数m_SampleCurrentは，(m_CurrentSize.cx * 3 + 8) * course * 2のバイト数を有する１次元配列を指すと認められる。SamplePoint()の記述から，この１次元配列は，各行が，まず８バイトの領域の後に，３バイトを１単位とする領域がm_CurrentSize.cx個並び，そのような行がcourse * 2個からなる２次元の配列を表現している。

(d)　また，市川ソフトにおいては，変数m_SamplePointは，SamplePoint()の記述から，sizeof (CPoint) * m_SampleCurrentSize.cx * m_sampleCurrentSize.cy * 2のバイト数を有する１次元配列を指すと認められる。この１次元配列は，CPointクラスのインスタンスが m_SampleCurrentSize.cx 個並ぶ行が，m_sampleCurrentSize.cy * 2個からなる２次元配列を表わしていると認められる。各行に並ぶCPointクラスのインスタンスは，交点のＸ座標，Ｙ座標を格納するのに用いられている。

(e)　したがって，以下の対応関係が認められる。

原告ソフト	市川ソフト
m_ViewHG	m_SampleCurrent
m_TenshonHG	m_SamplePoint
m_Spin	m_SampleCurrentSize.cx
m_Maisu	course及びm_sampleCurrentSize.cy

(f)　上記箇所において，原告ソフトと市川ソフトは，いずれも，２次元の並びを１次元の配列というデータ構造で表現している。

(g)　相違点の第１として，原告ソフトでは，m_ViewHGが４バイトの領域の並びにより構成される２次元配列を表わすのに対し，市川ソフトでは，m_SampleCurrentの各行が８バイトの領域を持ち，３バイトを１単位とする領域の並びにより構成される２次元配列を表わしている。

相違点の第２としては，原告ソフトでは，m_TenshonHGがshort型の２つの値を持つ小領域の並びにより構成される２次元配列を表わすのに対し，市川ソフトでは，m_SamplePointがCPointクラスのインスタンスの並びにより構成される２次元配列を表わしている。

(h)　しかし，第１の相違点については，別紙「アルゴリズムに関する当事者の主張」図９に示すように，原告ソフトでは，m_ViewHGの指す配列を構成する各領域は３つの値を持つことから，４バイトの領域の並びから構成されるか，それとも３バイトを１単位とする領域の並びから構成されるかは，大きな違いとは認められない。

(i)　第２の相違点については，CPointクラスのインスタンスは，SamplePoint()における CPointクラスのコンストラクタの呼ばれ方からすると，２つの値を持つことから，short型の２つの値を持つ小領域か，それともCPointのクラスのインスタンスからなるかも大きな違いとは認められない。

ｂ　その他の類似点

(a)　その他の類似点は，別紙「アルゴリズムに関する当事者主張」原告の主

張２のとおりである。

(b) i　また，原告ソースコードでも，市川ソースコードでも，インスタンス変数には，すべて「m_」との接頭辞が付いている。

ii　また，市川ソースコードでは，SamplePointMove関数からSamplePointMove_Exec関数が呼ばれているが，原告ソースコードでも，TenshonDataFosei関数からTenshonDataFoseiExec関数が呼ばれている。

iii　このように，原告ソースコードと市川ソースコードとは，「m_」との接頭辞及び「Exec」との接尾辞の使い方において類似している。

(c) i　市川ソースコードのSamplePoint関数において，m_SampleCurrentは組織図を表わすデータを指していると考えられるが，原告ソースコードと同じようにバイト列である。

ii　原告ソースコードと市川ソースコードとは，この点でも類似している。

iii　被告は，データをバイト列に入れることは，一般的な手法であると反論するが，ここで問題にしているのは，プログラミング言語上の表現である。例えば，組織図は，２次元に広がる図形として認識され，これをプログラミング言語で表現する場合，素直に表現するのであれば，２次元配列を用いる。しかし，原告ソフトでは，m_ViewHG という１次元配列を用いており，市川ソフトでも，m_SampleCurrent という１次元配列を用いている。C++ 言語であれば，２次元配列よりも更に操作のしやすいデータ構造をプログラミング時に定義できるにもかかわらず，被告は，全くそのようなことをしていない。

(ｲ)　依拠

被告は，原告ソフトに依拠して，市川ソフトを作成した。

(ｳ)　故意又は過失

被告は，原告の著作権を侵害することにつき，故意又は過失があった。

(ｴ)　まとめ

以上のとおり，市川ソフトは，原告ソフトのプログラム著作権(複製権，翻案権)

を侵害する。

イ　被告の主張

(ｱ)　類似性

ａ　データ構造の類似

(a)　原告の主張(ｱ)ａ(a)は認める。

(b)　同(b)は認める。

(c)　同(c)は認める。

(d)　同(d)は認める。

(e)　同(e)は認める。

(f)　同(f)は認める。

(g)　同(g)は認める。

(h)　同(h)は否認する。

原告ソフトにおけるm_ViewHGの指す配列を構成する領域と市川ソフトにおけるm_SampleCurrentの指す配列を構成する領域は，両方とも組織図データを格納する領域で，糸の黒丸の情報，交差位置の右側の糸情報，交差位置の左側の糸情報の３種類のデータを取り扱うものであり，２次元の配列に関する３種類のデータを格納するだけの内容では，プログラムとして異なった記述をすることは難しい。その中で，原告ソフトでは４バイト，市川ソフトでは３バイトを使用していること自体が大きな違いである。

また，原告ソフトは，単にこの３種類のデータを管理しているのに対し，市川ソフトでは，先頭の８バイトの付加データに繰り返しの指示に使用するタグ，編み機のスピード，巻き取り停止，機械停止の４種類のデータを付加しているのであり，組織図を格納するデータ領域にも違いが認められる。

(i)　同(i)は否認する。

データを処理する場合，２次元配列を使用すると処理速度が遅くなるので，処理速度を上げるため１次元配列のように考えて処理することは，一般的なプログラミ

ングの手法であり，これをもってデータ構造が類似しているとはいえない。

ｂ　その他の類似点

(a)　原告の主張(ｱ)ｂ(a)に対する被告の認否は，別紙「アルゴリズムに関する当事者の主張」被告の主張２のとおりである。

(b)　同(b)のうち，ⅰ及びⅱは認め，ⅲは否認する。

「m_」との接頭辞及び「Exec」との接尾辞の使い方は，単に名前の付け方にすぎない。

(c)　同(c)のうち，ⅰは認め，ⅱは否認する。

データをバイト列に入れることは，プログラムを書く上で一般的な手法であるし，前記ａ(ⅰ)のとおり，処理速度を上げるために２次元の並びを１次元の配列というデータ構造で表現することも一般的な手法である。

(ｲ)　依拠

原告の主張(ｲ)は否認する。

両毛システムズにおいて原告ソフトを作成する際，原告との打合せ及びソフトの基本設計は，被告が行ったが，実際のプログラミングは，他の２名の従業員と分担し，被告は主にテンション図の作成を行った。そして，被告は，両毛システムズを退職する際，原告ソフト開発時に作成した資料等を全く持ち出していないし，現在その内容を再現することもできない。

(ｳ)　故意又は過失

原告の主張(ｳ)は否認する。

(ｴ)　まとめ

原告の主張(ｴ)は争う。

(4)　画面表示の著作権の侵害の成否

ア　原告の主張

(ｱ)　画面表示の内容

原告ソフトの画面表示(甲１０のＤ)では，縦横の升目状のシートが表示され，ス

ピンドルが交差し得る位置を表示する小さな黒斑点が中央部分に表示されている升目と，黒斑点がない升目とが市松模様状に整然と表示されている。

黒斑点の部分を市松模様状に表示することで，対応する糸掛図(甲１０のＢ)を表示する際に，糸の交差状態を整然かつ明瞭に表示させることができる。

また，黒斑点の升目にてスピンドルを交差させる処理を表示する場合には，楕円状の黒丸が表示される(甲１０のＣ)が，黒丸を表示すると黒斑点は視認できないため，この場合も整然とした様子を観者に与えることができる。

(ｲ)　創作性

上記(ｱ)の画面表示には，創意工夫による表現上の特徴があり，創作性がある。

(ｳ)　類似性

市川ソフトの画面表示(甲１１③，④)は，中央部に黒斑点がある升目とない升目とが市松模様状に表示されたり，黒斑点が黒丸で覆われる様子が表されており，原告ソフトの画面表示と酷似している。

(ｴ)　依拠

被告は，原告ソフトに依拠して，市川ソフトを作成した。

(ｵ)　故意又は過失

被告は，原告ソフトと同様の画面表示となることを知っていたものであり，故意又は過失があった。

ｲ　被告の主張

(ｱ)　原告の主張(ｱ)(画面表示の内容)は認める。

(ｲ)　原告の主張(ｲ)(創作性)は否認する。

原告の主張するシートや組織図の画面表示は，原告ソフトが作成される以前のトーションレース編み機用ソフトウェア，更に遡れば手書き時代に使用されていたのと同じものであり，創作性は認められない。

(ｳ)　原告の主張(ｳ)(類似性)は明らかに争わない。

(ｴ)　原告の主張(ｴ)(依拠性)及び(ｵ)(故意又は過失)は否認する。

(5)　原告の損害

ア　原告の主張

(ｱ)　原告は，被告の行為により，少なくとも原告ソフトの開発委託代金の支出が全く無意味になったため，その代金額１３４９万３０００円と同額の損害を被った。

(ｲ)　弁護士費用の損害として，１３０万円が相当である。

イ　被告の主張

原告の主張は，いずれも否認する。

第３　当裁判所の判断

１　争点(1)(営業秘密としてのアルゴリズムの不正使用)について

(1)　請求内容について

原告は，不正競争防止法２条１項７号，３条２項に基づき，侵害の停止又は予防に必要な措置として，請求第２項記載の行為を求めているが，その請求内容が特定されていないから，これらの訴えは，不適法なものとして却下されるべきである。

(2)　営業秘密の使用について

ア　被告が原告アルゴリズム全体をそのまま使用して市川ソフトを作成したのであれば，プログラムとしての表現が異なっていても，不正競争防止法２条１項７号の不正競争行為が成立する余地があるが，後記２(2)及び(3)に説示の事実によれば，被告は，市川ソフトを作成するに当たり，原告アルゴリズムをそのまま使用したものではなく，多くの点で原告アルゴリズムとは異なる処理手順を採用し，一部原告アルゴリズムと同様の処理手順を採用した箇所についても，技術上の合理性の観点から当然採用される部類に属する手法を採用したものであり，原告アルゴリズムや原告ソフトそのものを使用又は開示するに等しい結果を何ら招来していないものであるから，被告が市川ソフトを作成するに当たり，原告アルゴリズムを使用したものと認めることはできない。

イ　よって，原告の不正競争防止法２条１項７号，３条に基づき市川ソフトの

製造等の差止め及び侵害の停止又は予防に必要な措置を求める原告の請求(請求第１，第２項)，並びに同法４条に基づき損害賠償の支払を求める請求(請求第３項)は，理由がない。

２　争点(2)(債権者代位権に基づく差止請求)について

(1)　請求内容

原告は，債権者代位権に基づく差止請求として，請求第２項記載の行為を求めているが，その請求内容が特定されていないから，これらの訴えは，不適法なものとして却下されるべきである。

(2)　信義則上の秘密保持義務の内容

ア　被告は，前提事実(1)イ及び(2)ウのとおり，平成９年６月３０日まで両毛システムズに在職し，原告から委託を受けた原告ソフトの開発に責任者として関与したものであるから，信義則上，両毛システムズに在職中知り得た秘密を保持する義務を負っていると考えられる。

イ　被告がどのような内容の秘密保持義務を負うかについて検討する。

(ｱ)　被告がソフトウェア開発の前提として原告から開示された原告に特有のトーションレースの編み方のノウハウや，原告アルゴリズム全体をそのまま他に開示するような行為(原告アルゴリズムと全く同じアルゴリズムに基づくソフトを作成して市川鉄工に成果物として交付する行為を含む。)は，信義則上の秘密保持義務に違反すると考えられる。

(ｲ)　しかしながら，システムエンジニア又はプログラマーがあるソフトウェアの開発によって得たものは，一面で委託者から委託されたソフトウェア開発の成果物であるが，他面で，従来からシステムエンジニア又はプログラマーとして有していた技術を適用した結果であったり，技術上の合理性の観点から必然である処理手順であることが考えられ，これらの点を無視して信義則上の秘密保持義務を広く負わせることは，システムエンジニア等に同種のソフトウェアの開発に関与することを実際上禁止して職業選択の自由を制約し，社会経済的にも技術の蓄積によるソフ

トウェアの開発コストの削減を妨げる結果となりかねない。

以上の点からすると，以前に同種ソフトウェアの開発に関与した被告が信義則上の秘密保持義務に反したか否かは，原告アルゴリズムと市川アルゴリズムとが一致する割合はどの程度か，一致する部分について，当該システムエンジニア等が従来から有していた技術の適用の結果といえるか，又は技術上の合理性の観点からそのような手順を採用することが当然か，市川ソフトウェアやその前提となる市川アルゴリズムの一部が開示されることにより，従前の雇用主である両毛システムズ又は開発委託者である原告のノウハウ等が開示される結果となるか等を総合して判断するほかはない。

(3)　秘密保持義務違反の有無

ア　原告ソフトのTenshonDataSet関数(甲１５の１の２８３７行～２８８６行)が，別紙「アルゴリズムに関する当事者の主張」図４のフローチャート(以下「原告フローチャート」という。)に示される処理を行っていることは，当事者間に争いがない。

イ　そして，TenshonDataSet関数が，組織図からイメージ図を作成する処理の過程において，①移動後の交点の厳密な位置を求めるのではなく，各交点についてその上下に存在する交点位置の存在及び糸の張力を考慮して徐々に移動させることを繰り返しながら近似解を求めること，②糸の張力が大きいほど，その糸のＸ座標についてTenshonMove_1(Ｘ座標) がより多くの回数呼ばれること，③所定の回数だけ関数を呼ぶことで計算を打ち切っていること，さらに，これに対応する市川ソフトのSampleCreate関数が同様の処理を行っていることは，当事者間に争いがない。

ウ　原告は，その他のアルゴリズムも共通していると主張しており，その根拠として，①原告ソフトのSetTenshonHG関数と市川ソフトのSamplePoint関数，②原告ソフトのTenshonCenterMove関数と市川ソフトのSamplePointMove関数の類似点を指摘する。

その指摘内容自体は当事者間に争いがないが，いずれも，ソースコードのごく一

部について，表現が類似し又は機能が共通しているというものであって，市川ソフトが，上記アのフローチャートに示される処理と同一の処理を行っていることを裏付けるものではない。

エ　かえって，当事者間に争いのない事実に併せ，証拠(甲１５の１)及び弁論の全趣旨によれば，少なくとも以下の相違点が認められる。

(ｱ)　原告ソフトのTenshonDataSet関数と市川ソフトのSampleCreate関数

市川ソフトは，組織図データからサンプル図を作成する処理で，編む順番を記述したデータを使用し，ImageProgCouse関数で処理してから実際のサンプル図を作成する処理を行っているため，サンプル図が原告ソフトによる場合と異なる場合がある。

(ｲ)　原告ソフトのSetTenshonHG関数と市川ソフトのSamplePoint関数

これらの関数は，組織図の交点データ(黒丸)の位置を，組織図右上を原点とした相対的な距離で示す座標値に変換する処理を行う関数であるが，原告ソフトのSetTenshonHG関数(原告フローチャートのＳ４０２)には，組織図１つの升目のサイズを整数で行い，実際にサンプルを作成する大きさで計算するという考え方はないが，市川ソフトにおいては，実際にサンプルを作成する大きさで，１単位を０．１㎜として変換するという方法を採用している。

(ｳ)　原告ソフトのTenshonCenterMove関数と市川ソフトのSamplePointMove関数

これらの関数は，交点データを，交点の上下，斜め上，斜め下に別の交点が存在する場合にそれに応じて移動する処理を行う関数であるが，原告ソフトのTenshonCenterMove関数(原告フローチャートのＳ４０３)においては，糸の張力を考慮せず，組織図データと交点データを使用して移動処理を行っているが，市川ソフトにおいては，これらに併せ糸のテンション量を考慮して移動を行っている。

なお，原告ソフトにおいては，その後の処理過程において，TenshonYarnHG関数により糸のテンション値の設定を行い(原告フローチャートのＳ４０４)，張力に伴う交点移動の処理を行っている(原告フローチャートのＳ４０５)。

オ　以上によれば，結局，原告ソフトと市川ソフトに共通すると認められるアルゴリズムは，①組織図からイメージ図を作成する際，糸の張力に応じて糸の交点位置を移動すること，②その交点位置の移動を，上記イ①～③の手順で行うことであり，それ以外に共通するアルゴリズムがあるとは認められない。

カ　上記オ①について検討すると，証拠(乙１の１・２，２の１，３)及び弁論の全趣旨によれば，格子模様で糸の交点を黒丸とした組織図及びこれに糸を掛けた糸掛け図は，本件開発委託契約がされた平成８年８月の時点で，トーションレース業界において手書きの意匠図や編み機用プログラムに広く用いられていたと認められ，また，証拠(乙５)及び弁論の全趣旨によれば，トーションレースを実際に編み上げた場合に糸の張力が影響することは，昭和４０年当時のトーションレース編み機の説明書(乙５)にも記載されていることであって，イメージ図を作成する場合に，糸の張力に応じて糸の交点位置を移動させることは当然のことと認められる。

キ　上記オ②について，原告は，糸の交点の位置を連立方程式により求めることも可能であると主張するが，弁論の全趣旨によれば，多数の糸が無原則に交差している状態で１本の糸を引っ張った場合の移動を連立方程式で解くことは実際上不可能であると認められる。また，原告は，通常は「安定な条件」に到達したときに計算を打ち切ると主張するが，弁論の全趣旨によれば，そのような安定な条件を見出すことは困難であることが認められる。

そうすると，上記オ②の手法以外に採用が容易な手順があると認めることはできず，上記オ②の手法は，近似解により組織図からイメージ図を作成する方法として，技術上の合理性の観点から当然採用される部類に属する手法であると認められる。

ク　さらに，原告は，被告との打合せにおいて，原告アルゴリズムの基礎となる内容を教示したと主張するが，証拠(甲２１の１～２０)及び弁論の全趣旨によれば，原告は，被告との打合せにおいて，トーションレースの編み上げの実際を説明し，原告ソフトが有すべき機能の打合せをしたこと認められるものの，それ以上に，原告アルゴリズムを作成するにつき，有用な手順又はそのような手順に到るヒント

となる情報を提供したことを認めるに足りる証拠はない。

また，証拠(乙２の１～３)及び弁論の全趣旨によれば，トーションレースの編み上げの実際は，トーションレースの編み機の製造等を業とする市川鉄工からも容易に説明を受けられる情報であると認められる。

そして，弁論の全趣旨によれば，トーションレース編み上げ用のソフトウェアの開発に経験の長い被告が協力せず，他のシステムエンジニアが初めから同ソフトウェアの開発を行おうとすれば，トーションレース編み上げの基礎から理解をし，試行錯誤を行わなければならないため，その開発に時間を要することはうかがわれるが，それ以上に，被告が原告ソフトの開発を行ったことによって原告アルゴリズムや原告ソフトそのものが他に開示される結果となったとの事情は認められない。

ケ　これらの事実によれば，被告は，市川ソフトの作成に当たり，原告ソフトの開発に関与した経験が大いに役立ったが，原告アルゴリズムをそのまま使用したものではなく，多くの点で原告アルゴリズムとは異なる処理手順を採用し，一部原告アルゴリズムと同様の処理手順を採用した箇所についても，技術上の合理性の観点から当然採用される部類に属する手法を採用したものであり，原告アルゴリズムや原告ソフトそのものを開示するに等しい結果を何ら招来していないものである。

したがって，市川アルゴリズムに想到し，それに基づいて市川ソフトを作成したことをもって，被告が両毛システムズ退職後も負う信義則上の秘密保持義務に反したものと認めることはできない。

コ　よって，被代位債権が認められないから，債権者代位権に基づく原告の差止請求は，その余の点について判断するまでもなく理由がない。

３　争点(3)(プログラム著作権の侵害)

(1)　請求内容について

原告は，プログラム著作権の侵害があったとして，著作権法１１２条２項に基づき，侵害の停止又は予防に必要な措置として，請求第２項記載の行為を求めているが，その請求内容が特定されていないから，これらの訴えは，不適法なものとして

却下されるべきである。

(2)　複製権又は翻案権侵害について

ア　原告ソースコード(甲１５の１)とこれに対応する市川ソースコードを比較すると，TenshonDataSet関数に対応するSampleCreate関数，SetTenshonHG関数に対応するSamplePoint関数，TenshonCenterMove関数に対応するSamplePointMove関数，IchikawaWriteData関数に対応するWriteToWeave関数は，いずれもプログラムとしての表現が全く異なっていると認められる。

イ　これに対し，原告は，以下の類似点があると主張するが，以下に述べるとおり，いずれもソースコードの類似性を認めるに足りるものではない。

(ｱ)　データ構造の類似について

原告は，原告ソフトの変数m_ViewHGと市川ソフトの変数m_SampleCurrentが，それぞれ組織図データを格納していること，原告ソフトの変数m_TenshonHGと市川ソフトの変数m_SamplePointが，それぞれ交点座標のデータを格納していること，いずれも２次元配列を１次元の配列のデータ構造で表現していることから，原告ソフトと市川ソフトのデータ構造が類似していると主張する。

しかし，弁論の全趣旨によれば，両者は，トーションレース業界で広く用いられている組織図等に関するデータであるところ，組織図データは糸の黒丸の情報，交差位置の右側の糸情報，交差位置の左側の糸情報の３種類のデータであり，交点座標のデータはＸ座標，Ｙ座標の２種類のデータであって，いずれも単純な内容であること，データ１単位のバイト数，繰返しの指示に使用する付加データの有無，short型(２バイト)の２つの値を持つ小領域から構成されるかCPointクラスのインスタンスの並びにより構成されるかという点において違いがあることが認められるから，この点から両者のソースコードが類似していると認めることはできない。

(ｲ)　その他の類似点について

ａ　原告は，市川ソフトのSampleCreate関数(市川ソースコード１頁４３行～６４行)の記述において，TenshonDataSet関数と同じ処理がされていると主張する

が，SampleCreate関数全体の半分に満たない箇所において，表現はともかく共通した処理がされているにとどまる。

ｂ　原告は，全６５行に及ぶSamplePoint関数のうち，作業領域作成に関する７行の記述が実質的に同一であること，繰り返し行うfor文の２行が共通していること，ｘの値を実質的に２ずつ増やす，Ｙ座標を計算するために４や２で除すこと，CSizeで求めた値を変数に代入して関数が終了する点が共通すると主張するが，いずれも，ごく一部のソースコードの表現や処理内容が類似しているにとどまる。

ｃ　原告は，市川ソフトのSamplePointMove関数において，原告ソフトのTenshonCenterMove関数と同じ，Ｙ座標の値を「不連続に大きくする」処理が行われていると主張するが，弁論の全趣旨によれば，これは，組織図で表された柄を繰り返し編む際の組織図上下のつながりを考慮した処理をしているものであって，これが共通しているからといって，交点データを移動する処理を行う関数である上記関数全体が類似しているとはいえない。

ｄ　原告は，市川ソフトのWriteToWeave関数の１行「if (weave != 0) dat ¦= msk;」が原告ソフトのIchikawaWriteData関数と同じ処理を行っていると主張するが，これをもって上記関数全体が類似しているとはいえない。

ｅ　インスタンス変数や関数の名前の付け方及び２次元情報を１次元配列のデータ構造で表現する手法は，それ自体でソースコードの類似を認める根拠にはならない。

ウ　以上によれば，原告ソースコードと市川ソースコードとが類似していると認めることはできないから，プログラム著作権(複製権又は翻案権)に基づく原告の請求は，その余の点について判断するまでもなく，いずれも理由がない。

４　争点(4)(画面表示の著作権の侵害)

(1)　請求内容について

原告は，画面表示の著作権侵害があったとして，著作権法１１２条２項に基づき，侵害の停止又は予防に必要な措置として，請求第２項記載の行為を求めているが，

その請求内容が特定されていないから，これらの訴えは，不適法なものとして却下されるべきである。

(2)　複製権又は翻案権侵害について

ア　原告は，升目状のシートにスピンドルが交差し得る位置を表示する小さな黒斑点が中央部分に表示されている升目と，黒斑点がない升目とが市松模様状に整然と表示されている画面表示(甲１０のＤ)，及び黒斑点の升目にてスピンドルを交差させる処理を表示する場合に楕円状の黒丸を表示する画面表示(甲１０のＣ)につき，原告が著作権を有すると主張する。

イ　しかし，証拠(乙１の１・２，２の１，３)及び弁論の全趣旨によれば，上記画面表示は，コンピュータ化される以前から使われていた手書きシートをコンピュータ画面上に表示したにすぎないものであり，しかも，本件開発委託契約がされた平成８年８月以前に，これらを画面表示した編み機用プログラムが複数存在したことが認められる。

ウ　したがって，上記アの各画面表示に，原告の著作権を認めることはできないから，画面表示の著作権(複製権又は翻案権)に基づく原告の請求は，その余の点について判断するまでもなく，いずれも理由がない。

５　結論

以上のとおり，原告の訴えのうち請求第２項の訴えは，いずれも不適法であるから却下することとし(仮に適法であっても理由がないことは，前記に各説示のとおりである。)，その余はいずれも理由がないから棄却することとし，主文のとおり判決する。

東京地方裁判所民事第４０部

裁判長裁判官

市　　川　　正　　巳

裁判官

大　　竹　　優　　子

裁判官

頼　　晋　　一

（別紙）

物　件　目　録

株式会社市川鉄工製造のトーションレース編み機(Type　BLC２－６４及びBLC２－９６)に対応し，そのソフトウェアを装着すると，コンピュータのディスプレイ上で作成した組織図及び編み柄図どおりに，その編み機が自動的にトーションレースを編み上げてくれる，平成１２年に開発されたソフトウェア。

上記ソフトウェアの詳細は，以下のとおり。

トーションレースの編成に必要な組織図上のスピンドルの入れ替え位置と各スピンドルで使用される糸の情報とを記憶させ，前記スピンドルの入れ替え位置を画像表示させた後，該画像上で前記位置を修正移動させ，該移動後の確定した位置を繋いで各糸の軌跡を作成することを特徴とするトーションレースの編み上がりイメージ作成方法である。

このイメージ作成方法によれば，トーションレースの編成に必要な組織図上のスピンドルの入れ替え位置と各スピンドルで使用される糸の情報とを記憶させ，前記スピンドルの入れ替え位置を画像表示させた後，該画像上で前記位置を修正移動させ，該移動後の確定した位置を繋いで各糸の軌跡を作成することを特徴としているから，入力されたトーションレースの編成に必要な組織図に，各スピンドルで使用される糸の具体的情報が加味されて実際に糸が交差する位置を割り出されて編み上がりイメージが画像上で簡易に作成・確認できることから，実際にトーションレース機に掛けて試作編みを繰り返す必要がなくなる。

(別紙)

アルゴリズムに関する当事者の主張

第１　原告の主張

１　原告アルゴリズムの内容

(1)　TenshonDataSet関数のアルゴリズム

ア　TenshonDataSet関数(甲１５の１の２８３２行～２８８６行)は，CTorshonDocクラスのインスタンスが有するメソッドの１つであり，横軸方向に複数のスピンドルが配置され，横軸に平行に複数のコースが縦軸方向に配置されていることを前提として，複数のスピンドルを縦軸方向に移動させる際，スピンドルが交差する位置に黒丸を配置した組織図から，トーションレースの編み上がりのイメージ図を作成する。

イ　後記図１～図３は，組織図からイメージ図を作成する処理の過程の概略を示す。

組織図は，図１のように，格子が２次元に配置され，格子の形成する升目の中には，黒丸を含むものがある。この黒丸の位置において，スピンドルが交差することで糸が交差する。すなわち，横軸方向に配置されたスピンドルが図１の上から下の方向に向かって移動し，その際，黒丸の位置で隣接する２つのスピンドルが交差して糸が交差する。

その結果，図２のようにスピンドルの軌跡を形成する複数本の糸が上下方向の上から下に伸ばされ，図１の黒丸に相当する位置で糸が交差する図が得られる。

しかし，糸には張力がかかっているため，糸の交差位置は図２の位置から移動する。この移動後の糸の交差位置は，張力を考慮にいれた連立方程式により厳密に求めることも可能であるが，原告ソフトでは，各交点についてその上下(斜め上，斜め下を含む。)に存在する交点位置の存在及び糸の張力を考慮して徐々に移動させることを繰り返し，近似解を求めている。イメージ図(図３)は，その結果の図であ

る。

TenshonDataSet関数は，最終的に，図３に相当するビットマップ画像を生成する。

ウ　TenshonDataSet関数の処理の流れを示すフローチャートは，図４のとおりである。

(ｱ)　ステップＳ４０１においては，YarnDataSeiriView関数を呼ぶ。YarnDataSeiriView関数は，糸を流す処理を行う。「糸を流す」とは，組織図上で糸を上から下に伸ばし，スピンドルが交差する位置で隣接する糸を交差させることをいう。すなわち，図１の黒丸の位置でスピンドルが交差したように糸を交差させながら上から下に伸ばすことを意味する。後に説明するように，組織図に対応するデータ構造は，コース(升目の行)を単位に構成されており，コースのどの位置に何色の糸が通るかの情報が格納される。この情報の行間での整合性を取る処理がYarnDataSeiriView関数の行う処理である。

(ｲ)　ステップＳ４０２においては，SetTenshonHG関数を呼ぶ。SetTenshonHG関数は，交点データの作成の処理を行う。すなわち，図１の黒丸ごとの位置を求める。具体的には，左から右方向へ伸長するＸ軸と上から下方向へ伸長するＹ軸とのＸＹ座標による交点の座標位置を求め，その座標位置をその交点に対応する黒丸に関連付けることが行われる。

なお，原告ソフトにおいて，SetTenshonHG関数が計算する交点の位置は，升目の縦と横の長さをTCN_SIZEという整数の定数と考えて計算され，後のステップＳ４０６において現実の表示サイズに変換する。

(ｳ)　ステップＳ４０３においては，TenshonCenterMove(0)とTenshonDataFosei()を呼ぶ。これにより，交点位置をセンターに移動する処理が行われる。「センターに移動する」とは，交点の上と下(斜め上，斜め下を含んでもよい。)に別の交点が存在すれば，その別の交点の位置に応じて，交点の位置を移動することをいう。

なお，TenshonCenterMove(0)の0は引数の値であり，ステップＳ４０５では，TenshonCenterMove関数に引数として1を与えて，TenshonCenterMove(1)を呼ぶことが

行われる。

(ｴ)　ステップＳ４０４においては，SetTenshonYarnHG()を呼ぶ。これにより，各糸の張力(テンション)の値が所定の配列データに格納される。各糸の張力の値は，ソフトの中で定数値として定めることもできるが，原告ソフトでは，CTorshonAppクラスに動的に変更可能に格納された値を用いるようになっている。

(ｵ)　ステップＳ４０５においては，張力に伴う交点の移動の処理を行う。図６は，その処理のフローチャートを示す。

ａ　ステップＳ６０１において，retという変数に１を代入する。retという変数は，ステップＳ６０２からステップＳ６０６までのステップが形成するループを回るかどうかの制御を行うための変数である。

ｂ　ステップＳ６０２では，retの値が０でないかどうかを判断する。もし，retの値が０であれば，処理を終了し，そうでなければ，ステップＳ６０３へ処理を進める。

ｃ　ステップＳ６０３では，いったん，retに０を代入する。

ｄ　ステップＳ６０４では，Ｘ軸方向に糸の張力を検査する。糸の張力の値は，m_TenshonYarnの指すバイト列を参照することにより得ることができる。すなわち，例えば，Ｘ軸の小さい値から大きな値に向けて変数を大きくしながら，m_TenshonYarnのその変数の値番目の要素に格納されている値を取得する。もし，その値，すなわち糸の張力が０になっていなければ，TenshonMove_1(Ｘ座標)を３回呼ぶ。「TenshonMove_1(Ｘ座標)」とは，TenshonMove_1に現在着目しているＸ座標の値を引数として与えて呼ぶことを意味する。TenshonMove_1(Ｘ座標)を３回呼んだ後には，糸の張力を下げる。この処理は，m_TenshonYarnの指すバイト列に格納された値を小さくする(例えば，１を引く。)ことで行われる。次に，retに１を代入する。

ｅ　ステップＳ６０５では，retが０でないかどうかを判断し，０であれば，ステップＳ６０２へ戻る。また，０でなければ，ステップＳ６０６へ処理を進める。

ｆ　ステップＳ６０６では，TenshonCenterMove(0)を呼び，TenshonDataFosei

()を呼ぶ。

ｇ　すなわち，糸の張力が大きいほど，その糸のＸ座標についてTenshonMove_1(Ｘ座標)がより多くの回数呼ばれ，糸の張力の総和が大きいほど，TenshonCenterMove(0)とTenshonDataFosei()がより多くの回数呼ばれるようになっている。

(ｶ)　ステップＳ４０６においては，TenshonDataConv関数を呼ぶ。これにより，表示サイズに合うように座標位置の変換処理を行う。

(ｷ)　ステップＳ４０７において，ビットマップデータの作成の処理を行う。

エ　図５は，CTorshonDocクラスのインスタンスの持つ主なデータ構造を示す。

CTorshonDocクラスのインスタンスは，m_TenshonYarn，m_ViewHG，m_TenshonHGというインスタンス変数を持つ。これらのインスタンス変数は，バイト列を指すポインタであり，それぞれのバイト列は，スピンドル数(m_Spin)，m_Maisu*m_Spin*8，m_Maisu*m_Spin*8の値に等しいバイト数の長さを持つ(これらのバイト列は動的に確保され，長さなどの管理データを含んでいるが，ここでは管理データの部分が無いものとして説明する。)。

m_TenshonYarnの指すバイト列は，ステップＳ４０４でいうところの所定の配列データとなる。

m_ViewHBの指すバイト列は，主に図１に例示される組織図を表現するために用いられ，スロットに黒丸があるかどうか，何色の糸がどの位置に通るかを保持する。

m_TenshonHGの指すバイト列は，糸の交点位置をスロットの中の黒丸に関連付けるために用いられる。

オ　TenshonDataSet関数に呼び出される関数の概要は，以下のとおりである。

(ｱ)　図７は，YarnDataSeiriView関数が，糸を流す処理において，m_ViewHGの指すバイト列をどのように区切って扱うかを示す。

(ｲ)　図８は，YarnDataSeiriView関数が，糸を流す処理において，二つのポインタ変数としてsbuf1とsbuf2を用いるが，この二つのポインタ変数の位置関係を示す。

(ｳ)　図９は，YarnDataSeiriView関数が，糸を流す処理において，sbuf1の指すバイト列からの糸情報の読み出しの仕方，また，sbuf2の指す位置を基準とした糸情報の格納の仕方を示す。

(ｴ)　図１０は，SetTenshonHG関数が，交点データを作成する処理において，m_ViewHGの指すバイト列をどのように区切って扱うかを示す。

(ｵ)　図１１は，SetTenshonHG関数が，交点データを作成する処理の流れの概要を示す。

(ｶ)　図１２は，TenshonCenterMove関数が，交点位置を移動させる処理の前半の流れの概要を示す。

(ｷ)　図１３は，TenshonCenterMove関数が，交点位置を移動させる処理の後半の流れの概要を示す。

(ｸ)　図１４は，TenshonDataFosei関数が，TenshonDataFoseiExec関数にどのような引数を与えて呼出しを行うかを示す。

(ｹ)　図１５は，TenshonDataFoseiExec関数の処理の概要を示す。

(ｺ)　図１６は，TenshonMove_1関数の処理の概要を示す。

(2)　IchikawaWriteData関数のアルゴリズム

ア　IchikawaWriteData関数(甲１５の１の４４８１行～４５３４行)は，dataCourse，inBuf，otBuf，course，tag，Speedという６個の引数を持ち，otBufの指すバイト列に，他の引数の値を格納する処理を行う。

イ　具体的な処理は，以下のとおりである。

(ｱ)　courseの値を十進数で表わす文字列に変換してotBufに格納する。

(ｲ)　tagの値を文字列に変換してotBufに格納する。

(ｳ)　Speedの値を，少なくとも整数部分の桁数が１で小数点以下の桁数が３となる文字列に変換してotBufに格納する。

(ｴ)　inBuf［m_Wale*４＊２＊dataCourse+(m_ViewCourse-１)＊m_Spin*４］の値が０でなければ，文字ＹをotBufへ書き込み，０であれば，文字ＮをotBufへ書き

込む。ここに，m_Wale，m_ViewCourse，m_Spinは，CTorshonDocクラスのインスタンス変数である。

(ｵ)　otBufへ文字Ｎを書き込む。

(ｶ)　IchikawaWrite_mdatacnv関数を呼ぶ。

(ｷ)　IchikawaWrite_mdatacnv関数によって得られたデータであるmdataの後ろの要素から，その要素の値を十進数に変換してotBufに格納する。

(ｸ)　図１７は，IchikawaWrite_mdatacnv関数の処理の概要を示す。この関数は，buf，mdataを引数として取るが，bufの指すバイト列に格納された値が０かそうでないかを，ビットのｏｎ，ｏｆｆとしてmdataに格納する処理を行う。まず，mdataは１２バイトのバイト列であるので，全てのバイト位置に０を代入し，bitcnt，bytcntを０に初期設定し，変数ｌを０，１，２，…，m_Spin-１まで変化させながら，次の処理を行う。すなわち，buf［(l%２)＊m_Wale_*４］が０でなければ，mdata［bytcnt］のbitcnt目のビットを１にする。次に，bitcntを１だけ増やす。次に，ｌが７，１５，２３，…である場合に，bitcntの値を０として，bytcntを１だけ増やす。そして，bufの指す位置を４バイト右へずらす。なお，「ｌ％２」とは，ｌを２で割ったときの余りを表わす。

２　原告アルゴリズムと市川アルゴリズムの類似点

(1)　原告ソフトのTenshonDataSet関数と市川ソフトのSampleCreate関数

ア　原告ソフトのTenshonDataSet(甲１５の１の２８３７行以下)は，２８４８行～２８６９行において，①移動後の交点の厳密な位置を求めるのではなく，各交点についてその上下(斜め上，斜め下を含む。)に存在する交点位置の存在，また糸の張力を考慮して徐々に移動させることを繰り返しながら，近似解を求め，②糸の張力が大きければ大きいほど，その糸のＸ座標について TenshonMove_1(Ｘ座標)がより多くの回数呼ばれる，③繰り返し関数を呼ぶことによって近似解を求める場合，通常のように(例えば，微分方程式の解を数値的に近似的に求める場合などでは)，関数が呼ばれる前後での変化が所定の値以下になるなど安定な条件に到達し

たと判断されれば計算を打ち切るというのではなく，所定の回数だけ関数を呼ぶことで計算を打ち切っている。

これに対し，市川ソフトの SampleCreate関数(平成１８年２月２０日付被告第３準備書面添付のソースコード(以下「市川ソースコード」という。)１頁２行～２頁５行)の１頁の４３行「SamplePoint()」～６４行「SamplePointMove(3)」までの記述を検討すると，①SamplePointMove()，SamplePointPull() という関数を繰り返し呼ぶことで，交点の位置を徐々に移動させ，②SpinHGという変数は，m_SampleCurrentSize.cx の大きさを有する一次元配列を指していると思われるところ，SpinHG のＸ座標に相当する要素の値が大きければ大きいほど，SamplePointPull(Ｘ座標) がより多くの回数呼ばれており，③安定な条件に到達したと判断することなく，所定の回数だけ関数を呼ぶことで計算を打ち切っており，原告アルゴリズムと共通している。

イ　被告の主張に対する反論

(ｱ)　被告は，SampleCreate 関数は，組織図データをプログラムされたイメージ展開する処理を行うようになっていると主張する。

しかし，原告ソフトにおいても組織図データを表示する部分はあるのであり，編み上がりのイメージ図を作成する処理において，被告が説明するような「組織図データをプログラムされたイメージに展開する処理を行う」処理を更に実行することは，アルゴリズムの本質的な違いにならない。

また，具体的に，市川ソースコードのどの部分が該当するのかが不明であるが，「int course = ImageProgCourse();」の箇所が該当するのであれば，単に，ImageProgCourse() を，SampleCreate関数から呼ぶようにしただけであり，アルゴリズムの本質的違いとはならない。

(ｲ)　被告は，張力の強い方に糸が引っ張られ，交点が移動することは誰でも思いつくことであると主張するが，それは，本件委託契約当時の技術文献などを参照して判断するべきものである。

仮に，張力の強い方に糸が引っ張られ，交点が移動することが誰でも思いつくものであったとしても，交点をどのようなデータ構造でプログラム言語で表現し，張力の強さに応じてどのように移動させるのかは，直ちに思いつくものではなく，一義的に決まるものでもない。

(2)　原告ソフトのSetTenshonHG関数と市川ソフトのSamplePoint関数の対比

ア　原告ソフトにおけるm_TenshonHG(甲１５の１の２８８８行～)は，市川ソフトにおけるm_SamplePoint(市川ソースコード２頁６行～最終行) に対応するから，原告ソフトSetTenshonHG関数の記述の一部(甲１５の１の２８９３行～２８９９行は，市川ソフトの記述の一部(市川ソースコード２頁１３行「// 領域」～１９行「…bufSize);」)と実質的に同一である。

イ　原告ソフトにおけるm_Maisu は，市川ソフトにおけるm_SampleCurrentSize.cyに対応するから，原告ソフトの記述(甲１５の１の２９０９行)は，市川ソフトの記述(市川ソースコード２頁３６行「for … ; y++)[」)と同じであり，この y に関する for 文の中に，x に関する for 文(甲１５の１の２９２２行と市川ソースコード２頁５５行「for …; x++)[」)が共にあるのも同じである。

ウ　原告ソフトでは，x の値は２ずつ増えるのに対して，市川ソフトでは１ずつ増えるという違いはある。しかし，市川ソフトの上記記述とこれに続く同５６行「if ((x % 2 =…」によれば，y の値が固定されていれば，for文の内側のifの条件は，x が偶数か奇数の場合のどちらかでしか真にならないので，この点から原告ソフトと市川ソフトとは実質同一である。

エ　原告ソフトでは，交点の位置のＹ座標を計算するために，４で除したり(甲１５の１の２９１４行～２９２０行)，２で除したり(甲１５の１の２９２３行～２９２６行)しているが，市川ソフトでも(市川ソースコード２頁４０行「if (stop)[」～５０行「…= ky / 2;」)，Ｙ座標を計算するために，４で除したり２で除したりしている。

オ　原告ソフト(甲１５の１の２９３６行～２９３８行)と市川ソフト(市川ソー

スコード２頁下から３行の「m_SampleSize = …」から下から２行「return TRUE;」まで)は，いずれも，CSize で求めた値を変数に代入して関数が終了している。

カ　被告の主張に対する反論

上記１(1)ウ(イ)のとおり，原告ソフトのSetTenshonHG関数では，TCN_SIZE という整数の定数を升目の大きさとし，整数値で交点の座標を計算しているが，必要に応じて TCN_SIZE の値を 512 や 1024 など大きな値に変更することにより，任意の精度で交点の座標を求めることができ，また，０．１mmをどれだけの整数値に対応させるのかも任意のことである。また,「実際にサンプルを作成する大きさ」で計算することと，仮の値で計算した後に縮小ないし拡大をすることとは，アルゴリズムの本質的な違いにならない。

(3)　原告ソフトのTenshonCenterMove関数と市川ソフトのSamplePointMove関数の対比

ア　原告ソフトTenshonCenterMove()(甲１５の１の２９４２行～)において，着目している交点の右上の交点を探索する記述(甲１５の１の２９８１行～２９９６行)の部分においては，着目している交点のＹ座標の値よりも小さいＹ座標の値の交点を探索すれば十分なはずであるが(図１４参照)，２９９２行で 「y1--; 」によりＹ座標の値を小さくした後，２９９３行，２９９４行で「 y1 」の値が負になると,「y1 」に「 m_Maisu * 2 」を加えてＹ座標の値を大きくして(このことを,「不連続に大きくする」という。)，着目している交点より右下にある交点を探索している。

これに対応する市川ソフトの部分(市川ソースコード３頁１行～)は，SamplePointMove() から呼ばれる関数SamplePointMove_Exec() から更に呼ばれる関数SamplePointMove_RighTop() と思われるところ，その記述(市川ソースコード５頁４９行「CPoint CTochonDoc ::…」～５３行「….cy * 2;」及び６８行「y--;」～６９行「….cy * 2;」)は，Ｙ座標の値を格納する変数 sy 及び y から１を引き，結果が負になれば，y に m_Maisu * 2 に相当する m_SampleCurrentsize.cy * 2 を代入

し，y の値を不連続に大きくしている。

イ　Ｙ座標の値を不連続に変化させる同様の記述は，原告ソフトにおいては，着目している交点の左上，右下，左下の交点の探索の処理にも存在する。これに対応する市川ソフトの処理の記述は， SamplePointMove_LeftTop()，SamplePointMove_RigthBottom()，SamplePointMove_LeftBottom() の関数の記述の中にあると思われるが，Ｙ座標の値を不連続に大きくしたり小さくしたりしている処理の記述が見られる(原告ソフトでは，右下，左下の交点の探索においてＹ座標の値が m_Maisu*2 を超えると，Ｙ座標の値から m_Maisu*2 を減じて不連続に小さくしており，一方，市川ソフトでは，Ｙ座標の値が m_SampleCurrentSize.cy * 2 を超えると，Ｙ座標の値を１にして不連続に小さくしており，結果として原告ソフトと市川ソフトとでは，同じように不連続に変化する。)。

ウ　被告の主張に対する反論

(ｱ)　確かに，原告ソフトのTenshonCenterMove関数では，糸の張力を考慮していないが，原告ソフトでは，図６のとおり，糸のテンション量を考慮して交点位置を変更する処理が行われている。したがって，そもそもSamplePointMove関数をTenshonCenterMove関数に対応付けるのは正しくない。

(ｲ)　また，被告は，SamplePointMove関数において，lpYarnという配列を参照していることをもって，市川ソフトでは糸の張力を考慮して交点を計算していると反論していると思われるが，市川ソフトの SamplePointMove 関数において，引数 sign の値が１以下であれば，lpYarn の配列のうち，lpYarn[yn1].yaYarnWidth，lpYarn[yn2].yaYarnWidth の値しか参照されず，「yaYarnWidth」という名称からは，これらは張力を示すとは思えない。むしろ，「yaSpring」という名称から張力を示すと思われるのは， lpYarn[yn1].yaSpring，lpYarn[yn2].yaSpring であろう。しかし，これらは，引数 sign の値が１より大となっているときにのみ参照されないことになっている。市川ソフトでの SampleCreate 関数での SamplePointMove 関数の呼び方を見ると，SampleCreate 関数の後半でのみ SamplePointMove(2)，Samp

lePointMove(3) が呼ばれている。以上からすれば，交点の計算の際に張力が考慮されるのは，SampleCreate 関数の後半のみである。

一方，SampleCreate関数に対応する原告ソフトの TenshonDataSet関数でも，糸の張力を考慮して交点を移動させるのは，後半部分である２８５３行目から２８６９行目である。したがって，原告ソフトと市川ソフトとでは，張力を考慮に入れて交点を移動させるアルゴリズムは，同じ又は類似している。

(4)　原告ソフトのIchikawaWriteData関数と市川ソフトのWriteToWeave関数

ア　市川ソフトのWriteToWeave関数(市川ソースコード８頁１０行～最終行)中の「if (weave != 0) dat ¦= msk;」は，原告ソフトのIchikawaWriteData関数(甲１５の１の４４８１行～４５３４行)「ビットのｏｎ，ｏｆｆとしてmdataに格納する処理」(別紙図１７参照)に相当する。

イ　被告の主張に対する反論

被告は，「市川ソフトは，バイナリーデータを全て別のブロックに分けて出力する」点で異なると主張するが，市川ソフトのWriteToWeave関数では，m_Bufというインスタンス変数と思われる変数が指しているバイト列にデータが書き込まれているので，m_Bufは原告ソフトでのotBufに相当するので，被告の上記主張は，誤りである。

また，被告は，テキストデータを作成する点とバイナリデータを作成する点とで違いがあると主張するが，テキストデータを生成するのか，バイナリデータを生成すのかは，アルゴリズムの違いを生じさせるものではない。

(5)　まとめ

以上のように，被告が原告ソフトのアルゴリズムをそのまま流用し，市川ソフトを作成したことは，明らかである。

第２　被告の主張

１　原告アルゴリズムの内容

原告の主張する内容は，いずれもプログラムの説明にすぎず，具体的にサンプル

図を作成するための処理手順として営業秘密となっている部分を示していない。組織図データからサンプル図を作成する処理は，いくつかの工程を踏みながらデータを加工し，実際のサンプルに似せる技術であり，方程式を利用してできるものではない。重要なのは，各工程の具体的内容の構成及び全体の流れである。

２　原告アルゴリズムと市川アルゴリズムの類似点

(1)　原告ソフトのTenshonDataSet関数と市川ソフトのSampleCreate関数

ア　原告の主張２(1)は明らかに争わないが，市川ソフトが同じアルゴリズムを使用していることは否認する。

イ　原告の主張するアルゴリズムが，単に糸の交点を求め，糸の張力に応じて交点位置を移動するという次元のものであるなら，営業秘密には当たらない。従前，組織図として，交点位置に丸穴を書く方法と，糸が通る位置を有線で表す糸掛けという方法があったところ，糸の張力を考慮し，張力の強い方に糸が引っ張られて交点が移動するテンション図を作成することは，誰でも思いつくことである。

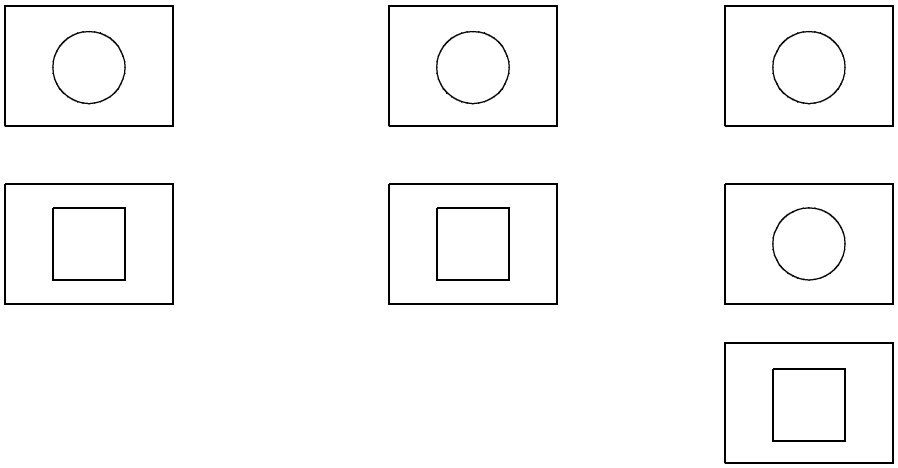
ウ　原告は，①糸の交点の位置を連立方程式により求めることも可能であると主張するが，多数の糸がランダムに交差している状態で１本の糸を引っ張った場合の移動を公式とするのは不可能である。また，②通常は「安定な条件」に到達したときに計算を打ち切ると主張するが，そのような安定な条件を見出すことは不可能である。そのような不可能な方法をとっていないことをもって，共通点とすべきではない。

エ　上記関数は，いずれも，サンプル図を作成するメイン関数であるが，市川ソフトにおいては，交点データ(組織図の黒丸の位置を表すデータ)を作成する前に，組織図データをプログラムされたイメージに展開する処理を１段階で行っており，２段階を要する原告アルゴリズムとは異なる。

原告は，「単に，ImageProgCourse()を，SampleCreate関数から呼ぶようにしただけであり，アルゴリズムの本質的違いとはならない」と主張するが，市川ソフトは，組織図データからサンプル図を作成する処理で，編む順番を記述したデータを

使用し，ImageProgCouse関数で処理してから実際のサンプル図を作成する処理を行っている。その結果，例えば，下記の組織図データの○の部分を２回編んでから□の部分を１回編むという指示を行った場合，原告ソフトのサンプル図と市川ソフトのサンプル図は，下記のように異なる。

組織図データ　　原告ソフトのサンプル図　　市川ソフトのサンプル図



(2)　原告ソフトのSetTenshonHG関数と市川ソフトのSamplePoint関数

ア　原告の主張２(2)は明らかに争わないが，市川ソフトが同じアルゴリズムを使用していることは否認する。

イ　原告は，作業領域を作成する方法，繰り返し行うfor文の記述等，ソースコードの一部が似ていると主張しているだけであって，アルゴリズムとは無関係である。

なお，Ｙ軸を計算するために４で除したり２で除したりしているのは，Ｙ軸方向に紋紙１枚分で表される長さを除しているだけで，交点位置を計算するアルゴリズムとは関係がない。

ウ　上記関数は，組織図の黒丸の位置を，組織図右上を原点とした相対的な距離で示す座標値に変換する処理を行う関数であるが，原告ソフトにおいては，組織

図１つの升目のサイズをＸ方向８，Ｙ方向８として変換しており，実際にサンプルを作成する大きさで計算するという考え方はないが，市川ソフトにおいては，実際にサンプルを作成する大きさで，１単位を０．１㎜として変換するという格段に正確な方法を採用している。

仮の値で計算した後で縮小・拡大する方法と実際のサイズで計算することは，同じ結果とはならない。

(3)　原告ソフトのTenshonCenterMove関数と市川ソフトのSamplePointMove関数

ア　原告の主張２(3)は明らかに争わないが，市川ソフトが同じアルゴリズムを使用していることは否認する。

イ　原告が主張する「不連続に大きくする」との内容は，トーションレースを編む場合，組織図で表された柄を繰り返し編んで製品を作るという構造のため，組織図の上下のつながりを考慮して処理する必要があるためであって，アルゴリズムとは無関係である。

ウ　交点データを移動する処理を行う関数であるが，原告ソフトにおいては，糸の張力を考慮せず，組織図データと交点データを使用して移動処理を行っているが，市川ソフトにおいては，これらに併せ糸のテンション量を考慮して移動を行っており，より簡素化され，進歩している。

(4)　原告ソフトのIchikawaWriteData関数と市川ソフトのWriteToWeave関数

ア　原告の主張２(4)は明らかに争わないが，市川ソフトが同じアルゴリズムを使用していることは否認する。

イ　いずれも，市川の編み機データ作成処理を行う関数であるが，原告ソフトと市川ソフトでは，作成する編み機データの形式が異なる。すなわち，原告ソフトは，１レコード内にタグ，スピード，巻き取り等をテキストデータである組織図データと一緒に出力するが，市川ソフトは，バイナリーデータをすべて別のブロックに分けて出力する点で異なる。

(5)　まとめ

原告の主張２(5)は争う。

以　上

図1

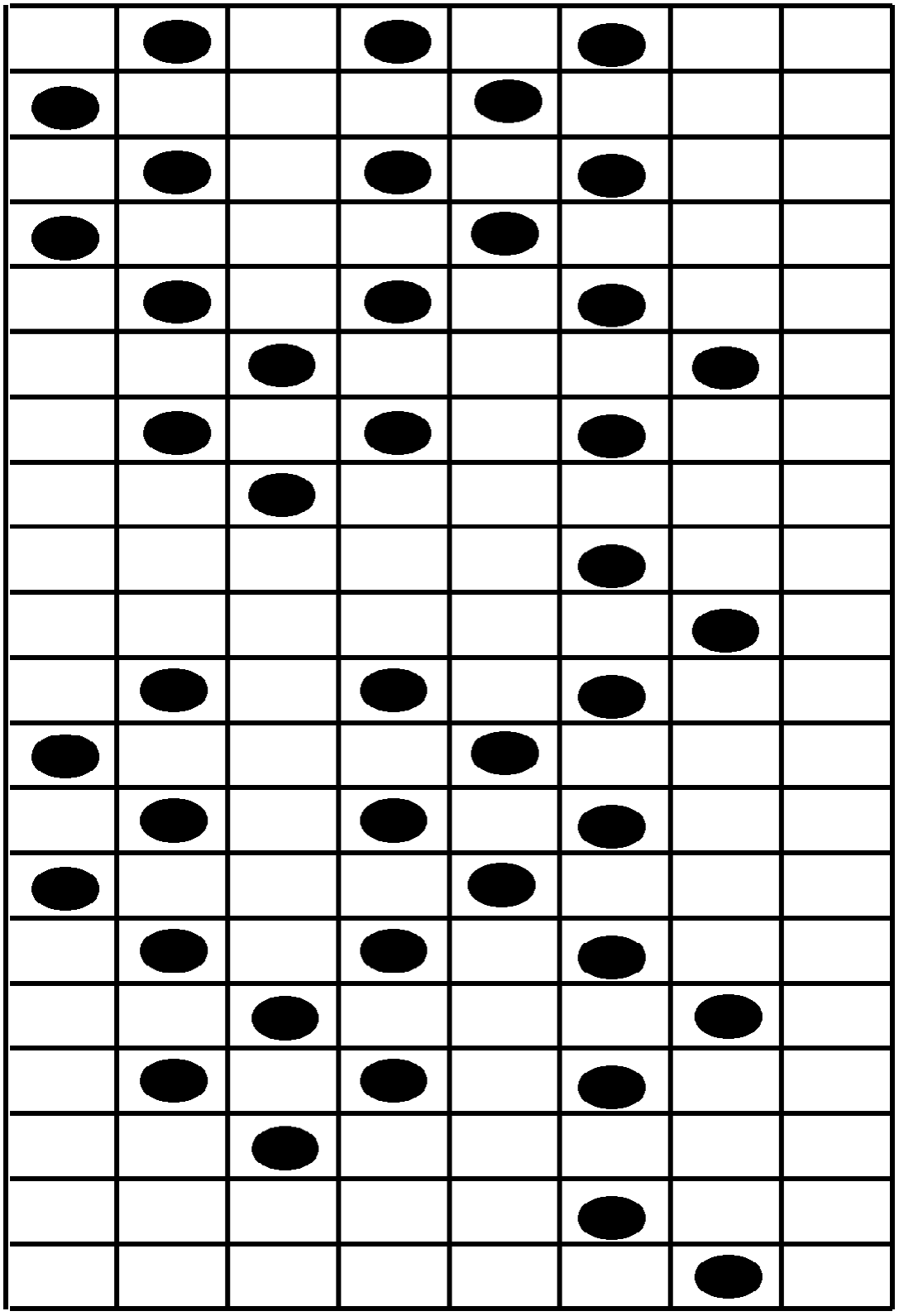


図2

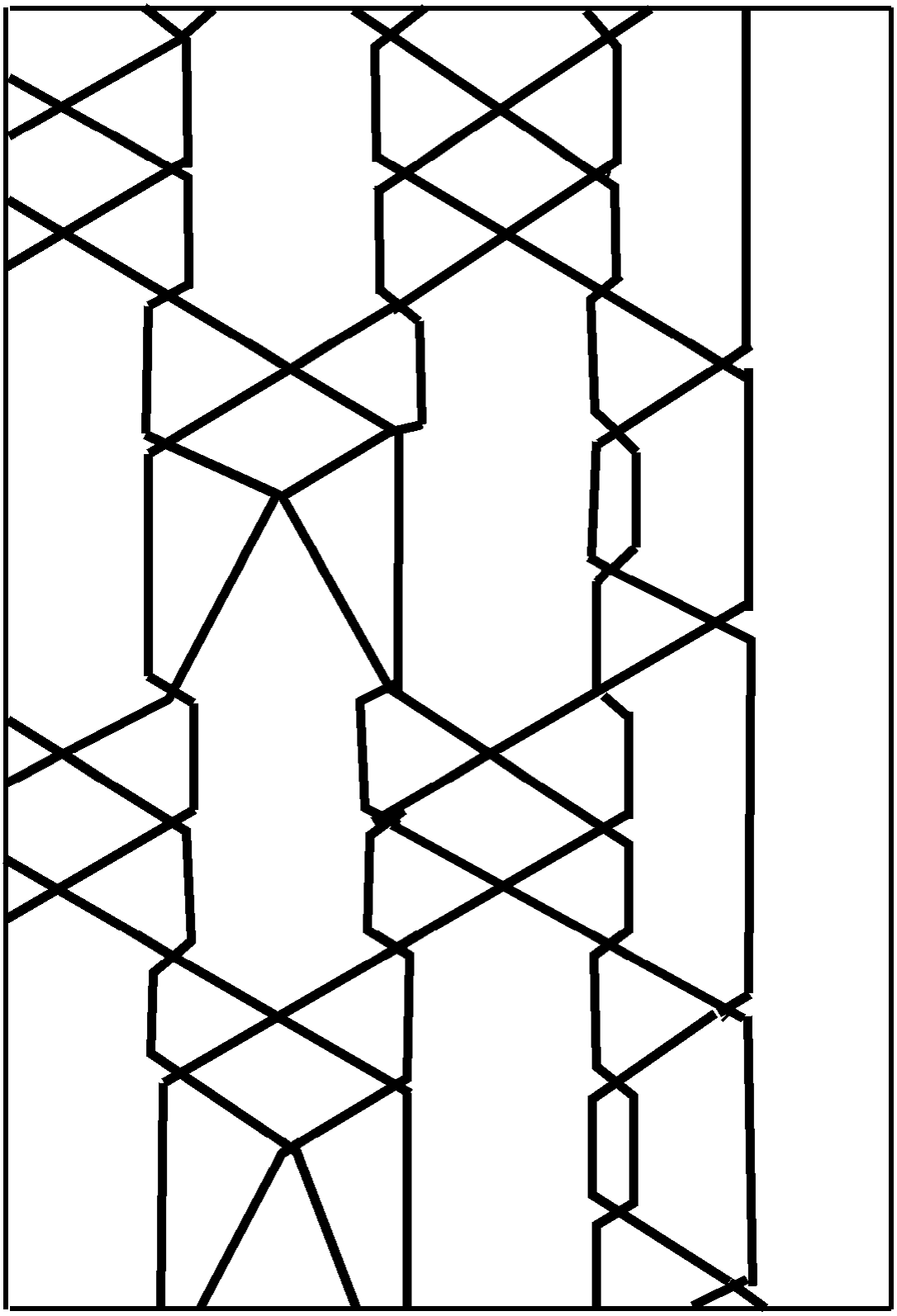


図3

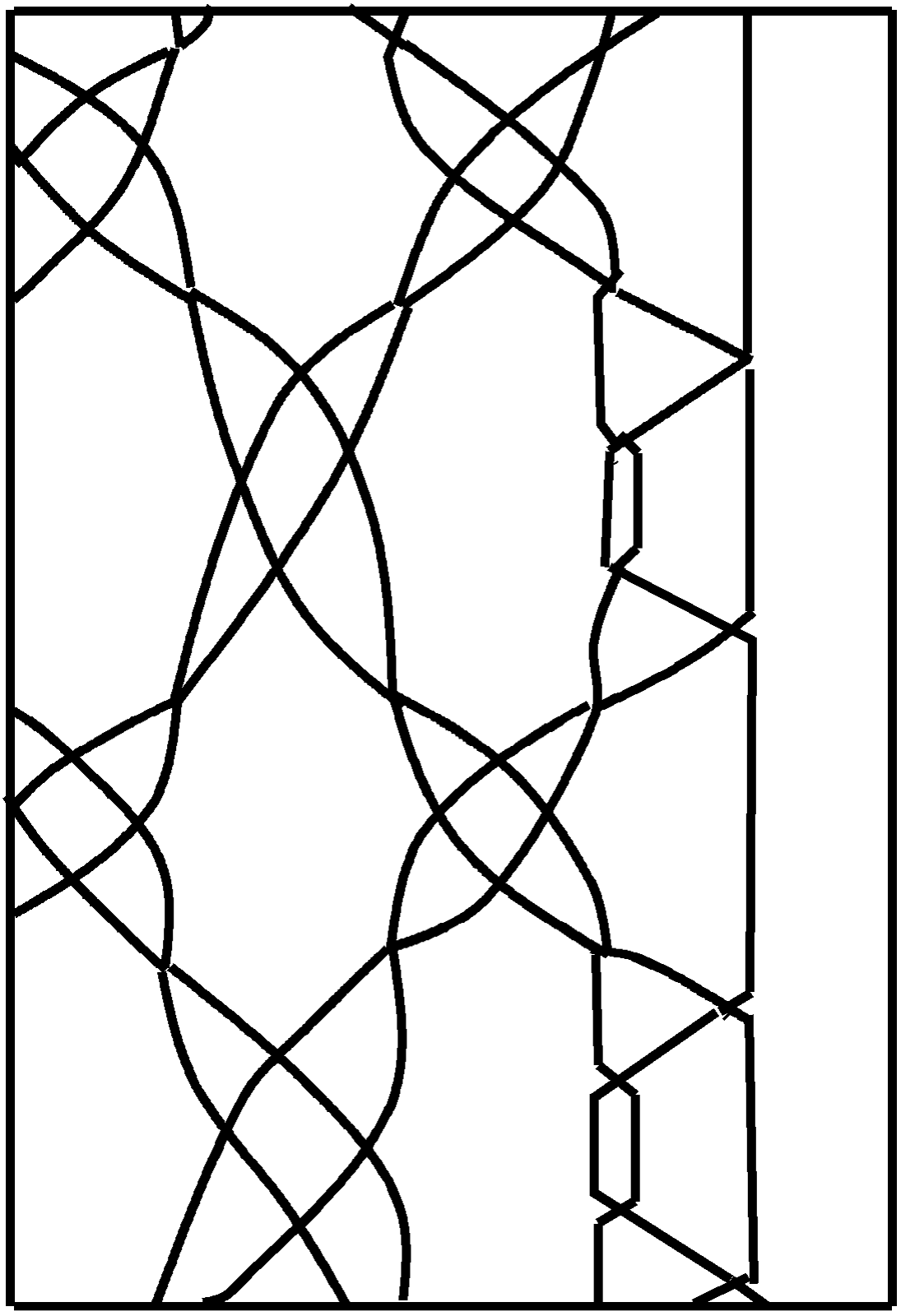


図4

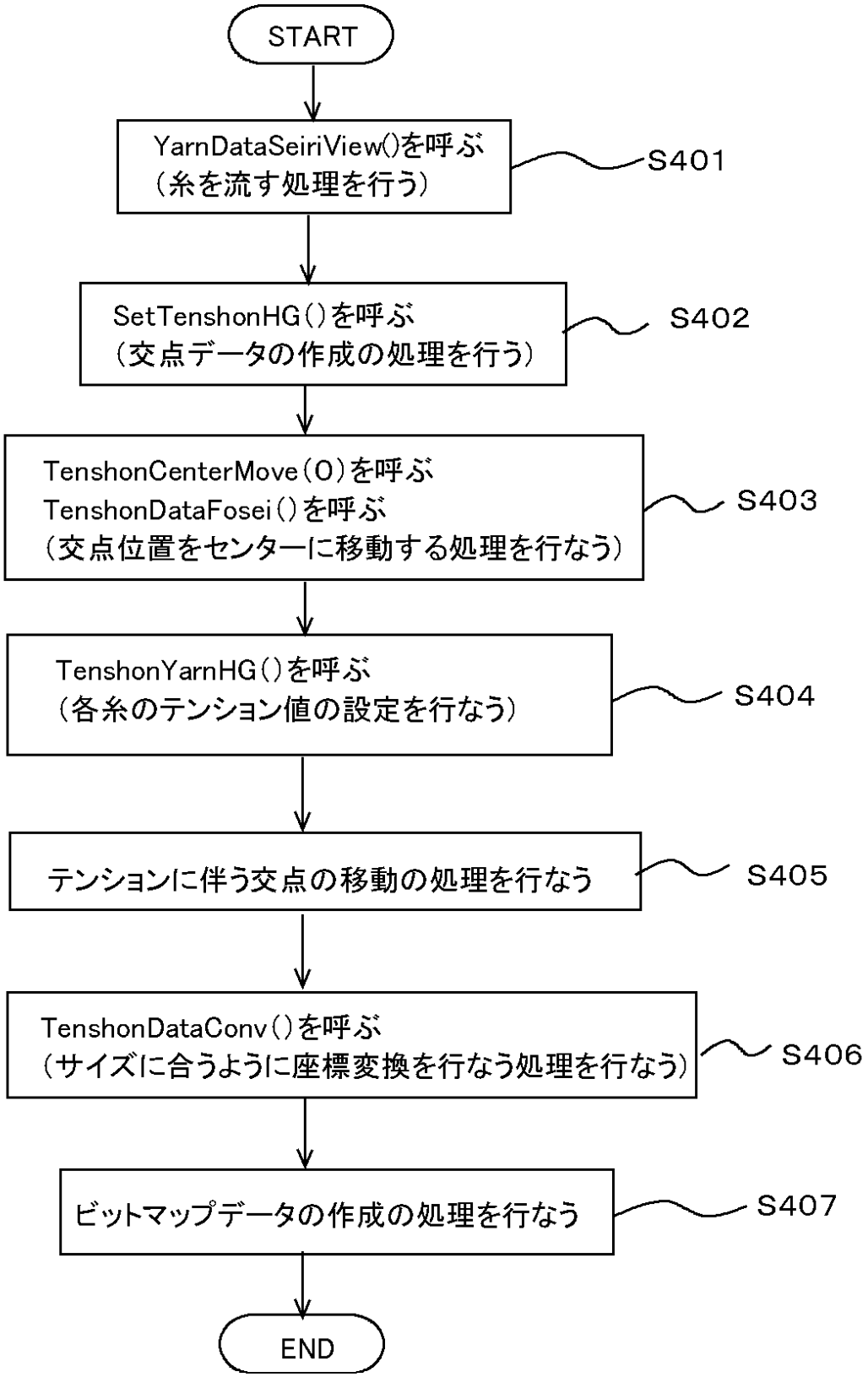

START
YarnDataSeiriView()を呼ぶ
（糸を流す処理を行う）
S401
SetTenshonHG()を呼ぶ
（交点データの作成の処理を行う）
S402
TenshonCenterMove(0)を呼ぶ
TenshonDataFosei()を呼ぶ
（交点位置をセンターに移動する処理を行なう）
S403
TenshonYarnHG()を呼ぶ
（各糸のテンション値の設定を行なう）
S404
テンションに伴う交点の移動の処理を行なう
S405
TenshonDataConv()を呼ぶ
（サイズに合うように座標変換を行なう処理を行なう）
S406
ビットマップデータの作成の処理を行なう
S407
END

図5

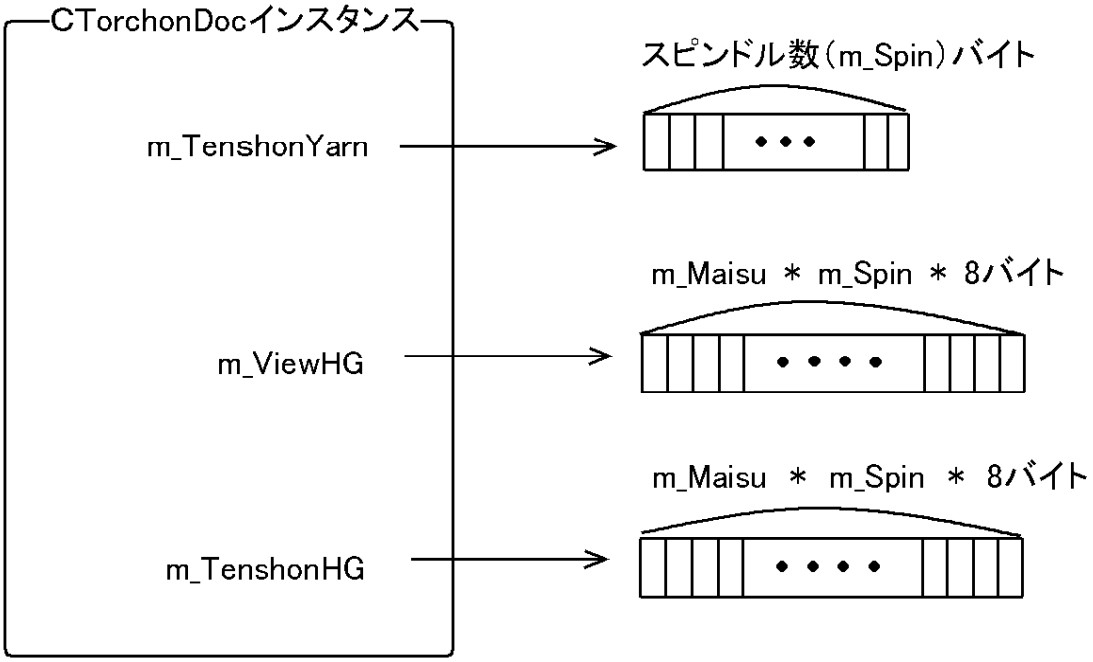

CTorchonDocインスタンス
m_TenshonYarn
スピンドル数（m_Spin）バイト
m_ViewHG
m_Maisu ＊ m_Spin ＊ 8バイト
m_TenshonHG
m_Maisu ＊ m_Spin ＊ 8バイト

図6

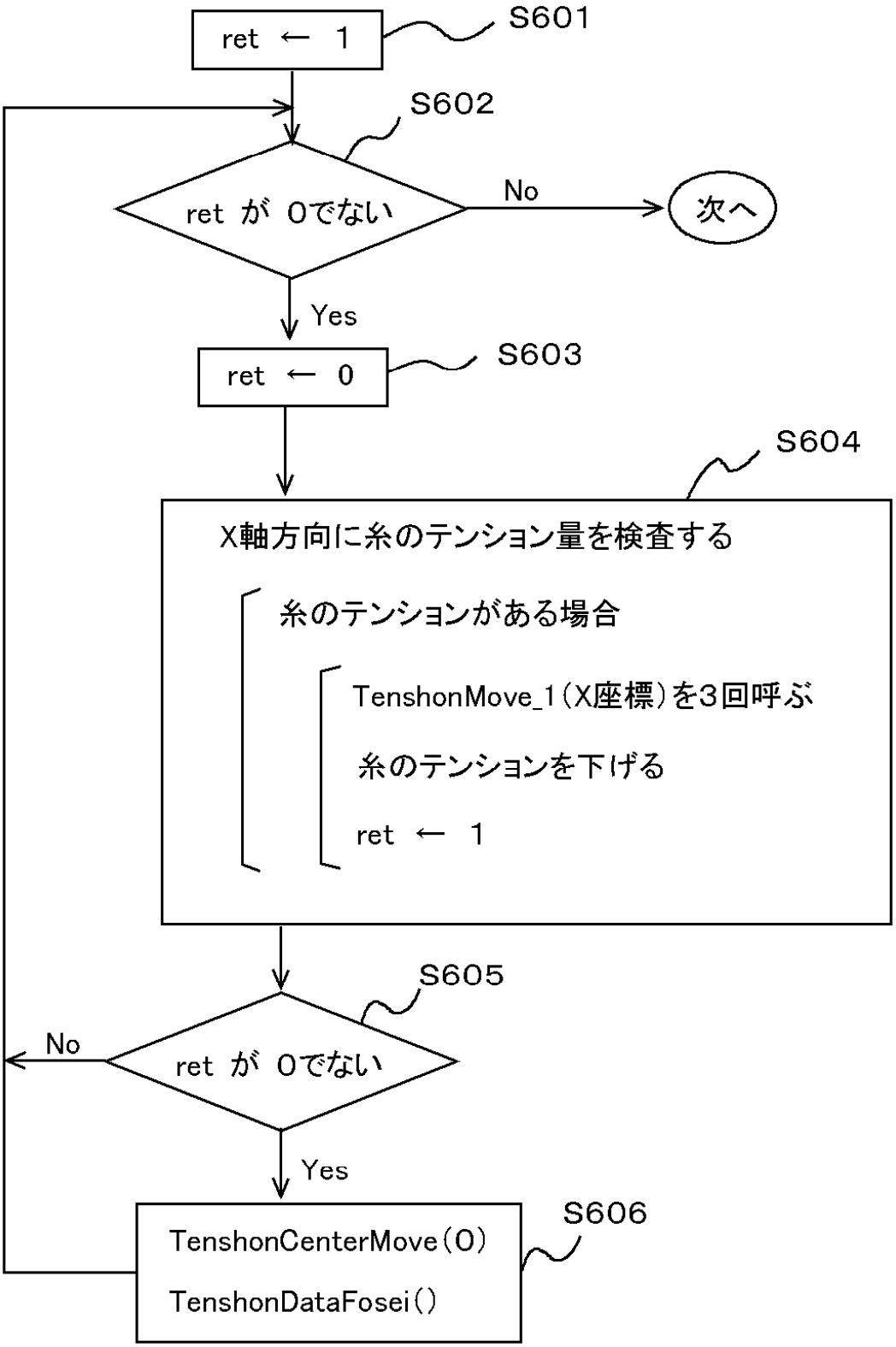

ret ← 1
S601
S602
ret が 0でない
No
次へ
Yes
ret ← 0
S603
S604
X軸方向に糸のテンション量を検査する
糸のテンションがある場合
TenshonMove_1(X座標)を3回呼ぶ
糸のテンションを下げる
ret ← 1
S605
No
ret が 0でない
Yes
TenshonCenterMove(0)
TenshonDataFosei()
S606

図7　YarnDataSeiriView()

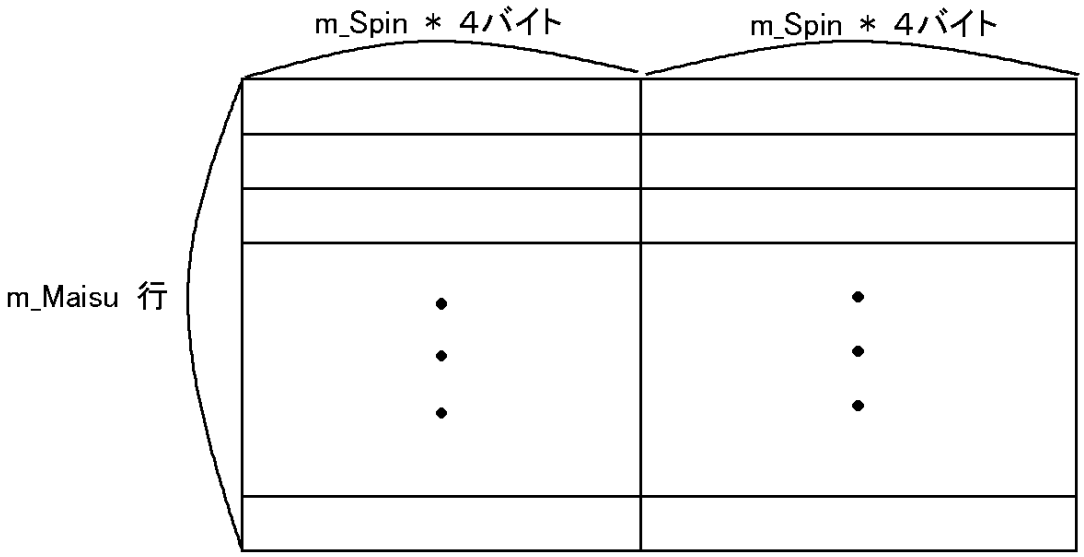
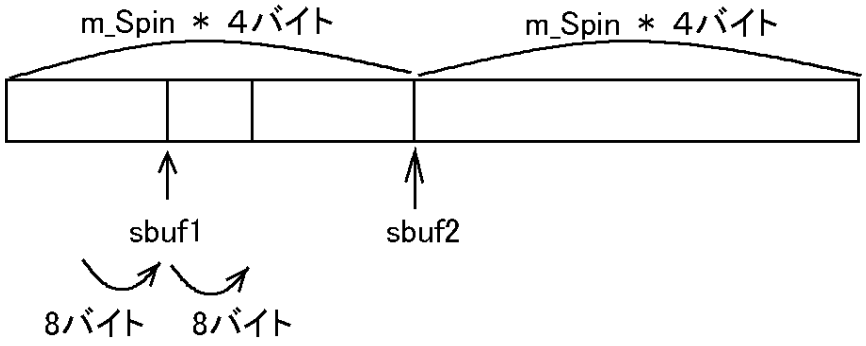


図8　YarnDataSeiriView()

(a)



(b)

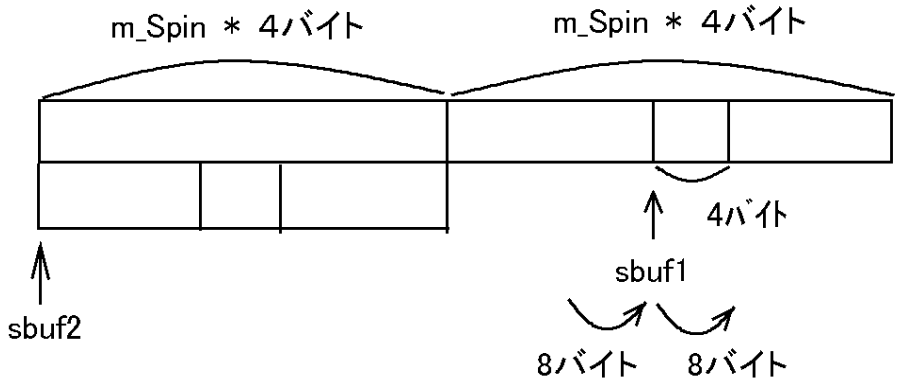


図9　YarnDataSeiriView()

(a)

(b)

```
* (sbuf2 + m_Spin * 4 - 1) ← dat1
* (sbuf2 + 6 ) ← dat2
* (sbuf2 + 6 ) ← dat1
* (sbuf2 + 6 ) ← dat2
* (sbuf2 + 6 ) ← dat1
* (sbuf2 + 6 ) ← dat2
•
•
•
```

図10　　SetTenshonHG()

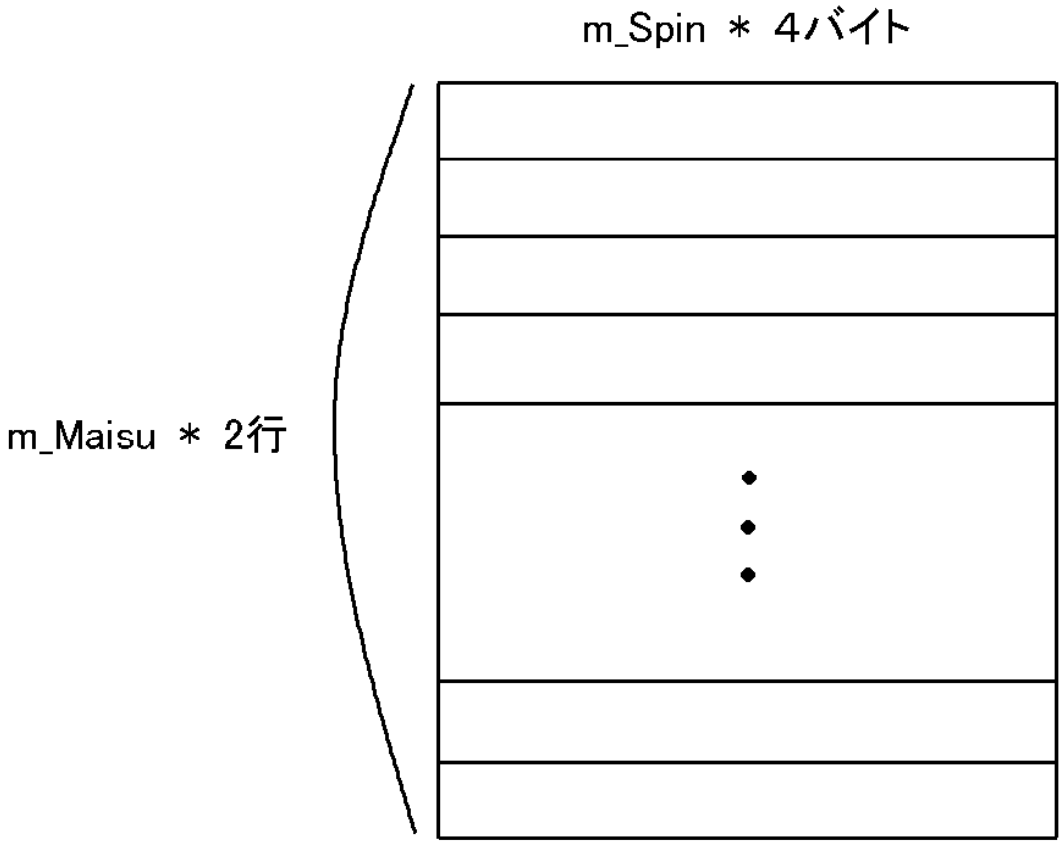


図11　SetTenshonHG()

初期化処理

　x1 ← 0

　y1 ← TCN_SIZE （予め定められた定数）

　偶数行目（0行目、2行目、4行目・・・）の場合

　　次の奇数行目の最左位置に●が入っていれば、y1に
　　格納されている値を4分の1にする

　奇数行目（1行目、3行目、5行目・・・）の場合

　　最左位置に●が入っていれば、y1に格納されている
　　値を4分の1にする
　　m_ViewHGの検査する位置、m_TenshonHGの代入する位置を4バイト
　　にずらす　　x1 ← 1

ループ （m_ViewHGの検査する位置とm_TenshonHGの代入する位置とを8バイトずつ
　右にずらしながら m_Spin/2 回まわる）

　m_ViewHGの検査する位置に●が入っていれば
　　m_TenshonHGの代入する位置に
　　TCN_SIZE ＊（ループを回った回数） ＋ TCN_SIZE/2
　　ylen ＋ y1/2
　を代入する

次のループのための処理

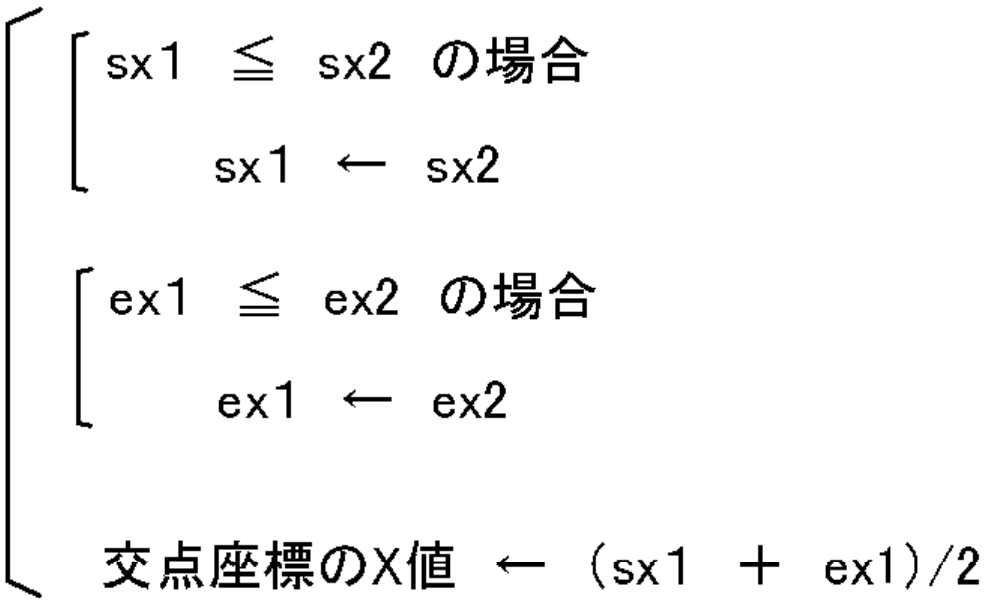
　ylenに格納されている値をy1だけ大きくする

図12 TenshonCenterMove (BOOL sw)

交点座標が格納されている場合

- 真上のスロットに交点座標が格納されている場合
  - sx1 ← ex1 ← 真上のスロットの交点座標のX値
  - sy1 ← ey1 ← 真上のスロットの交点座標のY値 － ly
- 右上のスロットに交点座標が格納されている場合
  - sx1 ← 右上のスロットの交点座標のX値 － lx
  - sy1 ← 右上のスロットの交点座標のY値 － ly
- 左上のスロットに交点座標が格納されている場合
  - ex1 ← 左上のスロットの交点座標のX値 － lx
  - ey1 ← 左上のスロットの交点座標のY値 － ly
- 真下のスロットに交点座標が格納されている場合
  - sx2 ← ex2 ← 真上のスロットの交点座標のX値
  - sy2 ← ey2 ← 真上のスロットの交点座標のY値 － ly
- 右下のスロットに交点座標が格納されている場合
  - sx2 ← 右下のスロットの交点座標のX値 － lx
  - sy2 ← 右下のスロットの交点座標のY値 － ly
- 左下のスロットに交点座標が格納されている場合
  - ex2 ← 左下のスロットの交点座標のX値 － lx
  - ey2 ← 左下のスロットの交点座標のY値 － ly

図13　TenshonCenterMove（BOOL sw）



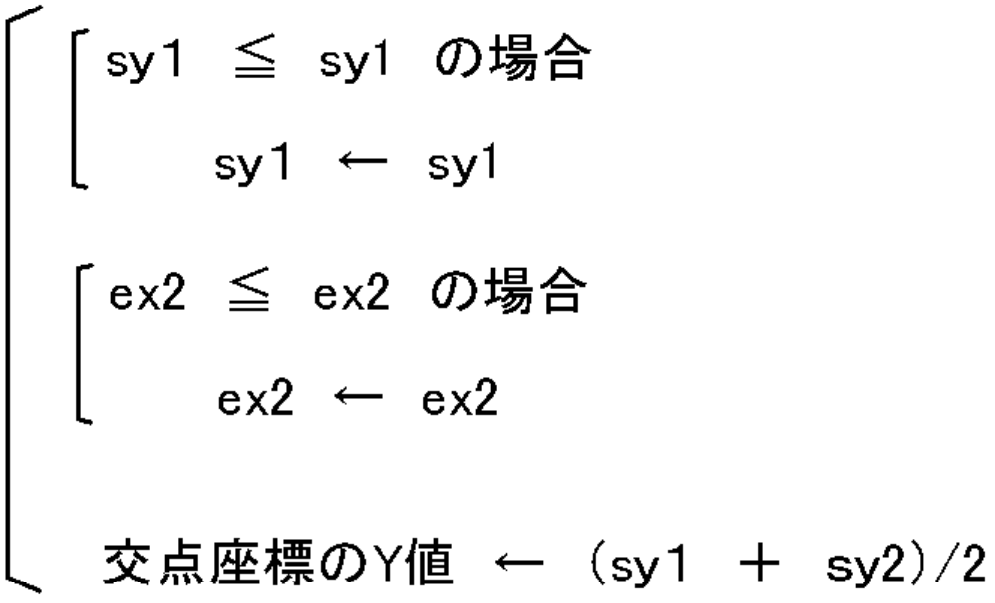


図14　TenshonDataFosei()

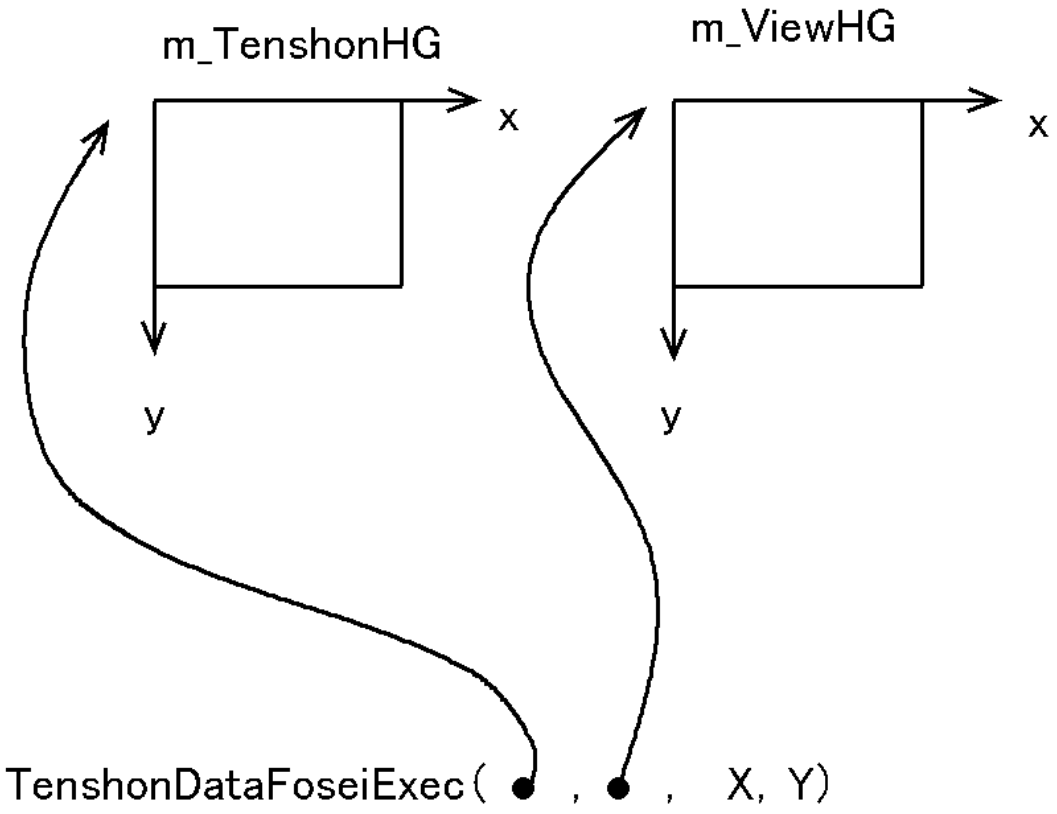


図15　TenshonDataFoseiExec(tbuf, vbuf ,　X, Y)

真上のスロットに交点座標が格納されている場合

sx0 ← 真下のスロットの交点座標のX値

右下のスロットに交点座標が格納されている場合

sx1 ← 右下のスロットの交点座標のX値

左下のスロットに交点座標が格納されている場合

sx2 ← 左下のスロットの交点座標のX値

図16　TenshonMove_1（yarn）

X座標がyarnで、m_TenshonHGの配列の上から下へループ（y座標をyとする）

交点座標が（x, y）又は（x－1, y）にある場合

- 真上のスロットに交点座標が格納されている場合
  - sx1, ex1
  - sy1, ey1

  をセットする
- 右上のスロットに交点座標が格納されている場合
  - sx1, sy1 をセットする
- 左上のスロットに交点座標が格納されている場合
  - ex1, ey1 をセットする
- 真下のスロットに交点座標が格納されている場合
  - sx2, ex2
  - sy2, ey2

  をセットする
- 右下のスロットに交点座標が格納されている場合
  - ex1, ey1 をセットする
- 左下のスロットに交点座標が格納されている場合
  - ex2, ey2 をセットする
- 交点座標が（x, y）にある場合
  - sx ← （sx1＋ ex2 ）/2
- 交点座標が（1-x, y）にある場合
  - sx ← （sx2＋ ex1 ）/2
- 交点座標のX値をsxにする

図17　　IchikawaWrite_mdatacnv（ buf, mdata ）

mdata[0], mdata[1], ・・・, mdata[11]へ0を代入する
bitcnt, bytcnt へ0を代入する

I = 0,1,2,・・・・,m_SpinIについて以下を行なう

- buf [(I%　＊ m_Wale ＊ 4] が0でない場合
  - mdata[bytcnt]のbitcut目のビットを1にする
- bitcnt を1だけ増やす
- Iが7, 15, 23, ・・・・である場合
  - bitcnt ＝0
  - bytcutの値を1増やす
- bufの指す位置を4バイト右へずらす