(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4983519号
(P4983519)
(45)発行日　平成24年7月25日(2012.7.25)　(24)登録日　平成24年5月11日(2012.5.11)

(51)Int.Cl.　F I
G06F 9/44 (2006.01)　G06F 9/06 620K

請求項の数 6 (全 45 頁)

(21)出願番号　特願2007-257758(P2007-257758)
(22)出願日　平成19年10月1日(2007.10.1)
(65)公開番号　特開2009-87144(P2009-87144A)
(43)公開日　平成21年4月23日(2009.4.23)
審査請求日　平成22年8月4日(2010.8.4)

(73)特許権者　000002945
オムロン株式会社
京都市下京区塩小路通堀川東入南不動堂町
８０１番地
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平
(74)代理人　100096781
弁理士　堀井　豊
(74)代理人　100098316
弁理士　野田　久登

最終頁に続く

(54)【発明の名称】開発支援装置および開発支援プログラム

(57)【特許請求の範囲】
【請求項１】
画像処理装置において実現される画像処理機能であって、複数の画像処理モジュールによって特定される画像処理機能をカスタマイズする装置であって、
前記画像処理モジュールは、当該画像処理モジュールの画像処理機能を特定する第１画像処理モジュールと、前記第１画像処理モジュールに対する設定情報の入力に利用される画面を表示するため情報である第２画像処理モジュールとを含み、
ユーザの操作を受け付ける操作入力部と、
複数の画像処理モジュールを記憶する画像処理モジュール記憶部と、
前記第２画像処理モジュールに基づいて表示された画面に対して入力された情報に基づいて、前記第１画像処理モジュールに対して画像処理動作の設定を行なう設定処理手段とを備え、
前記第１画像処理モジュールは、当該第１画像処理モジュールにおいて使用される変数の定義と、画像処理の対象とする範囲の最大個数を特定する情報と、各前記画像処理の対象とする範囲の形状と大きさと位置を特定する情報と、前記画像処理の基準となる画像情報の最大個数を特定する情報とを含み、
前記第２画像処理モジュールは、前記第１画像処理モジュールに対して各前記画像処理の対象とする範囲の形状を設定するための画面である図形設定用画面を表示させる第１表示情報と、前記第１画像処理モジュールによって実現される画像処理機能が実現された結果の出力に対するパラメータを設定するための画面であるパラメータ設定用画面を表示さ

せる第２表示情報と、前記画像処理の対象とする範囲についての位置を設定するための画面である座標設定用画面を表示させる第３表示情報とを含む、開発支援装置。
【請求項２】
前記第２画像処理モジュールに基づいて表示される画面に貼り付ける部品を記憶する部品記憶部と、
前記操作入力部が受け付けた操作に基づいて、前記第２画像処理モジュールを、前記部品記憶部に記憶された部品を使用して編集するモジュール開発部をさらに備える、請求項１に記載の開発支援装置。
【請求項３】
前記第１画像処理モジュールは、他の画像処理モジュールの機能を内包するか否かを特定する情報を含む、請求項１または請求項２に記載の開発支援装置。
【請求項４】
前記画像処理モジュールで使用する複数の変数の定義に関する情報を含むファイルを記憶する変数ファイル記憶部をさらに備える、請求項１～請求項３のいずれかに記載の開発支援装置。
【請求項５】
前記変数ファイル記憶部は、ヘッダ情報のファイルとして前記複数の変数の定義に関する情報を含むファイルを記憶し、
ＣＳＶ（Comma Separated Values）形式で記述されたファイルであって複数の変数の定義を記述したファイルをヘッダ情報のファイルに変換するファイル変換部をさらに備える、請求項４に記載の開発支援装置。
【請求項６】
画像処理装置において実現される画像処理機能であって、複数の画像処理モジュールによって特定される画像処理機能をカスタマイズする装置において実行される開発支援プログラムであって、
前記画像処理モジュールは、当該画像処理モジュールの画像処理機能を特定する第１画像処理モジュールと、前記第１画像処理モジュールに対する設定情報の入力に利用される画面を表示するため情報である第２画像処理モジュールとを含み、
ユーザの操作を受け付けるステップと、
受け付けた操作に基づいて、前記第１画像処理モジュールにおいて使用される変数の定義と、画像処理の対象とする範囲の最大個数を特定する情報と、各前記画像処理の対象とする範囲の形状と大きさと位置を特定する情報と、前記画像処理の基準となる画像情報の最大個数を特定する情報とを含むように、前記第１画像処理モジュールを作成するステップと、
受け付けた操作に基づいて、前記第１画像処理モジュールに対して各前記画像処理の対象とする範囲の形状を設定するための画面である図形設定用画面を表示させる第１表示情報と、前記第１画像処理モジュールによって実現される画像処理機能が実現された結果の出力に対するパラメータを設定するための画面であるパラメータ設定用画面を表示させる第２表示情報と、前記画像処理の対象とする範囲についての位置を設定するための画面である座標設定用画面を表示させる第３表示情報とを含むように前記第２画像処理モジュールを作成するステップとを前記装置に実行させる、開発支援プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、画像処理システムの開発支援に関し、特に、計測対象物の画像に対して画像処理を行なうことにより検査や判別を行なう画像処理検査装置を含むシステムにおける画像処理検査装置で実行されるプログラムの開発に関する。
【背景技術】
【０００２】
従来から、たとえば特許文献１や特許文献２に開示されるように、画像処理検査装置に

おける物品の視覚的な検査に関し、プログラムの開発が行なわれてきた。
【０００３】
たとえば、特許文献１では、ライブラリに集められ画像処理検査装置のメーカによって生成されるオブジェクトプログラムを用いてソースプログラムが作成され、当該ソースプログラムがコンパイル・リンクされることにより実行形式のプログラムが作成される。
【０００４】
また、特許文献２では、予めシステム開発用コンピュータに記録されている複数の画像処理モジュールを実行させる順番を組合わせて、検査対象画像の処理手順を決定する技術が開示されている。
【特許文献１】特開平１０－２２８３７１号公報
【特許文献２】特開２００３－２９６１１２号公報
【発明の開示】
【発明が解決しようとする課題】
【０００５】
従来の技術では、画像処理検査装置で実行させるプログラムの開発に際し、処理内容の構成単位であるオブジェクトプログラムや画像処理モジュールとしては、予め準備されたものが用いられていた。
【０００６】
なお、画像処理検査装置における物品の計測態様の多様化などに伴い、プログラムの開発についてより柔軟にカスタマイズできることが希望されるようになってきている。
【０００７】
しかしながら、画像処理検査装置のメーカにおいて個々の顧客の希望に沿うようなプログラム開発を行なった場合、コストの面で採算が取れなくなるという問題が生じる。ただし、上記したようなオブジェクトプログラムや画像処理モジュールをシステムインテグレータ（ＳＩ）に利用者の個々のニーズに合わせて作成させた場合、個々の処理の内容を記述したソースコードを作成しなければならず、ＳＩにかかる負担が大きすぎるという問題があった。
【０００８】
本発明はかかる実情に鑑み考え出されたものであり、その目的は、画像処理検査装置において実行されるプログラムの開発を容易にすることができる開発支援装置および開発支援プログラムを提供することである。
【課題を解決するための手段】
【０００９】
本発明に従った開発支援装置は、画像処理装置において実現される画像処理機能であって、複数の画像処理モジュールによって特定される画像処理機能をカスタマイズする装置であって、前記画像処理モジュールは、当該画像処理モジュールの画像処理機能を特定する第１画像処理モジュールと、前記第１画像処理モジュールに対する設定情報の入力に利用される画面を表示するため情報である第２画像処理モジュールとを含み、ユーザの操作を受け付ける操作入力部と、複数の画像処理モジュールを記憶する画像処理モジュール記憶部と、前記第２画像処理モジュールに基づいて表示された画面に対して入力された情報に基づいて、前記第１画像処理モジュールに対して画像処理動作の設定を行なう設定処理手段とを備え、前記第１画像処理モジュールは、当該第１画像処理モジュールにおいて使用される変数の定義と、画像処理の対象とする範囲の最大個数を特定する情報と、各前記画像処理の対象とする範囲の形状と大きさと位置を特定する情報と、前記画像処理の基準となる画像情報の最大個数を特定する情報とを含み、前記第２画像処理モジュールは、前記第１画像処理モジュールに対して各前記画像処理の対象とする範囲の形状を設定するための画面である図形設定用画面を表示させる第１表示情報と、前記第１画像処理モジュールによって実現される画像処理機能が実現された結果の出力に対するパラメータを設定するための画面であるパラメータ設定用画面を表示させる第２表示情報と、前記画像処理の対象とする範囲についての位置を設定するための画面である座標設定用画面を表示させる

第３表示情報とを含むことを特徴とする。

【００１０】

また、本発明の開発支援装置では、前記第２画像処理モジュールに基づいて表示される画面に貼り付ける部品を記憶する部品記憶部と、前記操作入力部が受け付けた操作に基づいて、前記第２画像処理モジュールを、前記部品記憶部に記憶された部品を使用して編集するモジュール開発部をさらに備えることが好ましい。

【００１１】

また、本発明の開発支援装置では、前記第１画像処理モジュールは、他の画像処理モジュールの機能を内包するか否かを特定する情報を含むことが好ましい。

【００１２】

また、本発明の開発支援装置では、前記画像処理モジュールで使用する複数の変数の定義に関する情報を含むファイルを記憶する変数ファイル記憶部をさらに備えることが好ましい。

【００１３】

また、本発明の開発支援装置では、前記変数ファイル記憶部は、ヘッダ情報のファイルとして前記複数の変数の定義に関する情報を含むファイルを記憶し、ＣＳＶ（Comma Separated Values）形式で記述されたファイルであって複数の変数の定義を記述したファイルをヘッダ情報のファイルに変換するファイル変換部をさらに備えることが好ましい。

【００１４】

本発明に従った開発支援プログラムは、画像処理装置において実現される画像処理機能であって、複数の画像処理モジュールによって特定される画像処理機能をカスタマイズする装置において実行される開発支援プログラムであって、前記画像処理モジュールは、当該画像処理モジュールの画像処理機能を特定する第１画像処理モジュールと、前記第１画像処理モジュールに対する設定情報の入力に利用される画面を表示するため情報である第２画像処理モジュールとを含み、ユーザの操作を受け付けるステップと、受け付けた操作に基づいて、前記第１画像処理モジュールにおいて使用される変数の定義と、画像処理の対象とする範囲の最大個数を特定する情報と、各前記画像処理の対象とする範囲の形状と大きさと位置を特定する情報と、前記画像処理の基準となる画像情報の最大個数を特定する情報とを含むように、前記第１画像処理モジュールを作成するステップと、受け付けた操作に基づいて、前記第１画像処理モジュールに対して各前記画像処理の対象とする範囲の形状を設定するための画面である図形設定用画面を表示させる第１表示情報と、前記第１画像処理モジュールによって実現される画像処理機能が実現された結果の出力に対するパラメータを設定するための画面であるパラメータ設定用画面を表示させる第２表示情報と、前記画像処理の対象とする範囲についての位置を設定するための画面である座標設定用画面を表示させる第３表示情報とを含むように前記第２画像処理モジュールを作成するステップとを前記装置に実行させることを特徴とする。

【発明の効果】

【００１５】

本発明によれば、開発支援装置において、画像処理モジュールとして、当該画像処理モジュールの画像処理機能を特定する第１画像処理モジュールと、当該第１画像処理モジュールに対する設定情報の入力に利用される画面を表示するため情報である第２画像処理モジュールとが準備される。これにより、ＳＩは、第２画像処理モジュールに基づいて表示される画面を利用して設定情報を入力することにより、カスタマイズされた画像処理モジュールを作成することができる。

【００１６】

したがって、ＳＩが画像処理検査装置において実行されるプログラムを開発する際に、当該プログラムに組み込む画像処理モジュールをカスタマイズしようとする場合、当該ＳＩに、そのような画像処理モジュール全体のソースコードを記述させる、といった負担を課す必要がなくなる。

【００１７】

これにより、画像処理検査装置において実行されるプログラムの開発を容易にすることができる。

【発明を実施するための最良の形態】

【００１８】

以下、本発明の開発支援装置の一実施の形態について、図面を参照しつつ説明する。なお、同一の構成要素には各図において同一の符号を付し、詳細な説明は繰返さない。

【００１９】

［開発支援装置のシステムの構成］

図１は、本実施の形態の開発支援装置のシステムの構成を模式的に示す図である。

【００２０】

図１を参照して、開発支援装置は、当該開発支援装置のハードウェア５０１と、ハードウェア５０１を制御するコンピュータシステムを全体的に管理するＯＳ（Operating System）５０２と、画像処理検査装置における具体的な処理内容を記述したプログラムからなる基本フロー５０７と、基本フロー５０７の各プログラムが参照するプログラムからなるライブラリ５０３とを含む。

【００２１】

基本フロー５０７は、画像処理検査装置における検査処理について、検査処理中にディスプレイに表示される画面の表示を制御する運転画面ＵＩ（User Interface）５０６と、当該検査処理に含まれる複数の処理段階のそれぞれの処理内容を制御するための画像処理モジュール（画像処理モジュールＡ５０９Ａ～画像処理モジュールＮ５０９Ｎ）とを含む。

【００２２】

画像処理モジュール５０９Ａ～５０９Ｎは、それぞれ、実際の処理内容を制御する処理モジュールＭＳ（Measurement Structure）５０４Ａ～５０４Ｎと、各処理モジュールＭＳについての設定情報をディスプレイ（画像処理検査装置または開発支援装置のディスプレイ）に表示させるための処理モジュールＵＩ（User Interface）５０５Ａ～５０５Ｎとを含む。

【００２３】

開発支援装置（または、画像処理検査装置）において、ＳＩは、設定データ５１０に基づいて、基本フロー５０７を作成する。基本フロー５０７は、上記したように、運転画面ＵＩ５０６と複数の画像処理モジュール５０９Ａ～５０９Ｎを含む。開発支援装置では、（後述する画像処理モジュール記憶部２９１に）複数の画像処理モジュールが用意されている。ＳＩは、それらの処理モジュールから、基本フロー５０７に組み込むべきものを選択するとともに、組み込む画像処理モジュールの処理モジュールＭＳに対して種々の値の設定を行なう。なお、ＳＩは、処理モジュールＭＳに対する値等の設定を行なう場合、当該処理モジュールＭＳのソースコードを直接書き換えたり追加したりすることもできるし、ディスプレイに当該処理モジュールＭＳに対応した処理モジュールＵＩに基づく画面を表示させて当該画面を介して設定するべき値を入力することもできる。また、処理モジュールＭＳに対する値等の設定は、対応する画像処理モジュールが基本フローに組み込まれる前に行なうこともできる。

【００２４】

ＳＩが処理モジュールＵＩを利用して処理モジュールＭＳの設定を行なうために、本実施の形態の開発支援装置では、画像処理モジュールとして、処理モジュールＭＳを作成するための素材（本明細書では、特に区別する必要がない場合には、処理モジュールＵＩを利用した設定が行なわれる前の状態の処理モジュールＭＳについても「処理モジュールＭＳ」と呼ぶ）が予め準備されるとともに、各処理モジュールＭＳに対応した処理モジュールＵＩが準備されている。これにより、ＳＩは、画像処理モジュールを作成する際（カスタマイズする際）に、処理モジュールＭＳにおけるプログラム（ソースコード）の記述を行なう代わりに処理モジュールＵＩに基づいて表示される画面に従って値（情報）を入力すればよくなる。処理モジュールＵＩに基づいた画面に従って情報が入力されることによ

り、処理モジュールＭＳを作成するための素材に対して適宜情報の書込みが行なわれ、これにより、ＳＩが意図する処理モジュールＭＳが作成される。

【００２５】

本実施の形態の開発支援装置では、ＳＩによる基本フローの作成の前に、処理モジュールＭＳの素材および当該処理モジュールＭＳに対応する処理モジュールＵＩが作成される。なお、処理モジュールＭＳは、後述するように、スケルトン形式で保存されているテンプレートに対して適宜情報の書換え（または、追加）が行なわれることにより、作成される。また、処理モジュールＵＩは、開発支援装置のハードディスク（後述するモジュール作成用データ記憶部２５）に記憶された画像情報に適宜情報が追加されることにより作成される。

【００２６】

開発支援装置において作成された基本フローは、ライブラリとともに、画像処理検査装置へインストールされる。画像処理検査装置では、基本フローを実行させることにより、部品の検査等の処理が実現される。なお、開発支援装置では、基本フローのソースコードが作成された場合、これに基づいて、開発支援装置と画像処理検査装置のそれぞれに対応したＯＳ上で動作する基本フローが作成されることが好ましい。

【００２７】

［開発支援装置のハードウェアの構成］

図２は、本発明の一実施の形態である開発支援装置のハードウェア構成を模式的に示す図である。開発支援装置１は、画像処理検査装置の画像処理機能をカスタマイズする装置であって、画像処理検査装置で動作する実行形式のプログラムを作成する。

【００２８】

図２を参照して、本実施の形態の開発支援装置１は、汎用のコンピュータで実現されており、ディスプレイ２と、ＣＰＵ（Central Processing Unit）およびメモリ４と、操作入力部７と、ハードディスク５と、リムーバブルメディア６を備える。

【００２９】

操作入力部７は、マウス２０とキーボード２１を含み、開発支援装置１のユーザであるＳＩからの操作入力を受け付ける。なお、開発支援装置１では、ＳＩによる操作とは別に、処理モジュールＭＳの素材等を作成する処理が実行される。操作入力部７は、このような処理における操作入力も受け付ける。

【００３０】

ハードディスク５は、カスタマイズプログラム記憶部１４と、画像処理モジュール記憶部２９１と、基本フロー記憶部２９４と、イベント処理モジュール記憶部２９２と、運転画面記憶部２９３と、テンプレート記憶部１６と、部品記憶部１５と、ＯＳ類記憶部２２と、ライブラリ類記憶部２３と、シミュレーションソフト記憶部１７と、マニュアル類記憶部１８と、実行プログラム記憶部１９と、モジュール作成用プログラム記憶部２４と、モジュール作成用データ記憶部２５とを含む。

【００３１】

カスタマイズプログラム記憶部１４は、カスタマイズプログラム（Application Producer）を記憶する。カスタマイズプログラムは、ＣＰＵに読み込まれて実行され、コンピュータを画像処理機能の開発支援装置１の各構成要素として機能させる。

【００３２】

画像処理モジュール記憶部２９１は、ソースコード形式の画像処理モジュールを記憶する。画像処理モジュールは、色面積の計算、エッジ位置検出などの画像処理の基本単位を実行するためのもので、複数個のファイルからなる。そして、画像処理モジュール記憶部２９１は、出荷時に予め組み込まれている画像処理モジュールと、ＳＩによって新たに作成された画像処理モジュールとを記憶する。

【００３３】

画像処理モジュールは、画像処理結果の表示部分、画像処理結果の出力部分、画像処理の処理内容である演算部分、および画像処理のパラメータの設定のためのユーザインタフ

ェース部分を少なくとも含む。画像処理のパラメータの設定には、サーチなどのモデル画像の領域座標および画像の設定、画像計測などの計測領域の設定、面積計測での穴埋め（輪郭の中を塗りつぶす）機能の実行の有無の設定、計測結果の判定を行なうための上下限値の設定などがある。

【００３４】

また、上記したように、各画像処理モジュールは、画像処理の基本単位を実行するための部分（処理モジュールＭＳ）とともに、当該基本単位の処理に対する各種の設定を行なうための画面を表示させるための部分（処理モジュールＵＩ）を含む。

【００３５】

基本フロー記憶部２９４は、ＳＩによって作成された画像処理モジュールの実行順序と、そのパラメータからなる基本フローを記憶する。

【００３６】

イベント処理モジュール記憶部２９２は、ＳＩによって新たに作成されたソースコード形式のイベント処理モジュールを記憶する。イベント処理モジュールは、画像処理検査装置による画像処理の実行中にイベントが発生した場合に、イベントが発生したタイミングで実行される処理を定めたものである。イベント処理モジュールは、画像処理モジュールのようなパラメータの設定のためのユーザインタフェース部分を含まない。

【００３７】

運転画面記憶部２９３は、ＳＩによって新たに作成された運転画面を記憶する。運転画面は、画像処理検査装置による画像処理の結果を表示するための画面であり、記憶されている運転画面は、ビジュアルベーシックのファイルからなる。

【００３８】

テンプレート記憶部１６は、画像処理モジュールのテンプレートおよび運転画面のテンプレートを記憶する。

【００３９】

運転画面のテンプレートは、ビジュアルベーシック（登録商標）のファイルからなる。テンプレートとは、新たな画像処理モジュールおよび運転画面を作成するときの原型となるものであって、１個以上のファイルで構成され、ファイル内には、実行する典型的な呼び出し関数が記述されているものである。ファイル内の呼び出し関数には、典型的な実行命令が記述されていたり、あるいはリターン値だけが記載されていたりする。なお、このようなテンプレートのプログラムパターンを、本明細書では適宜スケルトン形式という。

【００４０】

図３は、画像処理モジュールのテンプレートを構成するファイルの一覧とその機能の例を表わす図である。画像処理モジュール記憶部２９１に記憶されている画像処理モジュールも、「Sample」が個々の画像処理モジュール特有の名称となっている以外は、図３に示されたものと同様のファイル構成である。

【００４１】

図３を参照して、「Sample_jpn.msg」は、日本語のメッセージファイルである。「Sample_eng.msg」は、英語のメッセージファイルである。

【００４２】

ＵＩプログラム（処理モジュールＵＩ）は、画像処理のパラメータ（各種の閾値など）を設定するためのユーザインタフェースを規定したものであり、２個の必須ファイルを含む。「NormalForm.vb」は、通常使用されるユーザインタフェースを記述したビジュアルベーシックのファイルである。「NonstopForm.vb」は、計測中に一時的に上下限値を変更するための簡易ノンストップをするときのユーザインタフェースを記述したビジュアルベーシックのファイルである。

【００４３】

ＭＳプログラム（処理モジュールＭＳ）は、画像処理の処理内容である演算（つまり計測）、画像処理結果（つまり計測結果）の表示、画像処理結果（つまり計測結果）の出力などを規定したものであり、９個の必須ファイルと、５個の任意ファイルとを含む。ＭＳ

プログラムのうち、どのファイルが画像処理モジュールの表示部分、出力部分、および演算（計測）部分のいずれに対応するかがファイル名によって識別可能とされている。これにより、ＳＩがテンプレートを編集する際にどのファイルを修正すべきであるかが容易にわかる。また、ＭＳプログラムのテンプレートに記述される関数は、その中身が戻り値（リターン値）だけが記述されているスケルトンの形式である。

【００４４】

「AssignProc.cpp」は、処理ユニット登録時処理関連の処理を記述したテキストファイルである。処理ユニットは、画像処理モジュールと同義である。「Assign」は登録を意味し、「Proc」は処理を意味する「Processing」を略したものであるから、ＳＩがテンプレートの編集の際に、「AssignProc.cpp」が上記処理を記述するものであることを容易に識別できる。

「FigureUpdate.cpp」は、図形データ更新時処理関連の処理を記述したテキストファイルである。「Figure」は図形を意味し、「Update」は更新を意味するものであるから、ＳＩがテンプレートの編集の際に、「FigureUpdate.cpp」が上記処理を記述するものであることを容易に識別できる。

【００４５】

「ItemDefs.h」は、処理項目の各種定義を記述したテキストファイルである。処理項目は、画像処理モジュールおよび処理ユニットと同義である。「「Item」は項目を意味し、「Def」は定義を意味する「Definition」を略したものであるから、ＳＩがテンプレートの編集の際に、「ItemDefs.h」が上記処理を記述するものであることを容易に識別できる。

【００４６】

「ItemInit.cpp」は、処理項目の初期化処理関連の処理を記述したテキストファイルである。「Item」は項目を意味し、「Init」は初期化を意味する「Initialization」を略したものであるから、ＳＩがテンプレートの編集の際に、「ItemInit.cpp」が上記処理を記述するものであることを容易に識別できる。

【００４７】

「MeasureDisp.cpp」は、計測結果表示処理関連の処理を記述したテキストファイルである。「Measure」は計測を意味し、「Disp」は表示を意味する「Display」を略したものであるから、ＳＩがテンプレートの編集の際に、「MeasureDisp.cpp」が上記処理を記述するものであることを容易に識別できる。

【００４８】

「MeasureInit.cpp」は、計測初期化／終了関連の処理を記述したテキストファイルである。「Measure」は計測を意味し、「Init」は初期化を意味する「Initialization」を略したものであるから、ＳＩがテンプレートの編集の際に、「MeasureInit.cpp」が上記処理を記述するものであることを容易に識別できる。

【００４９】

「MeasureOut.cpp」は、計測結果出力処理関連の処理を記述したテキストファイルである。「Measure」は計測を意味し、「Out」は出力を意味するものであるから、ＳＩがテンプレートの編集の際に、「MeasureOut.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５０】

「MeasureProc.cpp」は、計測処理関連の処理を記述したテキストファイルである。「Measure」は計測を意味し、「Proc」は処理を意味する「Processing」を略したものであるから、ＳＩがテンプレートの編集の際に、「MeasureProc.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５１】

「UnitData.cpp」は、処理ユニットデータ設定／取得関連の処理を記述したテキストファイルである。「Unit」はユニットを意味し、「Data」はデータを意味するものであるから、ＳＩがテンプレートの編集の際に、「UnitData.cpp」が上記処理を記述するものであ

ることを容易に識別できる。

【００５２】

「FigureData.cpp」、処理ユニットの図形データ操作関連の処理を記述したテキストファイルである。「Figure」は図形を意味し、「Data」はデータを意味するものであるから、ＳＩがテンプレートの編集の際に、「FigureData.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５３】

「RenumProc.cpp」は、処理ユニットの参照番号更新処理関連の処理を記述したテキストファイルである。「Renum」は番号更新を意味する「Renumber」を略したものであり、「Proc」は処理を意味する「Processing」を略したものであるから、ＳＩがテンプレートの編集の際に、「RenumProc.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５４】

「SaveLoad.cpp」は、処理ユニットのデータのセーブおよびロード関連の処理を記述したテキストファイルである。「Save」セーブ（保存）を意味し、「Load」はロード（読出し）を意味するものであるから、ＳＩがテンプレートの編集の際に、「SaveLoad.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５５】

「SetupData.cpp」は、設定時の画像処理関連の処理を記述したテキストファイルである。「Setup」は設定を意味し、「Data」はデータを意味するものであるから、ＳＩがテンプレートの編集の際に、「SetupData.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５６】

「ThroughProc.cpp」は、スルー表示のための画像入力および生成処理関連の処理を記述したテキストファイルである。「Through」はスルー（直通）を意味し、「Proc」は処理を意味する「Processing」を略したものであるから、ＳＩがテンプレートの編集の際に、「ThroughProc.cpp」が上記処理を記述するものであることを容易に識別できる。

【００５７】

部品記憶部１５は、画像処理モジュールのユーザインタフェースの画面のテンプレートおよび運転画面のテンプレートに貼り付けることができる部品（ＡｃｔｉｖｅＸコントロール）を記憶する。また、部品記憶部１５は、イベント処理開発で利用する部品を記憶する。部品記憶部１５に記憶されている部品は、表示されている画面に含まれるメニューのうちのツールから選択できる。

【００５８】

なお、出荷時の開発支援装置の画像処理モジュール記憶部２９１には、上記した各テンプレートに処理内容が記述されることによって作成された画像処理モジュールも記憶されている。このような画像処理モジュールには、上記した処理モジュールＭＳと処理モジュールＵＩが含まれる。

【００５９】

本実施の形態の開発支援装置では、ＳＩは、基本フローを作成する際に、テンプレート記憶部１６に記憶されたテンプレートに直接データを入力することによって作成した画像処理モジュールを基本フローに組み込んでも良いし、予め画像処理モジュール記憶部２９１に記憶された画像処理モジュール（上記した処理モジュールＭＳの素材と処理モジュールＵＩとを含む）を適宜カスタマイズして組み込んでも良い。なお、開発支援装置の出荷時（または、少なくともＳＩに操作される前であって画像処理モジュールの作成が行なわれた後）に記憶されている画像処理モジュールがどのように作成されるかについては、後述する。

【００６０】

ＯＳ類記憶部２２は、Ｗｉｎｄｏｗｓ（登録商標）ＸＰおよびＶｉｓｕａｌＳｔｕｄｉｏを記憶する。

【００６１】
ライブラリ類記憶部２３は、ライブラリおよびドライバを記憶する。
シミュレーションソフト記憶部１７は、図１の画像処理機能の開発支援装置１で実行可能な実行形式のプログラムを起動するためのシミュレーションソフトを記憶する。
【００６２】
マニュアル類記憶部１８は、マニュアル、チュートリアル、ヘルプおよびサンプルを記憶する。
【００６３】
実行プログラム記憶部１９は、出荷時に予め組み込まれている画像処理モジュールの実行形式のプログラム、およびビルド実行部１３で作成された実行形式のプログラムを記憶する。
【００６４】
カスタマイズプログラム、部品、テンプレート、ＯＳ類、ライブラリ類、シミュレーションソフト、およびマニュアル類は、外部からリムーバブルメディア６を用いてハードディスク５にインストールすることができる。
【００６５】
ＣＰＵおよびメモリ４は、ＯＳ類記憶部２２からＷｉｎｄｏｗｓＸＰおよびＶｉｓｕａｌＳｔｕｄｉｏ（登録商標）を読出し、さらにカスタマイズプログラム記憶部１４からカスタマイズプログラムを読み出して、これらを実行することによって、実行制御部８、基本フロー開発部９、画像処理モジュール開発部１０、運転画面開発部１１、イベント処理開発部１２、および、ビルド実行部１３として機能する。また、ＣＰＵおよびメモリ４は、ＯＳ類記憶部２２からＷｉｎｄｏｗｓＸＰおよびＶｉｓｕａｌＳｔｕｄｉｏ（登録商標）を読出し、さらにモジュール作成用プログラム記憶部２４から画像モジュール作成用プログラムを読み出し、これらを実行することによって、画像処理モジュール作成部１０Ａとして機能する。
【００６６】
実行制御部８は、この画像処理機能の開発支援装置１の全体の動作を制御する。
基本フロー開発部９は、操作入力部７を通じたユーザの操作に従って、画像処理モジュール記憶部２９１に記憶されている複数個の画像処理モジュールの実行順序を決定して、決定した実行順序と、そのパラメータからなる基本フローを基本フロー記憶部２９４に記憶する。
【００６７】
画像処理モジュール開発部１０は、テンプレート記憶部１６に記憶されている画像処理モジュールのテンプレートを読み出して表示し、操作入力部７を通じたＳＩの操作に従って、画像処理モジュールのテンプレートを編集して、編集後の画像処理モジュールを画像処理モジュール記憶部２９１に記憶する。
【００６８】
また、画像処理モジュール開発部１０は、画像処理モジュール記憶部２９１に記憶されている処理モジュールＵＩに対応した画面を表示させ、そして、当該画面に対応して入力された情報を当該処理モジュールＵＩに対応する処理モジュールＭＳの素材に書込むことにより、画像処理モジュールのカスタマイズを行なう。カスタマイズされた画像処理モジュールは、画像処理モジュール記憶部２９１に記憶される。
【００６９】
画像処理モジュール作成部１０Ａは、ＳＩが利用する処理モジュールＵＩおよび当該処理モジュールＵＩを利用して入力された情報によって作成（編集）が可能な処理モジュールＭＳの素材を作成する。処理モジュールＵＩの作成は、モジュール作成用データ記憶部２５に記憶された画像情報が適宜変更することによって行なわれ、処理モジュールＭＳの素材は、テンプレート記憶部１６に記憶されている処理モジュールＭＳのテンプレートを編集することによって行なわれる。
【００７０】

運転画面開発部１１は、テンプレート記憶部１６に記憶されている運転画面のテンプレートを読み出して表示し、操作入力部７を通じたＳＩの操作に従って、運転画面のテンプレートに部品記憶部１５に記憶されている部品を貼り付けて運転画面のテンプレートを編集して、編集後の運転画面を運転画面記憶部２９３に記憶する。

【００７１】

イベント処理開発部１２は、操作入力部７を通じたＳＩの操作に従って、イベントモジュールを作成して、イベント処理モジュール記憶部２９２に記憶する。

【００７２】

ビルド実行部１３は、ソースコードをコンパイルしてオブジェクトコードを生成し、さらにオブジェクトコードとライブラリ類記憶部２３内のライブラリおよびドライバとをリンクすることによって、実行形式のプログラムを作成して実行プログラム記憶部１９に記憶する。ビルド実行部１３は、画像処理検査装置が動作するＯＳであるＷｉｎｄｏｗｓ（登録商標）ＣＥ用の実行形式のプログラムと、図１の画像処理機能の開発支援装置１が動作するＯＳであるＷｉｎｄｏｗｓＸＰ用の実行形式のプログラムの両方を作成する。ＷｉｎｄｏｗｓＣＥ用の実行形式のプログラムは、リムーバブルメディア６などを用いて、画像処理検査装置にインストールされる。ＷｉｎｄｏｗｓＸＰ用の実行形式のプログラムは、図１の装置で、画像処理検査装置におる画像処理をエミュレート（シミュレーション）するために用いられる。

【００７３】

ＣＰＵおよびメモリ４から出力される画面データは、ディスプレイ２に送られて表示される。

【００７４】

［ＳＩによる基本フローの作成］

本実施の形態の開発支援装置では、ＳＩが、操作入力部７に対して適宜操作を行なうことにより、基本フロー開発部９は、ディスプレイ２に、図４に示されるような画面（基本フロー開発用画面）を表示する。

【００７５】

図４を参照して、基本フロー開発用画面は、画像処理モジュールのリストが表示されるリスト欄８３と、基本フローにおける画像処理モジュールの順序を示す基本フロー欄９５を含む。画像処理モジュール記憶部２９１では、各画像処理モジュールは用途に応じた種別ごとに分類されて記憶されている。ここでの種別には、「検査・計測する」「画像を取り込む」「画像を補正する」「検査・計測を補助する」を含む。そして、リスト欄８３では、各画像処理モジュールの名称が、これらの種別に従って表示されている。

【００７６】

以下、基本フロー開発用画面を用いた基本フローの作成について説明する。

ＳＩが、マウス２０を操作してリスト欄８３の一つの項目を選択すると、基本フロー開発部９は、リスト欄８３の中の１つの画像処理モジュールを選択する。

【００７７】

ＳＩが、マウス２０を操作して追加ボタン８４を選択すると、基本フロー開発部９は、基本フロー欄９５の最下部にリスト欄８３で選択されている画像処理モジュールを追加する。

【００７８】

ＳＩが、マウス２０を操作して挿入ボタン８５を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択されている画像処理モジュールの直下にリスト欄８３で選択されている画像処理モジュールを挿入する。

【００７９】

ＳＩが、マウス２０を操作して設定ボタン８６を選択すると、画像処理モジュール開発部１０は、基本フローの中の選択されている画像処理モジュール（の処理モジュールＭＳ）のパラメータなどを設定するためのＵＩプログラムを画像処理モジュール記憶部２９１から読み出して表示する。ＳＩが、表示されている画面に含まれるメニューのうちのツー

ルから部品を選択し、部品記憶部１５から選択された部品を読み出して配置することによって、ユーザインタフェース画面が編集され、図示しない保存メニューを選択して、編集後のＵＩプログラムが画像処理モジュール記憶部２９１に記憶される。

【００８０】

ＳＩが、マウス２０を操作してコピーボタン９０を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択されている画像処理モジュールをコピーする。

【００８１】

ＳＩが、マウス２０を操作してペーストボタン９１を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択されている画像処理モジュールの直下に、コピーボタン９０によってコピーした画像処理モジュールを挿入する。

【００８２】

ＳＩが、マウス２０を操作して削除ボタン９２を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択されている画像処理モジュールを削除する。

【００８３】

ＳＩが、マウス２０を操作して移動（上へ）ボタン８８を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択される画像処理モジュールを１つ上に移動する。

【００８４】

ＳＩが、マウス２０を操作して移動（下へ）ボタン８９を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択される画像処理モジュールを１つ下に移動する。

【００８５】

ＳＩが、マウス２０を操作して名前の変更ボタン８７を選択すると、基本フロー開発部９は、基本フロー欄９５の中の選択されている画像処理モジュールの名前の変更を可能とする。ＳＩは、キーボード２１を操作することにより、画像処理モジュールの名前を入力することができる。

【００８６】

ＳＩが、マウス２０を操作してヘルプ９６を選択すると、基本フロー開発部９は、マニュアル類記憶部１８からヘルプを読み出して表示するる。

【００８７】

ＳＩが、マウス２０を操作して閉じる９７を選択すると、基本フロー開発部９は、基本フローが作成されている場合に、作成された基本フローを基本フロー記憶部２９４に記憶し、その処理を終了する。

【００８８】

図５は、基本フローの作成後の画面の例を表わす図である。

図５を参照して、基本フローは、３つの画像処理モジュール９８、９９、４０からなる。この基本フローでは、まずカメラ画像を入力し、次に入力した画像の面積重心を計算して、その後画像処理モジュールのデータを設定することを定めた基本フローが作成されている。

【００８９】

［ＳＩによる画像処理モジュールの作成］

次に、本実施の形態の開発支援装置における、ＳＩがテンプレート記憶部１６に記憶されたテンプレートを用いて画像処理モジュールを作成する処理について説明する。

【００９０】

図６は、ＳＩが画像処理モジュールの作成を行なう際にディスプレイ２に表示される画面の一例を示す図である。図７は、画像処理モジュール開発の処理手順を表わすフローチャートである。

【００９１】

ＳＩが操作入力部７を適宜操作することにより画像処理モジュールを作成するプログラムを起動させると、ディスプレイ２には図６に示された画面が表示される。そして、さらに図７を参照して、ＳＩが、当該画像に対して、マウス２０を操作してボタン６５を選択したときには（ステップＳ４０１でＹＥＳ）、画像処理モジュール開発部１０は、ＭＳプ

ログラムの編集処理を選択し、ＭＳプログラムのテンプレート４１をテンプレート記憶部１６から読み出して、図８に示すようにエディタ上に表示する。ＳＩが、キーボートから文字を入力、削除することによって、ＭＳプログラムのテンプレート４１を編集して、図示しない保存メニューを選択して、編集後のＭＳプログラムを画像処理モジュール記憶部２９１に記憶する（ステップＳ４０２）。

【００９２】

ＳＩが、マウス２０を操作してボタン６６を選択したときには（ステップＳ４０３でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８からＭＳプログラムの編集を説明したマニュアル３－２－５章を読み出して表示する（ステップＳ４０４）。

【００９３】

ＳＩが、マウス２０を操作してボタン６７を選択したときには（ステップＳ４０５でＹＥＳ）、画像処理モジュール開発部１０は、ＵＩプログラムの編集処理を選択して、ＵＩプログラムのテンプレートをテンプレート記憶部１６から読み出して図９に示すようなユーザインタフェース画面のテンプレート４６１を表示する。ＳＩが、表示されている画面に含まれるメニューのうちのツール４６２から部品を選択し、部品記憶部１５から選択された部品を読み出して配置することによって、図９のユーザインタフェース画面のテンプレート４６１を編集し、図示しない保存メニューを選択して、編集後のＵＩプログラムを画像処理モジュール記憶部２９１に記憶する（ステップＳ４０６）。

【００９４】

ＳＩが、マウス２０を操作してボタン６８を選択したときには（ステップＳ４０７でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８からＵＩプログラムの編集を説明したマニュアル３－２－６章を読み出して表示する（ステップＳ４０８）。

【００９５】

ＳＩが、マウス２０を操作してボタン６９を選択したときには（ステップＳ４０９でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８からユーザインタフェース画面のサンプルを読み出して表示する（ステップＳ４１０）。

【００９６】

ＳＩが、マウス２０を操作してボタン７０を選択したときには（ステップＳ４１１でＹＥＳ）、実行制御部８は、シミュレーションソフト記憶部１７からシミュレーションソフトを読み出して起動する（ステップＳ４１２）。

【００９７】

ＳＩが、マウス２０を操作してボタン７１を選択したときには（ステップＳ４１３でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８から画像処理モジュール開発のチュートリアルを読み出して表示する（ステップＳ４１４）。

【００９８】

ＳＩが、マウス２０を操作してボタン７２を選択したときには（ステップＳ４１５でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８から画像処理モジュール開発を説明したマニュアル３－２章を読み出して表示する（ステップＳ４１６）。

【００９９】

ＳＩが、マウス２０を操作してボタン７３を選択したときには（ステップＳ４１７でＹＥＳ）、画像処理モジュール開発部１０は、マニュアル類記憶部１８から画像処理モジュール開発のためのリファレンスを読み出して表示する（ステップＳ４１８）。

【０１００】

ＳＩが、マウス２０を操作して図示しないビルドメニューを選択したときには（ステップＳ４１９でＹＥＳ）、ビルド実行部１３は、画像処理モジュール記憶部２９１に記憶されているステップＳ４０２で作成されたＭＳのソースコードおよびステップＳ４０６で作成されたＵＩのソースコードをコンパイルしてオブジェクトコードを生成し、ライブラリ類記憶部２３から必要なライブラリとドライバを読み出してオブジェクトコードとリンクすることによって、ＷｉｎｄｏｗｓＣＥとＷｉｎｄｏｗｓＸＰ用のＭＳの実行形式のプログラムおよびＵＩの実行形式のプログラムを作成して実行プログラム記憶部１９に記憶す

る（ステップＳ４２０）。
【０１０１】
ＳＩが、マウス２０を操作して図示しない「戻る」ボタンを選択したときには（ステップＳ４２１でＹＥＳ）、画像処理モジュール開発部１０は、処理を終了する。
【０１０２】
次に、テンプレートがＳＩの編集によってどのように変更されるかの例を説明する。ここでは、テンプレートを編集して、他の画像処理モジュールのパラメータ設定を一括して行なうための画像処理モジュールを作成する場合について説明する。この画像処理モジュールは、座標の変更や上下限値の変更などのように、複数の数値を変更する場合に用いられる。
【０１０３】
図１０および図１１は、「ItemDefs.h」のテンプレートを表わす図である。
図１２および図１３は、編集後の「ItemDefs.h」を表わす図である。
【０１０４】
図１０～図１３を参照して、ＳＩの編集によって、テンプレートの定義文１９０が、定義文１９１に変更され、定義文１９２、１９３、１９４が追加されている。
【０１０５】
図１０～図１３を参照して、「ItemDefs.h」の内容を説明する。SETUPDATA構造体には、処理を実行する際に必要になるパラメータなどの設定データの構造が定義される。MEASUREDATA構造体には、処理（計測）を実行した際の結果のデータの構造が定義される。ItemDef.h中で定義した SETUPDATA構造体に含まれるデータの初期値については、AssignProc.cpp中に記述される。
【０１０６】
図１４は、「AssignProc.cpp」のテンプレートを表わす図である。
図１５および図１６は、編集後の「AssignProc.cpp」を表わす図である。
【０１０７】
図１４～図１６を参照して、ＳＩの編集によって、日付１５１が記入され、作成者１５２が記入され、関数AssignProcの中身の命令１５３が追加されている。
【０１０８】
図１４～図１６を参照して、「AssignProc.cpp」の内容を説明する。本ファイル内では以下の関数の処理を定義し、画像処理モジュールが登録される際のチェック処理などを記述する。
【０１０９】
int クラス名::AssignProc(*ptrProcUnit);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。NORMAL以外の値を返した場合は処理ユニット登録不可ということで、処理ユニットの登録が行われない。
【０１１０】
NORMAL（０）：処理ユニット登録可能
!NORMAL（０以外）：処理ユニット登録不可
設定データの初期化は、このメソッドに記述し、図形データの初期化(SetFigureType())は、このメソッドに記述する。本関数を用いることで、処理ユニットの登録順序などに制約がある場合のチェックを行なうことができる。
【０１１１】
図１７および図１８は、「FigureUpdate.cpp」のテンプレートを表わす図である。
本発明の実施形態では、ＳＩが、テンプレートを編集しなかったため、「FigureUpdate.cpp」は、変更されていない。
【０１１２】
図１７および図１８を参照して、「FigureUpdate.cpp」の内容を説明する。本ファイル内では以下のメソッドを記述し、図形データが更新されたときの処理を記述する。

【０１１３】
int クラス名::FigureUpdate(*ptrProcUnit, figureNo);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
int figureNo; 図形データ番号
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１１４】
NORMAL（０）：図形データ更新処理に成功
!NORMAL（０以外）：図形データ更新処理に失敗
モデルの登録処理は本メソッドに記述する。
【０１１５】
図１９および図２０は、「ItemInit.cpp」のテンプレートを表わす図である。
図２１および図２２は、編集後の「ItemInit.cpp」を表わす図である。
【０１１６】
図１９～図２２を参照して、ＳＩの編集によって、日付１５４が記入され、作成者１５５が記入されている。さらに、ＳＩの編集によって、テンプレートの"Sample"６５８、ITEM_MEASURE６５９、一般計測関連６６０、０（６６１）、"サンプル"６６２が、"SetUnitData4"１５８、ITEM_SUPPORT１５９、計測補助関連１６０、４（１６１）、"処理ユニットデータ設定"１６２に変更されている。
【０１１７】
図１９～図２２を参照して、「ItemInit.cpp」の内容を説明する。コンストラクタ・デストラクタおよび本クラスのインスタンスを生成する関数、本クラスのインスタンスを削除する関数を記述する。なお、本クラスのインスタンスを生成する関数および本クラスのインスタンスを削除する関数は、DLLのエクスポート関数にする。コンストラクタには画像処理モジュールをシステム（CORERA）に認識させるための情報を記述する。具体的には下記のメソッドに値を設定する。
【０１１８】
this->itemIdent = _T("Sample"); (A) 画像処理モジュール識別名
this->maker = _T("OMRON AST3"); (B) 画像処理モジュール製作者名
this->version = 100; (C) バージョン番号（×100）
this->itemKind = ITEM_MEASURE; (D) 画像処理モジュール種別（＝一般計測関連）
this->setupDataSize = sizeof(SETUPDATA); (E)設定データ構造体サイズ
this->measureDataSize = sizeof(MEASUREDATA); (F) 計測データ構造体サイズ
this->modelDataCount = 0; (G) モデルデータ最大個数
this->imageDataCount = 0; (H) 画像データ最大個数
this->innerUnitCount = 0;(I) 内包処理ユニット最大個数
this->figureDataCount = 0;(J) 図形データ最大個数
this->title = _T("サンプル"); (K) 画像処理モジュールタイトル名
デストラクタには、本クラスのインスタンスが消滅するときに実行する処理を記述する。本関数では、本クラスのインスタンスを生成し戻り値として返す処理を記述する。本関数の処理を終了する際は、戻り値として以下の値を返す。なお、図２０中の記述９９１Ｎについては後述する。
【０１１９】
ProcItem：インスタンスのポインタ：インスタンス生成成功時
NULL：インスタンス生成失敗時
図２３および図２４は、「MeasureDisp.cpp」のテンプレートを表わす図である。
【０１２０】
図２５～図２７は、編集後の「MeasureDisp.cpp」を表わす図である。
図２３および図２４および図２５～図２７を参照して、ＳＩの編集によって、日付１６３および日付１６４が記入され、作成者１６５および作成者１６６が記入され、関数MeasureDispTの中身の命令１６７が追加されている。

【０１２１】
本ファイル内では以下のメソッドを記述し、画像処理モジュール個別に計測結果を表示する際の表示処理を記述する。
【０１２２】
int クラス名::MeasureDispI(*ptrProcUnit, subNo, *ptrImageWindow);
void クラス名::MeasureDispG(*ptrProcUnit, subNo, *ptrImageWindow);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
int subNo; 表示種別の番号
ImageWindow *ptrImageWindow; 画像表示エリア情報へのポインタ
void クラス名::MeasureDispT(*ptrProcUnit, subNo, *ptrTextWindow);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
int subNo; 表示種別の番号
TextWindow *ptrTextWindow; 詳細結果表示エリア情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１２３】
NORMAL（０）：計測結果出力処理正常終了
!NORMAL（０以外）：計測結果出力処理異常終了
３つの関数は、以下の記述に対応する。
【０１２４】
MeasureDispI()：画像表示
MeasureDispG()：画像表示エリアへのグラフィック表示
MeasureDispT()：詳細結果表示エリアへの文字表示
引数subNoが-1の場合は、全画像処理モジュール共通で位置一覧表示モード時の表示となる。サーチ系の画像処理モジュールの場合は、モデル領域のみ表示する。処理ユニットの判定結果が”判定なし”の場合（MEASDATA 構造体中のjudge=J_NCの場合）、処理ユニットのMeasureDisp*()はコールされない。
【０１２５】
図２８および図２９は、「MeasureInit.cpp」のテンプレートを表わす図である。
本発明の実施形態では、ＳＩが、テンプレートを編集しなかったため、「MeasureInit.cpp」が変更されていない。
【０１２６】
図２８および図２９を参照して、「MeasureInit.cpp」の内容を説明する。本ファイル内では、以下の関数の処理を定義し、計測画面に入る時点で実行すべき初期化処理および計測終了時に実行すべき処理を記述する。
【０１２７】
int クラス名::MeasureInit(*ptrProcUnit);
int クラス名::MeasureEnd(*ptrProcUnit);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１２８】
NORMAL（０）：計測初期化処理正常終了
!NORMAL（０以外）：計測初期化処理異常終了
NORMAL以外の値を返した場合は画面にエラーメッセージが表示され、当該シーン内の全処理ユニットの非セーブＲＯＩモデルがクリアされる。また、NORMAL以外の値を返して画面にエラーメッセージが表示された場合も、メッセージボックスを終了するとそのまま計測が実行可能な状態になる。
【０１２９】
図３０は、「MeasureOut.cpp」のテンプレートを表わす図である。
本発明の実施形態では、ＳＩが、テンプレートを編集しなかったため、「MeasureOut.cpp」が変更されていない。

【０１３０】
図３０を参照して、「MeasureOut.cpp」の内容を説明する。本ファイル内では以下のメソッドを記述し、画像処理モジュール個別に計測結果を出力する際の結果出力処理を記述する。
【０１３１】
int クラス名::MeasureOut(*ptrProcUnit);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１３２】
NORMAL（０）：計測結果出力処理正常終了
!NORMAL（０以外）：計測結果出力処理異常終了
図３１は、「MeasureProc.cpp」のテンプレートを表わす図である。
【０１３３】
図３２～図３４は、編集後の「MeasureProc.cpp」を表わす図である。
図３１および図３２～図３４を参照して、ＳＩの編集によって、日付１６８が記入され、作成者１６９が記入され、関数MeasureProcの中身の命令１７０が追加されている。
【０１３４】
図３１および図３２～図３４を参照して、「MeasureProc.cpp」の内容を説明する。本ファイル内では以下のメソッドを記述し、計測時に実行する画像処理などの処理を記述する。
【０１３５】
int クラス名::MeasureProc(*ptrProcUnit);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１３６】
NORMAL（０）：計測処理正常終了
!NORMAL（０以外）：計測処理異常終了
図３５および図３６は、「RenumProc.cpp」のテンプレートを表わす図である。
【０１３７】
図３７および図３８は、編集後の「RenumProc.cpp」を表わす図である。
図３５～図３８を参照して、ＳＩの編集によって、日付１７１が記入され、作成者１７２が記入され、関数RenumProcの中身の命令１７３が追加されている。
【０１３８】
図３５～図３８を参照して、「RenumProc.cpp」の内容を説明する。本ファイル内では以下のメソッドを記述し、ユニット追加・削除・移動時の処理を記述する。
【０１３９】
int クラス名::RenumProc(*ptrProcUnit, *ptrRenumInfo);
ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
RenumInfo *ptrRenumInfo;フロー編集情報へのポインタ
本関数の処理を終了する際は、戻り値として以下の値を返す。
【０１４０】
NORMAL（０）：ユニット追加・削除・移動時処理正常終了
!NORMAL（０以外）：ユニット追加・削除・移動時処理異常終了
図３９～図４１は、「UnitData.cpp」のテンプレートを表わす図である。
【０１４１】
図４２～図５０は、編集後の「UnitData.cpp」を表わす図である。
図３９～図４１および図４２～図５０を参照して、ＳＩの編集によって、マクロ定義に関する定義文１７４が追加され、静的変数に関する定義文１７５が追加され、関数参照に関する定義文１７６が追加されている。関数SetUnitData(ptrProcUnit, dataNo, data)に関して、日付１７７が記入され、作成者１７８が記入され、中身の命令１７９が追加され

ている。関数SetUnitData(ptrProcUnit, dataIdent, data)に関して、日付１８０が記入され、作成者１８１が記入され、中身の命令１８２が追加されている。関数GetUnitData(ptrProcUnit, dataNo, data)に関して、日付１８３が記入され、作成者１８４が記入され、中身の命令１８５が追加されている。関数GetUnitData(ptrProcUnit, dataIdent, data)に関して、日付１８６が記入され、作成者１８７が記入され、中身の命令１８８が追加されている。さらに、関数setDataProcおよび関数getDataProcが追加されている。

【０１４２】

図３９～図４１および図４２～図５０を参照して、「UnitData.cpp」の内容を説明する。

【０１４３】

本ファイル内では以下のメソッドを記述し、データ設定時およびデータ取得時の処理を記述する。

【０１４４】

```
int クラス名::SetUnitData(*ptrProcUnit, dataNo, *data);
  ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
  int datano; 設定するデータのデータ番号
  ANYTYPE *data; 設定するデータ
int クラス名::SetUnitData(*ptrProcUnit, dataIdent, *data);
  ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
  TCHAR *dataIdent; 設定するデータの識別文字列へのポインタ
  ANYTYPE *data; 設定するデータ
int クラス名::GetUnitData(*ptrProcUnit, dataNo, *data);
  ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
  int datano; 取得するデータのデータ番号
  ANYTYPE *data; 取得するデータ
int クラス名::GetUnitData(*ptrProcUnit, dataIdent, *data);
  ProcUnit *ptrProcUnit; 処理ユニット情報へのポインタ
  TCHAR *dataIdent; 取得するデータの識別文字列へのポインタ
  ANYTYPE *data; 取得するデータ
ANYTYPE データタイプ構造体定義
  typedef struct {
    int type;
    int ival;
    double dval;
    TCHAR *string;
    struct {
      void *addr;
      int size;
    } variant;
  } ANYTYPE;
```

上記のtypeには以下のようにどういう型のデータを渡すかというデータ種別（を示すコード）が入れるものとする。

【０１４５】

T_INTEGER：整数
T_DOUBLE ：実数
T_STRING ：文字列
T_VARIANT：可変長データ
本関数の処理を終了する際は、戻り値として以下の値を返す。

【０１４６】

NORMAL（０）：データ設定・データ設定成功

!NORMAL（０以外）：データ取得・データ取得成功

［開発支援装置における画像処理ユニットの作成］

次に、本実施の形態の開発支援装置における、ＳＩによって編集されるための画像処理モジュール（処理モジュールＭＳの素材と処理モジュールＵＩ）を作成する処理について説明する。

【０１４７】

本実施の形態の開発支援装置では、画像処理モジュール作成部１０Ａは、テンプレート記憶部１６に記憶されたテンプレートを修正（主に、データを追加）することにより、ＳＩがカスタマイズできるような態様で画像処理モジュールを作成して画像処理モジュール記憶部２９１に記憶させる。ここで、画像処理モジュール作成部１０Ａが画像処理モジュールを作成する処理（画像処理モジュール作成処理）について、当該処理のフローチャートである図５１を参照して説明する。

【０１４８】

画像処理モジュール作成処理では、画像処理モジュール作成部１０Ａは、まずステップＳ１０で、処理モジュール情報の入力を受付ける処理を実行して、ステップＳ２０へ処理を進める。

【０１４９】

なお、画像処理モジュール作成部１０Ａは、ステップＳ１０において、図５２に示すような画面を表示させる。これから作成しようとする画像処理モジュールを識別するための情報である処理モジュール識別名を入力するための入力欄１０１、当該画像処理モジュールの日本語名称を入力するための入力欄１０２および英語名称を入力するための入力欄１０３、ならびに当該画像処理モジュールを構成するデータのハードディスク５における保存場所を入力するための入力欄１０４が表示されている。入力欄１０１～１０４に対するデータの入力は、当該画面上のボタン１０６に対して操作がなされることにより確定される。入力欄１０１でのデータの入力が確定されると、画像処理モジュール作成部１０Ａは、図３を参照して説明したような画像処理モジュールのテンプレートを構成するファイルについてのファイルおよびフォルダの名称を決定する。つまり、図５２に示されたようなデータの入力が確定された場合、日本語のメッセージファイルのファイル名は、「サーチ２_ipn.msg」とされ、英語のメッセージファイルのファイル名は「search2_eng.msg」とされる。

【０１５０】

また、これから作成しようとする画像処理モジュールならびに処理モジュールＵＩおよび処理モジュールＭＳのディレクトリ構造であって、後述するステップＳ７０で作成されるディレクトリ構造が、図５３に示されている。図５３に示されたディレクトリ構成における三箇所の「search2」は、入力欄１０１に入力されたテキスト情報に対応している。

【０１５１】

また、画像処理モジュール作成部１０Ａは、図５２の入力欄１０１に入力が確定されたデータに基づいて、「ItemInit.cpp」のテンプレートを一部書換える。具体的には、画像処理モジュール作成部１０Ａは、図２０の記述９９１Ｎを、図５４の記述９９１Ｘに書換える。図２０と図５４を参照して、画像処理モジュール作成部１０Ａは、入力欄１０１に入力を確定された情報に基づいて、処理項目識別名６５８，６０１Ｘおよび処理項目タイトル名６６２，６０２Ｘの記述を変更させている。

【０１５２】

次に、画像処理モジュール作成部１０Ａは、ステップＳ２０で、設定画面パターンを選択する処理を実行して、ステップＳ３０へ処理を進める。この処理は、処理モジュールＭＳによる画像処理に対して、ＳＩがどのような情報を設定可能とするかを選択する処理に相当する。具体的には、この処理では、ディスプレイ２に図５５に示すような画面が表示される。当該画面では、ＳＩが処理モジュールＭＳの処理内容を設定する際に表示させる画面として処理モジュールＵＩに含める、表示可能な画面についての情報を登録するための画面である。処理モジュールＭＳの処理内容を設定するための画面には、図形設定画面

、ＲＧＢ色指定画面、出力パラメータ設定画面、座標指定画面、およびＨＳＶ色指定画面の５種類の画面が含まれる。図５５の画面に表示されている入力欄１０９～１１３は、この５種類の画面のそれぞれについて、処理モジュールＵＩに含めるか否かを設定するために表示されている。入力欄１０９～１１３に入力された、各画面を処理モジュールＵＩに含ませるか否かの情報は、ボタン１１５を操作されることにより確定される。

【０１５３】

図５６は、図形設定画面の一例を示す図である。この画面では、画像処理検査装置において入力された画像の中の、処理対象とする領域の形状、位置、および大きさを設定するための画面である。当該画面において、入力欄１０９Ａは、処理対象の領域の形状を設定するためのものである。また、入力欄１０９Ｂは、入力欄１０９Ａにおいて指定された形状の、位置および大きさを設定するためのものである。当該画面では、表示欄１０９Ｄに、画像処理検査装置に入力される画像（または、そのサンプル画像）が表示される。また、入力欄１０９Ａ，１０９Ｂに入力された情報に対応する処理対象領域も、表示欄１０９Ｄに表示される。ＳＩは、表示欄１０９Ｄの表示を参照しながら、入力欄１０９Ａおよび入力欄１０９Ｂに対して入力する情報を調整できる。入力欄１０９Ａ，１０９Ｂに入力された情報は、ボタン１０９Ｃに対して操作がなされることにより確定される。

【０１５４】

図５７に、ＲＧＢ色指定画面の一例を示す。

この画面では、表示欄１１０Ｃに、画像処理検査装置に入力された画像（または、そのサンプル画像）が表示される。そして、入力欄１１０Ａには、表示欄１１０Ｃに表示された画像に対してエッジ検出を行なうための条件となる色情報が入力される。入力欄１１０Ａに入力された情報は、ボタン１１０Ｂに対して操作がなされることにより確定される。

【０１５５】

図５８は、出力パラメータ設定画面の一例を示す図である。この画面では、表示欄１１１Ｅに、画像処理検査装置に入力される画像（または、そのサンプル画像）が表示される。この画面は、表示欄１１１Ｅに表示される画像とモデル画像（たとえば、予め画像処理検査装置に記憶されている）との間で、ターゲット画像の位置にずれが生じた場合の対処に関する設定を行なう画面である。具体的には、入力欄１１１Ａには、ターゲットの座標として表示する座標を、位置ずれの修正を行なう前の座標とするか修正後の座標とするかを設定するためのものである。入力欄１１１Ｂは、表示欄１１１Ｅに表示させる画像を、位置ずれについての修正を行なったものとするかそのような修正を行なわないものとするかを設定するためのものである。入力欄１１１Ｃは、現在作成の対象となっている画像処理モジュールによる画像処理の結果として出力する判定内容を、同じ基本フローに含まれる他の画像処理モジュールによる判定結果も含めた総合的なものとするか、作成中の画像処理モジュール単独のものとするかを設定するためのものである。入力欄１１１Ａ～１１１Ｃに入力した情報は、ボタン１１１Ｄに対して操作がなされることにより確定される。

【０１５６】

図５９は、座標指定画面の一例を示す図である。図５９では、表示欄１１２Ｄに、画像処理検査装置に入力された画像（または、そのサンプル画像）が表示される。入力欄１１２Ｃには、図形設定画面（図５６参照）で指定された領域における、ＨＳＶ色指定画面（後述する図６０参照）において指定された色の条件を満たす面積が示される。また、入力欄１１２Ａには、そのような領域の重心なる座標が表示されている。入力欄１１２Ｃまたは入力欄１１２Ａに表示された値を変更することにより、ＳＩは、後述するＨＳＶ色指定画面に対する設定情報（色についての条件に関する情報）を、入力欄１１２Ｃおよび１１２Ａに対応するものに変更することができる。入力欄１１２Ｃおよび入力欄１１２Ａに入力された情報は、ボタン１１２Ｂを操作されることにより確定される。

【０１５７】

図６０は、ＨＳＶ色指定画面の一例を示す図である。

図６０に示された画面は、画像処理検査装置に入力された画像に対する処理の条件として色情報を指定するための画面である。ＳＩは、入力欄１１３Ａに対して、色（色相、彩

度、および明度）を指定する情報を入力する。この画面では、表示欄１１３Ｃに、画像処理検査装置に入力される画像（または、そのサンプル画像）が表示される。そして、表示欄１１３Ｃに表示される画像は、入力欄１１０Ａに対して入力された情報に基づいて、適宜処理を施される。たとえば、入力欄１１０Ａに特定の色を指定する情報が入力された場合、表示欄１１３Ｃでは、画像処理検査装置に入力された画像（または、そのサンプル画像）の中の、当該特定の色を有する画素のみの表示が行なわれる。

【０１５８】

入力欄１１３Ａでは、指定された色に基づいた画像処理に対する許容範囲を入力することもできる。入力欄１１３Ａに入力された情報は、ボタン１１３Ｂに対して操作がなされることにより確定される。

【０１５９】

以上説明したように、ステップＳ２０の設定画面パターン選択処理では、処理モジュールＭＳに対してＳＩが設定を行なう際に表示される画面が選択される。図５５～図６０を参照して説明した画面を表示させるための情報は、モジュール作成用データ記憶部２５に記憶されている。また、図５６～図６０を参照して説明した画面の中で、入力欄１０９～１１３（図５５参照）の中の設定を確定された入力欄に対応する画面については、当該画面を表示させるための情報が、画像処理モジュール記憶部２９１に、処理モジュールＵＩに含まれるように、記憶されるようになる。

【０１６０】

図５１に戻って、ステップＳ２０の処理の後、画像処理モジュール作成部１０Ａは、ステップＳ３０で、処理モジュールＭＳに組込む機能の選択を行なう。この処理では、画像処理モジュール作成部１０Ａは、ディスプレイ２に図６１に示すような画面を表示させる。図６１を参照して、この画面には、図３を参照して説明した５個の任意ファイル（「FigureData.cpp」「SetupData.cpp」「RenumProc.cpp」「ThroughProc.cpp」「SaveLoad.cpp」）のそれぞれに対応する入力欄１１６～１２０が表示されている。

【０１６１】

図６１に示された画面は、画像処理モジュールの処理モジュールＭＳに、これらの５個の任意ファイルのそれぞれを含めるか否かを設定するための画面である。入力欄１１６～１２０に入力された情報は、ボタン１２１に対して操作がなされることにより確定される。そして、この画面に入力された情報が確定されることにより、処理モジュールＭＳに、上記５種類の任意ファイルの中で、確定された内容に対応するファイルが組込まれる。

【０１６２】

図５１に戻って、ステップＳ３０の処理の後、画像処理モジュール作成部１０Ａは、ステップＳ４０で、バージョンや図形数などの、画像処理モジュールの基本的な情報の入力を受付ける処理を実行する。

【０１６３】

この処理では、画像処理モジュール作成部１０Ａは、ディスプレイ２に図６２に示したような画面を表示させる。図６２を参照して、当該画面には、これから作成する画像処理モジュールの製作者の名称を入力するための入力欄１２２、当該画像処理モジュールのバージョンを入力するための入力欄１２３、その種別（図４を参照して説明した「検査・計測をする」「画像を取込む」などに対応）を入力するための入力欄１２４、図形設定画面（図５６参照）の入力欄１０９Ａにおいて設定することが可能な図形の最大個数を入力（設定）するための入力欄１２５、表示欄１０９Ｄ（図５６参照）などに表示される画像の比較対象とされるデータ（モデルデータ）として参照されるデータの最大個数を入力するための入力欄１２６、表示欄１０９Ｄ（図５６参照）などに表示させる画像データの最大個数を入力（設定）するための入力欄１２７、内包処理ユニット（後述するように、これから作成する画像処理ユニット機能を取込む他の画像処理ユニット）の最大個数を入力（設定）するための入力欄１２８を含む。

【０１６４】

入力欄１２２～１２８に入力された情報は、ボタン１３０に対して操作がなされること

により確定される。なお、入力欄１２４に対する種別の入力は、画像処理モジュールの種類として予め定められた複数の種別の中からプロダウンメニュー等により一の種別を選択することにより行なわれる。

【０１６５】

図６２に示された画面に対する情報の入力が確定されると、これから作成される画像処理モジュールの処理モジュールＭＳにおいて、「ItemInit.cpp」のファイルが、テンプレート記憶部１６に記憶されていた状態から一部書換えられる。具体的には、テンプレート記憶部１６に記憶されている「ItemInit.cpp」の一部である、図２７中の部分９９１Ｎが、図６３に示される部分９９１Ｘに書換えられる。

【０１６６】

図２０と図６３とを参照して、上記したように図６２に示されたような情報の入力が確定されると、処理項目製作者名は、入力欄１２２に入力された情報に対応するように、処理項目製作者名６０３Ｘとされる。また、バージョン番号は、入力欄１２３に入力された情報に対応するように、バージョン番号６０４Ｘとされる。また、処理項目種別は、入力欄１２４に入力された情報に対応して、処理項目種別６０５Ｘとされる。また、図２０の図形データ最大個数６６３とモデルデータ最大個数６６１と画像データ最大個数６６４と内包処理ユニット最大個数６６５は、それぞれ、入力欄１２５～１２８に入力された情報に対応して、図形データ最大個数６０６Ｘとモデルデータ最大個数６０７Ｘと画像データ最大個数６０８Ｘと内包処理ユニット最大個数６０９Ｘにされる。

【０１６７】

図５１に戻って、ステップＳ４０の処理の後、画像処理モジュール作成部１０Ａは、ステップＳ５０で、図形の種類の選択を受付ける処理を行なう。

【０１６８】

この処理では、画像処理モジュール作成部１０Ａは、図６４に示された画面をディスプレイ２に表示される。図６４を参照して、当該画面では、入力欄１３１には、「長方形」「直線」「楕円」「円周」「円弧」「多角形」という、図形の種別が示されている。そして、入力欄１３１では、これらの種別の中で、入力欄１０９Ａ（図５６参照）においてＳＩが選択することができる種別として設定する図形の種別を選択する情報が入力される。また、入力欄１３２には、図形設定画面（図５６参照）において入力欄１３１で選択された各図形が選択された際にデフォルトとして表示されるときの位置および大きさ（座標）が入力される。入力欄１３１および入力欄１３２に入力された情報は、ボタン１３４に対して操作がなされることにより確定される。

【０１６９】

図６４に示された画面に対して入力された情報が確定されると、これから作成する画像処理モジュールの処理モジュールＭＳにおいて、「AssignProc.cpp」のデータを、当該画面に入力された情報に基づいて一部書換える。具体的には、図１４に示された「AssignProc.cpp」の中の記述９９２Ｎを、図６５に示された記述９９２Ｘへと書換える。記述９９２Ｘには、入力欄１３１において選択された種別に対応する記述６１０Ｘと、当該種別に対応して入力欄１３２に入力された座標に対応する記述６１１Ｘが含まれる。

【０１７０】

図５１に戻って、ステップＳ５０の処理の後、画像処理モジュール作成部１０Ｘは、ステップＳ６０で、機能を小計する画像処理モジュールの選択を受付ける処理を実行する。

【０１７１】

この処理では、画像処理モジュール作成部１０Ａは、図６６に示すような画面をディスプレイ２に表示させる。図６６を参照して、当該画面には、画像処理モジュール記憶部２９１に記憶される他の画像処理モジュールであって、これから作成する画像処理モジュールに機能を取込む画像処理モジュールを特定する情報を入力するための入力欄１３５が表示されている。図６６では、入力欄１３５には、５個の画像処理モジュールについての情報が入力される状態が示されている。この画面において情報を入力することができる画像処理モジュールの個数は、入力欄１２８（図６２参照）に入力を確定された数値に対応し

ている。入力欄１３５に入力された情報は、ボタン１３６に対して操作がなされることにより確定される。入力欄１３５に入力された情報が確定されると、図２１に示された「AssignProc.cpp」の中の記述９９２Ｎは、これから作成される画像処理モジュールの「AssignProc.cpp」では、図６７に示された記述９９２Ｙに書換えられる。なお、図６５に示された記述９９２Ｘに既に書換えられている場合には、この記述９９２Ｘに対して、記述９９２Ｙの中の記述６１２Ｘが追加される。

【０１７２】

図５１に戻って、ステップＳ６０の処理の後、画像処理モジュール作成部１０Ａは、ステップＳ７０で、以上説明したようにテンプレート記憶部１６に記憶されたテンプレートの中の一部の記述を書換えたものを新たな画像処理モジュールとして画像処理モジュール記憶部２９１に記憶させるために、ディレクトリの作成およびファイルの作成を行なって、ステップＳ８０へ処理を進める。

【０１７３】

なお、ここで作成されるファイルには、変数定義用のファイルが含まれる。ここで、変数定義用ファイルの作成について説明する。

【０１７４】

変数定義用ファイルを作成する際、画像処理モジュール作成部１０Ａは、ディスプレイ２に図６８に示すような画面を表示させる。

【０１７５】

図６８に示された画面は、画像処理モジュールの作成の際に（または、これに先立って）、ＣＳＶ（Comma Separated Value）形式で作成された変数定義用のファイルの保存場所を入力するための画面である。入力欄１３７には、当該ファイルの保存場所が入力され、ボタン１３８に対して操作がなされることにより当該保存場所に保存されたＣＳＶ形式のデータから、変数定義用のヘッダファイルが作成されるとともに、「AssignProc.cpp」などのデータの初期化が行なわれる。

【０１７６】

ＣＳＶ形式の変数定義用のファイルの一例を図７０に示す。このようなＣＳＶ形式のファイルは、たとえば、図６９に示されるような表形式のデータが開発支援装置１に対して入力されることにより、周知の技術により作成される。そして、ステップＳ８０では、画像処理モジュール作成部１０Ａは、図７０に示されるようなファイルを読込むことにより、当該ファイルを、図７１に示されるようなヘッダ情報のファイル（Itemdefs.h）に変換して、画像処理モジュール記憶部２９１に記憶する。

【０１７７】

以上説明した変数定義用のファイルには、図６９等から理解されるように、各変数についての名称（変数名）、形式（型）、識別子、初期値、下限値、上限値、および、外部の出力および／または入力される当該変数の値を参照するか否か（外部参照）についての情報を含む。本実施の形態の開発支援装置では、このような変数の定義に関する情報を、ファイルとして、一括して保持している。

【０１７８】

また、上記したデータの初期化により、「AssignProc.cpp」には図７２に示した記述が追加されるとともに、「UnitData.cpp」には図７３および図７４に示した定義文が追加される。

【０１７９】

図５１に戻って、ステップＳ８０では、上記のように作成した処理モジュールＭＳを構成するファイルと処理モジュールＵＩを構成するファイルのソースコードを、新たな画像処理モジュールとして画像処理モジュール記憶部２９１に記憶させて、画像処理モジュール作成処理を終了させる。

【０１８０】

以上説明した本実施の形態では、処理モジュールＭＳにより第１画像処理モジュールが構成され、処理モジュールＵＩにより第２画像処理モジュールが構成される。

【０１８１】
また、以上説明した本実施の形態では、開発支援装置１では、ＳＩによる、画像処理検査装置における検査処理のためのプログラム（基本フロー５０７）の開発が行なわれる。基本フロー５０７は、上記検査処理について、当該検査処理中に表示される画面の表示を制御する運転画面ＵＩ５０６と、当該検査処理に含まれる複数の処理段階の各処理内容を制御するための画像処理モジュール（画像処理モジュールＡ５０９Ａ～画像処理モジュールＮ５０９Ｎ）とを含む。各画像処理モジュール５０９Ａ～５０９Ｎは、実際の処理内容を制御する処理モジュールＭＳ５０４Ａ～５０４Ｎと、各処理モジュールＭＳについての設定情報をディスプレイに表示させるための処理モジュールＵＩ５０５Ａ～５０５Ｎとを含む。ＳＩは、処理モジュールＵＩによる画面表示に基づいて、処理モジュールＭＳに対する設定情報を入力できる。

【０１８２】
画像処理モジュール（処理モジュールＭＳおよび処理モジュールＵＩ）は、画像処理モジュール２９１に記憶される。ＳＩは、図６～図５０を参照して説明したように、所望の画像処理モジュールを、テンプレート記憶部１６に記憶されたテンプレートを用いて作成できる。また、ＳＩは、図５１～図６８を参照して説明したように、画像処理モジュール２９１に記憶された処理モジュールＭＳおよび処理モジュールＵＩを用いて、所望の画像処理モジュールを作成することもできる。なお、処理モジュールＵＩによって表示される画面は、画像処理モジュール開発部１０によって、操作入力部７に対して入力された操作内容に基づき、上記したように部品記憶部１５に記憶された部品（ＡｃｔｉｖｅＸコントロール）を貼り付けることにより編集することができる。

【０１８３】
今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

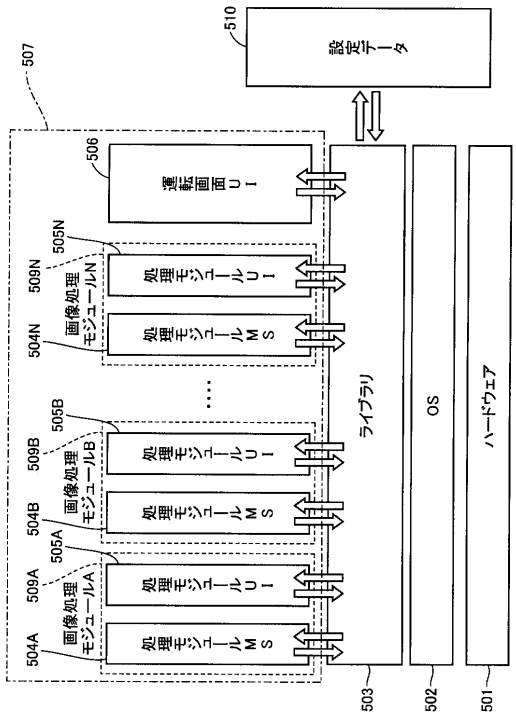
【図面の簡単な説明】

【０１８４】
【図１】本発明の一実施の形態である開発支援装置のシステムの構成を模式的に示す図である。
【図２】本発明の一実施の形態である開発支援装置のハードウェア構成を模式的に示す図である。
【図３】本発明の一実施の形態の開発支援装置において取扱われる画像処理モジュールのテンプレートを構成するファイルの一覧とその機能の例を表わす図である。
【図４】図２のディスプレイに表示される基本フロー開発用画面の一例を示す図である。
【図５】図２のディスプレイに表示される基本フロー開発用画面の他の例を示す図である。
【図６】ＳＩが画像処理モジュールの作成を行なう際にディスプレイ２に表示される画面の一例を示す図である。
【図７】画像処理モジュール開発の処理手順を表わすフローチャートである。
【図８】ＭＳプログラムの編集処理が選択されたときに表示される画面の例を表わす図である。
【図９】ＵＩプログラムの編集処理が選択されたときに表示される画面の例を表わす図である。
【図１０】「ItemDefs.h」のテンプレートを表わす図である。
【図１１】「ItemDefs.h」のテンプレートを表わす図である。
【図１２】編集後の「ItemDefs.h」を表わす図である。
【図１３】編集後の「ItemDefs.h」を表わす図である。
【図１４】「AssignProc.cpp」のテンプレートを表わす図である。
【図１５】編集後の「AssignProc.cpp」を表わす図である。

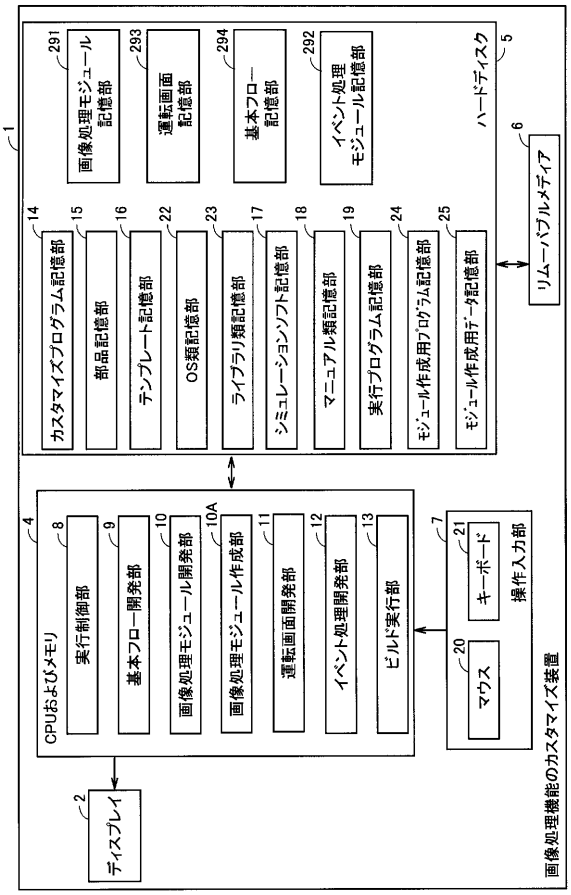
【図１６】編集後の「AssignProc.cpp」を表わす図である。
【図１７】「FigureUpdate.cpp」のテンプレートを表わす図である。
【図１８】「FigureUpdate.cpp」のテンプレートを表わす図である。
【図１９】「ItemInit.cpp」のテンプレートを表わす図である。
【図２０】「ItemInit.cpp」のテンプレートを表わす図である。
【図２１】編集後の「ItemInit.cpp」を表わす図である。
【図２２】編集後の「ItemInit.cpp」を表わす図である。
【図２３】「MeasureDisp.cpp」のテンプレートを表わす図である。
【図２４】「MeasureDisp.cpp」のテンプレートを表わす図である。
【図２５】編集後の「MeasureDisp.cpp」を表わす図である。
【図２６】編集後の「MeasureDisp.cpp」を表わす図である。
【図２７】編集後の「MeasureDisp.cpp」を表わす図である。
【図２８】「MeasureInit.cpp」のテンプレートを表わす図である。
【図２９】「MeasureInit.cpp」のテンプレートを表わす図である。
【図３０】「MeasureOut.cpp」のテンプレートを表わす図である。
【図３１】「MeasureProc.cpp」のテンプレートを表わす図である。
【図３２】編集後の「MeasureProc.cpp」を表わす図である。
【図３３】編集後の「MeasureProc.cpp」を表わす図である。
【図３４】編集後の「MeasureProc.cpp」を表わす図である。
【図３５】「RenumProc.cpp」のテンプレートを表わす図である。
【図３６】「RenumProc.cpp」のテンプレートを表わす図である。
【図３７】編集後の「RenumProc.cpp」を表わす図である。
【図３８】編集後の「RenumProc.cpp」を表わす図である。
【図３９】「UnitData.cpp」のテンプレートを表わす図である。
【図４０】「UnitData.cpp」のテンプレートを表わす図である。
【図４１】「UnitData.cpp」のテンプレートを表わす図である。
【図４２】編集後の「UnitData.cpp」を表わす図である。
【図４３】編集後の「UnitData.cpp」を表わす図である。
【図４４】編集後の「UnitData.cpp」を表わす図である。
【図４５】編集後の「UnitData.cpp」を表わす図である。
【図４６】編集後の「UnitData.cpp」を表わす図である。
【図４７】編集後の「UnitData.cpp」を表わす図である。
【図４８】編集後の「UnitData.cpp」を表わす図である。
【図４９】編集後の「UnitData.cpp」を表わす図である。
【図５０】編集後の「UnitData.cpp」を表わす図である。
【図５１】図２の画像処理モジュール作成部が実行する画像処理モジュール作成処理のフローチャートである。
【図５２】図５１の画像処理モジュール作成処理において表示される画面の一例を示す図である。
【図５３】図５１の画像処理モジュール作成処理において記述される情報の一例を示す図である。
【図５４】図５１の画像処理モジュール作成処理において記述される情報の他の例を示す図である。
【図５５】図５１の画像処理モジュール作成処理において表示される画面の他の例を示す図である。
【図５６】図２のディスプレイにおいてＳＩが画像処理モジュールのカスタマイズを行なう際に表示される画面の一例を示す図である。
【図５７】図２のディスプレイにおいてＳＩが画像処理モジュールのカスタマイズを行なう際に表示される画面の他の例を示す図である。
【図５８】図２のディスプレイにおいてＳＩが画像処理モジュールのカスタマイズを行な

う際に表示される画面のさらに他の例を示す図である。
【図５９】図２のディスプレイにおいてＳＩが画像処理モジュールのカスタマイズを行なう際に表示される画面のさらに他の例を示す図である。
【図６０】図２のディスプレイにおいてＳＩが画像処理モジュールのカスタマイズを行なう際に表示される画面のさらに他の例を示す図である。
【図６１】図５１の画像処理モジュール作成処理において表示される画面のさらに他の例を示す図である。
【図６２】図５１の画像処理モジュール作成処理において表示される画面のさらに他の例を示す図である。
【図６３】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【図６４】図５１の画像処理モジュール作成処理において表示される画面のさらに他の例を示す図である。
【図６５】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【図６６】図５１の画像処理モジュール作成処理において表示される画面のさらに他の例を示す図である。
【図６７】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【図６８】図５１の画像処理モジュール作成処理において表示される画面のさらに他の例を示す図である。
【図６９】図２の開発支援装置に入力されるデータを模式的に示す図である。
【図７０】図６９に示されたデータがＣＳＶ形式のデータに変換された状態を模式的に示す図である。
【図７１】図７０に示されたデータに基づいて作成されるファイルの内容を模式的に示す図である。
【図７２】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【図７３】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【図７４】図５１の画像処理モジュール作成処理において記述される情報のさらに他の例を示す図である。
【符号の説明】
【０１８５】
１　開発支援装置、２　ディスプレイ、４　ＣＰＵおよびメモリ、５　ハードディスク、７　操作入力部、８　実行制御部、９　基本フロー開発部、１０　画像処理モジュール開発部、１０Ａ　画像処理モジュール作成部、１３　ビルド実行部、１４　カスタマイズプログラム記憶部、１６　テンプレート記憶部、１９　実行プログラム記憶部、２０　マウス、２１　キーボード、２２　ＯＳ類記憶部、２３　ライブラリ類記憶部、２４　モジュール作成用プログラム記憶部、２５　モジュール作成用データ記憶部、２９１　画像処理モジュール記憶部、５０１　ハードウェア、５０２　ＯＳ、５０３　ライブラリ、５０４Ａ，５０４Ｂ，５０４Ｎ　処理モジュールＭＳ、５０５Ａ，５０５Ｂ，５０５Ｎ　処理モジュールＵＩ、５０６　運転画面ＵＩ、５０７　基本フロー、５０９Ａ，５０９Ｂ，５０９Ｎ　画像処理モジュール、５１０　設定データ。

【図１】



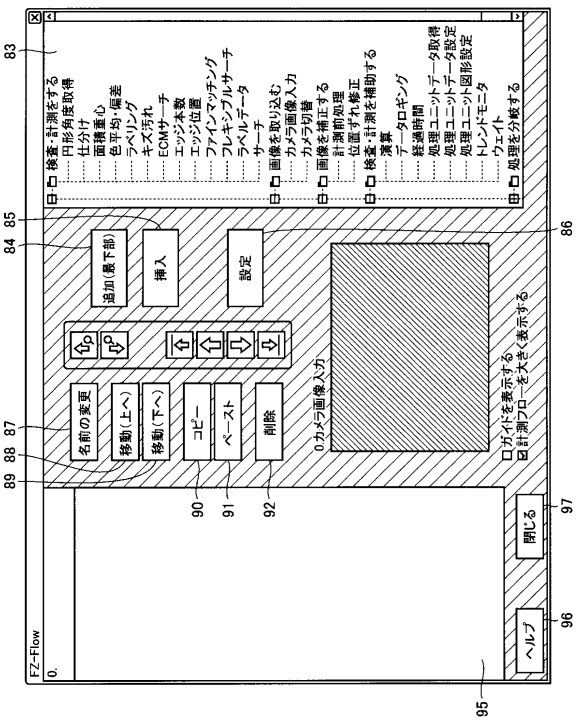
【図２】



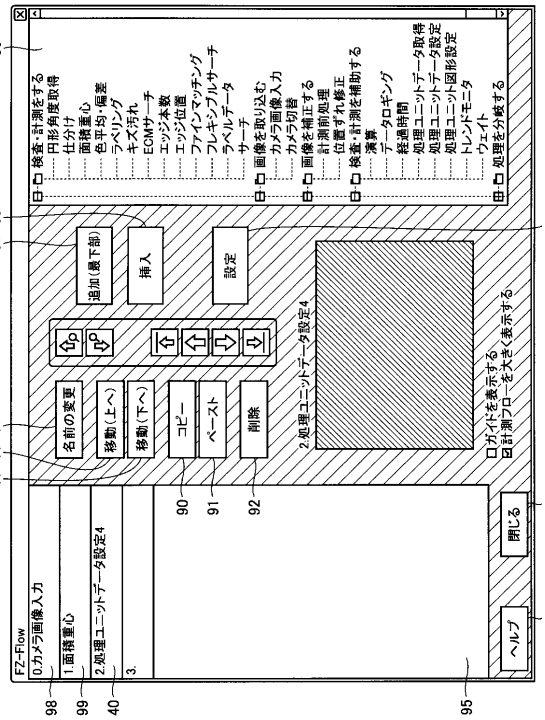
【図３】

```
_Sample
 ├ Sample_jpn.msg      日本語のメッセージファイル
 ├ Sample_eng.msg      英語のメッセージファイル
 ├ Sample_UI           UI
 │  ├ NormalForm.vb        通常使う設定画面
 │  └ NonstopFrom.vb       簡易ノンストップをするときの設定画面
 │
 └ Sample_MS           MS
    ├ AssignProc.cpp       処理ユニット登録時処理関連の処理
    ├ FigureUpdate.cpp     図形データ更新時処理関連の処理
    ├ ItemDefs.h           処理項目の各種定義
必須├ ItemInit.cpp          処理項目の初期化処理関連の処理
    ├ MeasureDisp.cpp      計測結果表示処理関連の処理
    ├ MeasureInit.cpp      計測初期化／終了処理関連の処理
    ├ MeasureOut.cpp       計測結果出力処理関連の処理
    ├ MeasureProc.cpp      計測処理関連の処理
    ├ UnitData.cpp         処理ユニットデータ設定／取得関連の処理
    │
    └ _OtherFile
       ├ FigureData.cpp    処理ユニットの図形データ操作関連の処理
       ├ RenumProc.cpp     処理ユニットの参照番号更新処理関連の処理
任意   ├ SaveLoad.cpp       処理ユニットデータセーブ／ロード関連の処理
       ├ SetupData.cpp     設定時の画像表示処理関連の処理
       └ ThroughProc.cpp   スルー表示のための画像入力／生成処理関連の処理
```

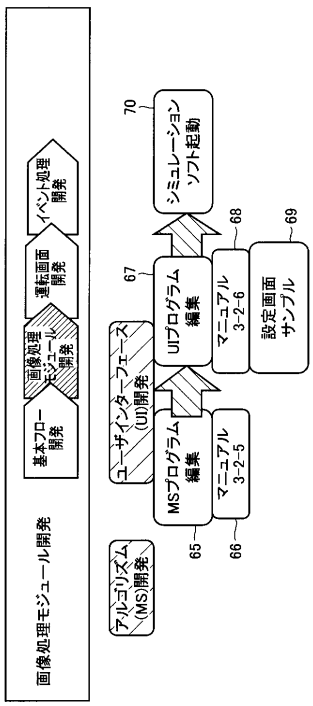
【図４】



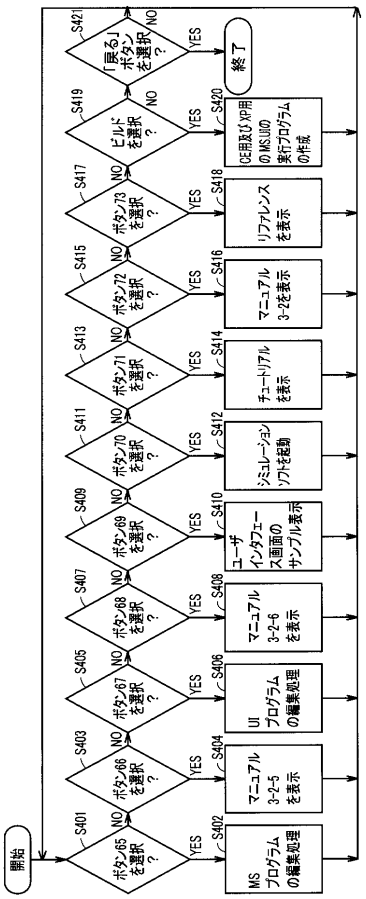
【図５】


FZ-Flow
0.カメラ画像入力
1.面積重心
2.処理ユニットデータ設定4
3.
名前の変更
移動(上へ)
移動(下へ)
コピー
ペースト
削除
追加(最下部)
挿入
設定
2.処理ユニットデータ設定4
ガイドを表示する
計測フローを大きく表示する
検査・計測をする
円形角度取得
仕分け
面積重心
色平均・偏差
ラベリング
キズ汚れ
EDMサーチ
エッジ本数
エッジ位置
ファインマッチング
フレキシブルサーチ
ラベルデータ
サーチ
画像を取り込む
カメラ画像入力
カメラ切替
画像を補正する
計測前処理
位置ずれ修正
検査・計測を補助する
演算
データロギング
経過時間
処理ユニットデータ取得
処理ユニットデータ設定
処理ユニット図形設定
トレンドモニタ
ウェイト
処理を分岐する
ヘルプ
閉じる
83
84
85
86
87
88
89
90
91
92
95
96
97
98
99
40

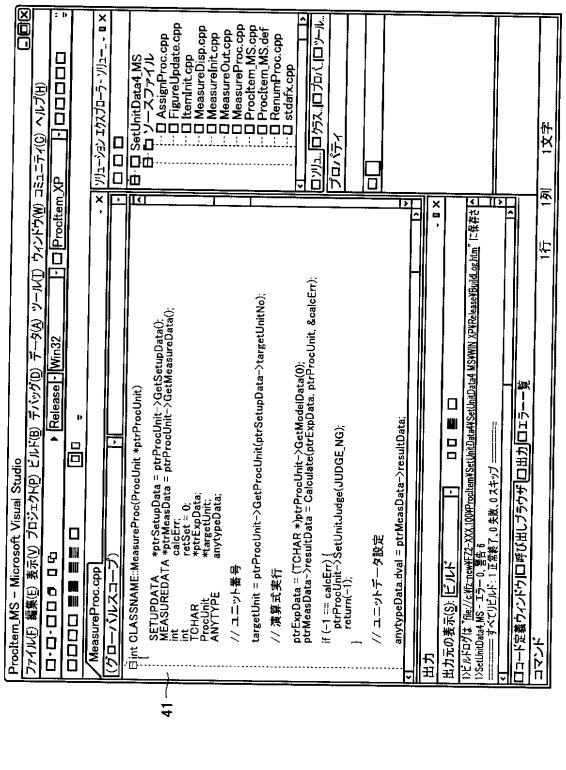
【図６】


画像処理モジュール開発
基本フロー開発
画像処理モジュール開発
運転画面開発
イベント処理開発
アルゴリズム(MS)開発
ユーザインターフェース(UI)開発
MSプログラム編集
UIプログラム編集
シミュレーションソフト起動
マニュアル3-2-5
マニュアル3-2-6
設定画面サンプル
チュートリアル→マニュアル3-2-1
マニュアル→3-2
リファレンス→6
65
66
67
68
69
70
71
72
73

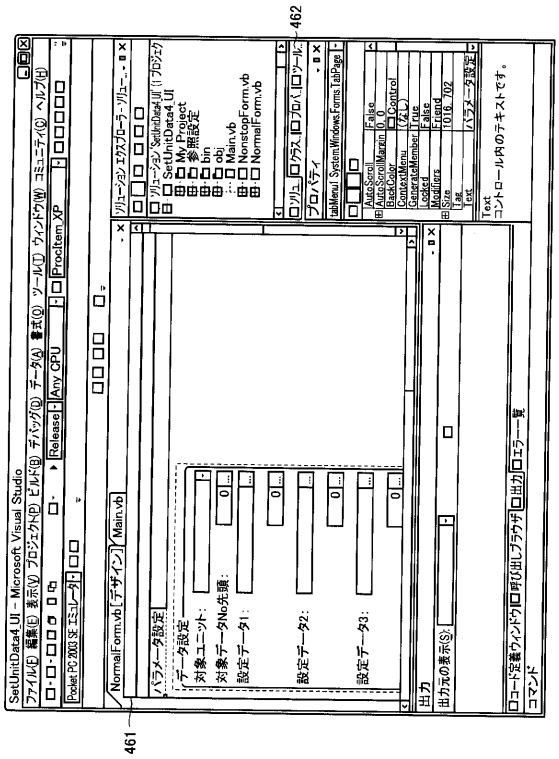
【図７】


開始
ボタン65を選択?
MSプログラムの編集処理
マニュアル3-2-5を表示
UIプログラムの編集処理
マニュアル3-2-6を表示
ユーザインタフェース画面のサンプル表示
シミュレーションソフト起動
チュートリアルを表示
マニュアル3-2-6を表示
リファレンスを表示
CE用及びXP用のMS,UIのプログラムの作成
終了
S401
S402
S403
S404
S405
S406
S407
S408
S409
S410
S411
S412
S413
S414
S415
S416
S417
S418
S419
S420
S421

【図８】


ProcItem.MS - Microsoft Visual Studio
MeasureProc.cpp
(グローバルスコープ)
SetUnitData4.MS
ソースファイル
AssignProc.cpp
FigureUpdate.cpp
ItemInit.cpp
MeasureDisp.cpp
MeasureInit.cpp
MeasureOut.cpp
MeasureProc.cpp
ProcItem.MS.cpp
ProcItem.MS.def
RerunProc.cpp
stdafx.cpp
出力
ビルド
1行 1列 1文字
41

【図９】



【図１０】

```
// ********************************************************************************
//                                          (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : ItemDefs.h
// 処 理 概 要 : 処理項目各種定義
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

// ************************************
//          マ ク ロ 定 義
// ************************************

// 処理項目クラス定義

#define  CLASSNAME     InheritedProcItem

class CLASSNAME : public ProcItem {
public:
        CLASSNAME(void);
        ~CLASSNAME(void);

        virtual int             AssignProc(ProcUnit *ptrProcUnit);
        virtual int             MeasureInit(ProcUnit *ptrProcUnit);
        virtual int             MeasureEnd(ProcUnit *ptrProcUnit);
        virtual int             MeasureProc(ProcUnit *ptrProcUnit);
        virtual int             MeasureOut(ProcUnit *ptrProcUnit);
        virtual int             MeasureDispI(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             MeasureDispG(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             MeasureDispT(ProcUnit *ptrProcUnit, int subNo, TextWindow
*ptrTextWindow);
//      virtual int             TriggerPre(ProcUnit *ptrProcUnit);                    190
//      virtual int             ThroughProc(ProcUnit *ptrProcUnit);                   /
//      virtual int             RenumProc(ProcUnit *ptrProcUnit, RenumInfo *ptrRenumInfo);
//      virtual int             SaveProc(ProcUnit *ptrProcUnit, SaveInfo *ptrSaveInfo);
//      virtual int             LoadProc(ProcUnit *ptrProcUnit, LoadInfo *ptrLoadInfo);
//      virtual int             LoadSetupData(ProcUnit *ptrProcUnit, void *loadAddress, int
loadSize);
//      virtual int             SetupDisp(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             FigureUpdate(ProcUnit *ptrProcUnit, int figureNo);
        virtual int             SetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data);
        virtual int             SetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE
*data);
        virtual int             GetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data);
        virtual int             GetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE
*data);
};
```

【図１１】

```
// 設定データ構造体定義

struct SETUPDATA {
};

// 計測データ構造体定義

struct MEASUREDATA {
};
```

【図１２】

```
// ********************************************************************************
//                                          (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : ItemDefs.h
// 処 理 概 要 : 処理項目各種定義
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

// ************************************
//          マ ク ロ 定 義
// ************************************

// 処理項目クラス定義

#define  CLASSNAME     InheritedProcItem

class CLASSNAME : public ProcItem {
public:
        CLASSNAME(void);
        ~CLASSNAME(void);

        virtual int             AssignProc(ProcUnit *ptrProcUnit);
        virtual int             MeasureInit(ProcUnit *ptrProcUnit);
        virtual int             MeasureEnd(ProcUnit *ptrProcUnit);
        virtual int             MeasureProc(ProcUnit *ptrProcUnit);
        virtual int             MeasureOut(ProcUnit *ptrProcUnit);
        virtual int             MeasureDispI(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             MeasureDispG(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             MeasureDispT(ProcUnit *ptrProcUnit, int subNo, TextWindow
*ptrTextWindow);
//      virtual int             TriggerPre(ProcUnit *ptrProcUnit);                    191
//      virtual int             ThroughProc(ProcUnit *ptrProcUnit);                   /
        virtual int             RenumProc(ProcUnit *ptrProcUnit, RenumInfo *ptrRenumInfo);
//      virtual int             SaveProc(ProcUnit *ptrProcUnit);
//      virtual int             LoadProc(ProcUnit *ptrProcUnit);
//      virtual int             SetupDisp(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow);
        virtual int             FigureUpdate(ProcUnit *ptrProcUnit, int figureNo);
        virtual int             SetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data);
        virtual int             SetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent,
ANYTYPE *data);
        virtual int             GetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data);
        virtual int             GetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent,
ANYTYPE *data);
};
```

【図１３】

192

```
// 処理項目固有定義

#define EXPRESSION_MAX_STRNUM 257                    // 文字列の最大数
```

// 設定データ構造体定義

193

```
struct SETUPDATA {
    long        targetUnitNo;                        // 対象ユニット No.
    long        targetDataNo;                        // 対象データ No.
    long        targetData1No;                       // 対象データ 1No.
    long        targetData2No;                       // 対象データ 2No.
    long        targetData3No;                       // 対象データ 3No.
};
```

// 計測データ構造体定義

194

```
struct MEASUREDATA {
    double      resultData;                          // 設定データ
    double      resultData1;                         // 設定データ 1
    double      resultData2;                         // 設定データ 2
    double      resultData3;                         // 設定データ 3
};
```

【図１４】

```
// ********************************************************************************
//                                        (C) Copyright OMRON CORPORATION 2005-2007
//                                                    All Rights Reserved
// --------------------------------------------------------------------------------
//                                                    FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : AssignProc.cpp
// 処 理 概 要 : 処理ユニット登録時処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include      "StdAfx.h"
#include      <FZ-Include.h>
#include      "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 処理ユニット登録時処理
// 関 数 名 : int AssignProc(ptrProcUnit)
// 引       数 : ProcUnit *ptrProcUnit;        処理ユニット情報へのポインタ
// 戻 り 値 : エラーコード
//                                    NORMAL(0) : 正常終了
//                                    それ以外    : 異常終了
// 機       能 :
// 日付          作成者                内容
// ------------------------------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::AssignProc(ProcUnit *ptrProcUnit)          992N
{
    return(NORMAL);
}
```

【図１５】

```
// ********************************************************************************
//                                        (C) Copyright OMRON CORPORATION 2005-2007
//                                                    All Rights Reserved
// --------------------------------------------------------------------------------
//                                                    FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : AssignProc.cpp
// 処 理 概 要 : 処理ユニット登録時処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include      "StdAfx.h"
#include      <FZ-Include.h>
#include      "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 処理ユニット登録時処理
// 関 数 名 : int AssignProc(ptrProcUnit)
// 引       数 : ProcUnit *ptrProcUnit;        処理ユニット情報へのポインタ
// 戻 り 値 : エラーコード
//                                    NORMAL(0) : 正常終了
//                                    それ以外    : 異常終了
// 機       能 :
// 日付          作成者          152          内容
// ------------------------------------------------------------------------------
// 2007/01/16   AST3 △△△△   新規作成
// ********************************************************************************
151
```

【図１６】

```
int CLASSNAME::AssignProc(ProcUnit *ptrProcUnit)
{
    SETUPDATA* ptrSetupData;                                          153

    // セットアップデータ初期化

    ptrSetupData = ptrProcUnit->GetSetupData();

    ptrSetupData->targetUnitNo = 0;
    ptrSetupData->targetDataNo = 0;

    // 演算式データの初期化

    TCHAR  *ptrExpData;
    int             allocSize = sizeof(TCHAR);

    for (int i = 0; i < 4; i++) {
        ptrProcUnit->ModelDataAlloc(i, allocSize);

        ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(i);
        _tcscpy(ptrExpData, _T(""));
    }
    return(NORMAL);
}
```

【図１７】

```
// ********************************************************************************
//                                          (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : FigureUpdate.cpp
// 処 理 概 要 : 図形データ更新時処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 図形データ更新時処理
// 関 数 名 : int FigureUpdate(ptrProcUnit, figureNo)
// 引 数 : ProcUnit *ptrProcUnit;          処理ユニット情報へのポインタ
//                  int    figureNo;       図形番号
// 戻 り 値 : エラーコード
//                  NORMAL(0) : 正常終了
//                  それ以外   : 異常終了
// 機 能 :
// 日付          作成者                    内容
// ----------  ----------  ------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************
```

【図１８】

```
int CLASSNAME::FigureUpdate(ProcUnit *ptrProcUnit, int figureNo)
{
        return(NORMAL);
}
```

【図１９】

```
// ********************************************************************************
//                                          (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : ItemInit.cpp
// 処 理 概 要 : 処理項目初期化処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 処理項目初期化処理
// 関 数 名 : CLASSNAME(void)
// 引 数 : なし
// 戻 り 値 : なし
// 機 能 :
// 日付          作成者                    内容
// ----------  ----------  ------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************
```

【図２０】

```
                                                                    991N
CLASSNAME::CLASSNAME(void)
{
                                        658
        this->itemIdent      = _T("Sample");    // 処理項目識別名
        this->maker          = _T("OMRON");     // 処理項目製作者名
        this->version        = 100;      659    // バージョン番号 (×100)          660
        this->itemKind       = ITEM_MEASURE;    // 処理項目種別 (=一般計測関連)
        this->setupDataSize  = sizeof(SETUPDATA);  // 計測データ構造体サイズ
        this->measureDataSize = sizeof(MEASUREDATA);    // 計測データ構造体サイズ
        this->figureDataCount = 0;      663     // 図形データ最大個数
        this->modelDataCount  = 0;      661     // モデルデータ最大個数
        this->imageDataCount  = 0;      664     // 画像データ最大個数
        this->innerUnitCount  = 0;  665  662    // 内包処理ユニット最大個数
        this->title          = _T("サンプル");   // 処理項目タイトル名
        // 暫定 → 本当は日本語／英語の処理項目名を入れる。
}

CLASSNAME::~CLASSNAME(void)
{
}

ProcItem *CreateProcItem(void)
{
        return(new CLASSNAME());
}

void DeleteProcItem(CLASSNAME *ptrInstance)
{
        delete ptrInstance;
}
```

【図２１】

```
// ********************************************************************************
//                                          (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : ItemInit.cpp
// 処 理 概 要 : 処理項目初期化処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 処理項目初期化処理
// 関 数 名 : CLASSNAME(void)
// 引 数 : なし
// 戻 り 値 : なし
// 機 能 :
// 日付                155   作成者          内容
// ----------  ----------  ------------------------------------------------------
// 2007/01/21   AST3 △△△△   新規作成
// ********************************************************************************
```

【図２２】

```
CLASSNAME::CLASSNAME(void)
{
        this->itemIdent      = _T("SetUnitData");      // 処理項目識別名
        this->maker          = _T("OMRON AST3");       // 処理項目製作者名
        this->version        = 100;                    // バージョン番号（×100）
        this->itemKind       = ITEM_SUPPORT;           // 処理項目種別（＝計測補助関連）
        this->setupDataSize  = sizeof(SETUPDATA);      // 設定データ構造体サイズ
        this->measureDataSize = sizeof(MEASUREDATA);   // 計測データ構造体サイズ
        this->figureDataCount = 0;                     // 図形データ最大個数
        this->modelDataCount  = 4;                     // モデルデータ最大個数
        this->imageDataCount  = 0;                     // 画像データ最大個数
        this->innerUnitCount  = 0;                     // 内包処理ユニット最大個数
        this->title = _T("処理ユニットデータ設定");    // 処理項目タイトル名
        // 暫定 → 本当は日本語／英語の処理項目名を入れる。
}

CLASSNAME::~CLASSNAME(void)
{
}

ProcItem *CreateProcItem(void)
{
        return(new CLASSNAME());
}

void DeleteProcItem(CLASSNAME *ptrInstance)
{
        delete ptrInstance;
}
```

158
159
160
161
162

【図２３】

```
// ********************************************************************************
//                          (C) Copyright OMRON CORPORATION 2005-2007
//                                      All Rights Reserved
// --------------------------------------------------------------------------------
//                                      FZ シリーズ処理項目
// ********************************************************************************
// ファイル名：MeasureDisp.cpp
// 処 理 概 要：計測結果表示処理関連の処理
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ************************************
//          マ ク ロ 定 義
// ************************************

// ************************************
//          外 部 宣 言
// ************************************

// ************************************
//          静 的 変 数
// ************************************

// ************************************
//          外 部 変 数
// ************************************

// ************************************
//          関 数 参 照
// ************************************

// ********************************************************************************
// 機 能 名 称：計測結果表示処理（画像表示）
// 関 数 名：int MeasureDispI(ptrProcUnit, subNo, ptrImageWindow)
// 引       数：ProcUnit    *ptrProcUnit;       処理ユニット情報へのポインタ
//                          int         subNo;              表示サブ番号
//                          ImageWindow *ptrImageWindow;
画像表示情報へのポインタ
// 戻 り 値：エラーコード
//                          NORMAL(0)：正常終了
//                          それ以外   ：異常終了
// 機       能：
// 日付             作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************
```

【図２４】

```
int CLASSNAME::MeasureDispI(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow)
{
        IMAGE           *image = ptrProcUnit->GetMeasureImage(0);

        ptrImageWindow->ImageDisp(image);

        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称：計測結果表示処理（グラフィック表示）
// 関 数 名：int MeasureDispG(ptrProcUnit, subNo, ptrImageWindow)
// 引       数：ProcUnit    *ptrProcUnit;       処理ユニット情報へのポインタ
//                          int         subNo;              表示サブ番号
//                          ImageWindow *ptrImageWindow;
画像表示情報へのポインタ
// 戻 り 値：エラーコード
//                          NORMAL(0)：正常終了
//                          それ以外   ：異常終了
// 機       能：
// 日付             作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::MeasureDispG(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow)
{
        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称：計測結果表示処理（テキスト表示）
// 関 数 名：int MeasureDispT(ptrProcUnit, subNo, ptrTextWindow)
// 引       数：ProcUnit    *ptrProcUnit;       処理ユニット情報へのポインタ
//                          int         subNo;              表示サブ番号
//                          TextWindow  *ptrTextWindow;
テキスト表示情報へのポインタ
// 戻 り 値：エラーコード
//                          NORMAL(0)：正常終了
//                          それ以外   ：異常終了
// 機       能：
// 日付             作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::MeasureDispT(ProcUnit *ptrProcUnit, int subNo, TextWindow
*ptrTextWindow)
{
        return(NORMAL);
}
```

【図２５】

```
// ********************************************************************************
//                          (C) Copyright OMRON CORPORATION 2005-2007
//                                      All Rights Reserved
// --------------------------------------------------------------------------------
//                                      FZ シリーズ処理項目
// ********************************************************************************
// ファイル名：MeasureDisp.cpp
// 処 理 概 要：計測結果表示処理関連の処理
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ************************************
//          マ ク ロ 定 義
// ************************************

// ************************************
//          外 部 宣 言
// ************************************

// ************************************
//          静 的 変 数
// ************************************

// ************************************
//          外 部 変 数
// ************************************

// ************************************
//          関 数 参 照
// ************************************

// ********************************************************************************
// 機 能 名 称：計測結果表示処理（画像表示）
// 関 数 名：int MeasureDispI(ptrProcUnit, subNo, ptrImageWindow)
// 引       数：ProcUnit    *ptrProcUnit;       処理ユニット情報へのポインタ
//                          int         subNo;              表示サブ番号
//                          ImageWindow *ptrImageWindow;
画像表示情報へのポインタ
// 戻 り 値：エラーコード
//                          NORMAL(0)：正常終了
//                          それ以外   ：異常終了
// 機       能：
// 日付             作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************
```

【図２６】

```
int CLASSNAME::MeasureDispI(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow)
{
        IMAGE           *image = ptrProcUnit->GetMeasureImage(0);

        ptrImageWindow->ImageDisp(image);

        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称 : 計測結果表示処理 (グラフィック表示)
// 関 数 名 : int MeasureDispG(ptrProcUnit, subNo, ptrImageWindow)
// 引 数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                                  int     subNo;              表示サブ番号
//                                  ImageWindow *ptrImageWindow;
画像表示情報へのポインタ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外  : 異常終了
// 機 能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------
// 2007/??/??  AST3 ○○○○  新規作成
// ********************************************************************************

int CLASSNAME::MeasureDispG(ProcUnit *ptrProcUnit, int subNo, ImageWindow
*ptrImageWindow)
{
        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称 : 計測結果表示処理 (テキスト表示)
// 関 数 名 : int MeasureDispT(ptrProcUnit, subNo, ptrTextWindow)
// 引 数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                                  int     subNo;              表示サブ番号
//                                  TextWindow *ptrTextWindow;
テキスト表示情報へのポインタ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外  : 異常終了
// 機 能 :
// 日付 163    165   作成者                  内容
// ---------- ---------- ----------------------------------------
// 2007/01/16  AST3 △△△△  新規作成
// 2007/03/14  CEC  ×××× ~166   言語対応
// ********************************************************************************
}
```

164

【図２７】

```
int CLASSNAME::MeasureDispT(ProcUnit *ptrProcUnit, int subNo, TextWindow
*ptrTextWindow)
{
        MEASUREDATA     *ptrMeasData = ptrProcUnit->GetMeasureData();   ~167
        SETUPDATA       *ptrSetupData = ptrProcUnit->GetSetupData();
        long            unitJudge = ptrProcUnit->GetUnitJudge();

        ptrTextWindow->DrawJudgeText(unitJudge);

        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("Unit%d")), ptrSetupData->targetUnitNo);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("DataNumber%d")), ptrSetupData->targetDataNo);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("SetData%f")), ptrMeasData->resultData);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("DataNumber1%d")), ptrSetupData->targetData1No);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("SetData1%f")), ptrMeasData->resultData1);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("DataNumber2%d")), ptrSetupData->targetData2No);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("SetData2%f")), ptrMeasData->resultData2);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("DataNumber3%d")), ptrSetupData->targetData3No);
        ptrTextWindow->DrawText(unitJudge, TRUE,
                                                        ptrProcUnit-
>GetText(_T("SetData3%f")), ptrMeasData->resultData3);

        return(NORMAL);
}
```

【図２８】

```
// ********************************************************************************
//                              (C) Copyright OMRON CORPORATION 2005-2007
//                                   All Rights Reserved
// ---------------------------------------------------------------------------
//                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : MeasureInit.cpp
// 処 理 概 要 : 計測初期化／終了処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称 : 計測初期化処理
// 関 数 名 : int MeasureInit(ptrProcUnit)
// 引 数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外  : 異常終了
// 機 能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------
// 2007/??/??  AST3 ○○○○  新規作成
// ********************************************************************************

int CLASSNAME::MeasureInit(ProcUnit *ptrProcUnit)
{
        return(NORMAL);
}
```

【図２９】

```
// ********************************************************************************
// 機 能 名 称 : 計測終了処理
// 関 数 名 : int MeasureEnd(ptrProcUnit)
// 引 数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外  : 異常終了
// 機 能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------
// 2007/??/??  AST3 ○○○○  新規作成
// ********************************************************************************

int CLASSNAME::MeasureEnd(ProcUnit *ptrProcUnit)
{
        return(NORMAL);
}
```

【図３０】

```
// ********************************************************************************
//                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名：MeasureOut.cpp
// 処 理 概 要：計測結果出力処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称：計測結果出力処理
// 関 数 名：int MeasureOut(ptrProcUnit)
// 引       数：ProcUnit *ptrProcUnit;        処理ユニット情報へのポインタ
// 戻 り 値：エラーコード
//                                  NORMAL(0)：正常終了
//                                  それ以外   ：異常終了
// 機       能：
// 日付                 作成者                 内容
// --------------------------------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::MeasureOut(ProcUnit *ptrProcUnit)
{
        return(NORMAL);
}
```

【図３１】

```
// ********************************************************************************
//                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名：MeasureProc.cpp
// 処 理 概 要：計測処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称：計測処理
// 関 数 名：int MeasureProc(ptrProcUnit)
// 引       数：ProcUnit *ptrProcUnit;        処理ユニット情報へのポインタ
// 戻 り 値：エラーコード
//                                  NORMAL(0)：正常終了
//                                  それ以外   ：異常終了
// 機       能：
// 日付                 作成者                 内容
// --------------------------------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::MeasureProc(ProcUnit *ptrProcUnit)
{
        return(NORMAL);
}
```

【図３２】

```
// ********************************************************************************
//                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名：MeasureProc.cpp
// 処 理 概 要：計測処理関連の処理
// ********************************************************************************

// ****************************************
//          インクルードファイル
// ****************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ****************************************
//          マ ク ロ 定 義
// ****************************************

// ****************************************
//          外 部 宣 言
// ****************************************

// ****************************************
//          静 的 変 数
// ****************************************

// ****************************************
//          外 部 変 数
// ****************************************

// ****************************************
//          関 数 参 照
// ****************************************

// ********************************************************************************
// 機 能 名 称：計測処理
// 関 数 名：int MeasureProc(ptrProcUnit)
// 引       数：ProcUnit *ptrProcUnit;        処理ユニット情報へのポインタ
// 戻 り 値：エラーコード
//                                  NORMAL(0)：正常終了
//                                  それ以外   ：異常終了
// 機       能：
// 日付                 作成者                 内容
// --------------------------------------------------------------------------------
// 2007/01/16   AST3 △△△△   新規作成
// ********************************************************************************
```

168

169

【図３３】

```
int CLASSNAME::MeasureProc(ProcUnit *ptrProcUnit)                    170
{
    SETUPDATA       *ptrSetupData = ptrProcUnit->GetSetupData();
    MEASUREDATA     *ptrMeasData = ptrProcUnit->GetMeasureData();
    int             calcErr;
    int             retSet = 0;
    TCHAR           *ptrExpData;
    ProcUnit        *targetUnit;
    ANYTYPE         anytypeData;

    // ユニット番号

    targetUnit = ptrProcUnit->GetProcUnit(ptrSetupData->targetUnitNo);

    // 演算式実行

    ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(0);
    ptrMeasData->resultData = Calculate(ptrExpData, ptrProcUnit, &calcErr);

    if (-1 == calcErr) {
        ptrProcUnit->SetUnitJudge(JUDGE_NG);
        return(-1);
    }

    // ユニットデータ設定

    anytypeData.dval = ptrMeasData->resultData;
    anytypeData.type = T_DOUBLE;

    retSet |= targetUnit->SetUnitData(ptrSetupData->targetDataNo, &anytypeData);

    // 演算式実行 1

    ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(1);
    ptrMeasData->resultData1 = Calculate(ptrExpData, ptrProcUnit, &calcErr);

    if (-1 == calcErr) {
        ptrProcUnit->SetUnitJudge(JUDGE_NG);
        return(-1);
    }

    // ユニットデータ設定 1

    anytypeData.dval = ptrMeasData->resultData1;
    anytypeData.type = T_DOUBLE;

    retSet |= targetUnit->SetUnitData(ptrSetupData->targetData1No, &anytypeData);
```

【図３４】

```
        // 演算式実行 2

        ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(2);
        ptrMeasData->resultData2 = Calculate(ptrExpData, ptrProcUnit, &calcErr);

        if (-1 == calcErr) {
                ptrProcUnit->SetUnitJudge(JUDGE_NG);
                return(-1);
        }

        // ユニットデータ設定 2

        anytypeData.dval = ptrMeasData->resultData2;
        anytypeData.type = T_DOUBLE;

        retSet |= targetUnit->SetUnitData(ptrSetupData->targetData2No, &anytypeData);

        // 演算式実行 3

        ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(3);
        ptrMeasData->resultData3 = Calculate(ptrExpData, ptrProcUnit, &calcErr);

        if (-1 == calcErr) {
                ptrProcUnit->SetUnitJudge(JUDGE_NG);
                return(-1);
        }

        // ユニットデータ設定 3

        anytypeData.dval = ptrMeasData->resultData3;
        anytypeData.type = T_DOUBLE;

        retSet |= targetUnit->SetUnitData(ptrSetupData->targetData3No, &anytypeData);

        // 総合判定

        if (NORMAL != retSet) {
                ptrProcUnit->SetUnitJudge(JUDGE_NG);
                return(-1);
        } else {
                ptrProcUnit->SetUnitJudge(JUDGE_OK);
        }

        return(NORMAL);
}
```

【図３５】

```
// ********************************************************************************
//                                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
//ファイル名 : RenumProc.cpp
//処 理 概 要 : 処理ユニット参照番号更新処理関連の処理
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ************************************
//          マ ク ロ 定 義
// ************************************

// ************************************
//          外 部 宣 言
// ************************************

// ************************************
//          静 的 変 数
// ************************************

// ************************************
//          外 部 変 数
// ************************************

// ************************************
//          関 数 参 照
// ************************************

// ********************************************************************************
// 機 能 名 称 : 処理ユニット参照番号更新処理
// 関 数 名 : int RenumProc(ptrProcUnit, ptrRenumInfo)
// 引 数 : ProcUnit *ptrProcUnit;  処理ユニット情報へのポインタ
//                                 RenumInfo *ptrRenumInfo; フロー編集情報へのポインタ
// 戻 り 値 : エラーコード
//                                 NORMAL(0) : 正常終了
//                                 それ以外 : 異常終了
// 機 能 :
// 日付        作成者        内容
// --------------------------------------------------------------------------------
// 2007/??/??  AST3 ○○○○  新規作成
// ********************************************************************************
```

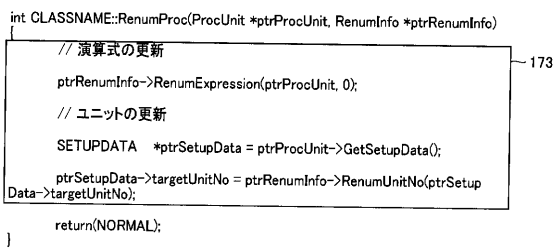
【図３６】

```
int CLASSNAME::RenumProc(ProcUnit *ptrProcUnit, RenumInfo *ptrRenumInfo)
{
        return(NORMAL);
}
```

【図３７】

```
// ********************************************************************************
//                                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : RenumProc.cpp
// 処 理 概 要 : 処理ユニット参照番号更新処理関連の処理
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ************************************
//          マ ク ロ 定 義
// ************************************

// ************************************
//          外 部 宣 言
// ************************************

// ************************************
//          静 的 変 数
// ************************************

// ************************************
//          外 部 変 数
// ************************************

// ************************************
//          関 数 参 照
// ************************************

// ********************************************************************************
// 機 能 名 称 : 処理ユニット参照番号更新処理
// 関 数 名 : int RenumProc(ptrProcUnit, ptrRenumInfo)
// 引 数 : ProcUnit *ptrProcUnit;  処理ユニット情報へのポインタ
//                                 RenumInfo *ptrRenumInfo; フロー編集情報へのポインタ
// 戻 り 値 : エラーコード
//                                 NORMAL(0) : 正常終了
//                                 それ以外 : 異常終了
// 機 能 :
// 日付        作成者        172   内容
// --------------------------------------------------------------------------------
// 2007/01/21  AST3 △△△△  新規作成
// ********************************************************************************
```

【図３８】



```
int CLASSNAME::RenumProc(ProcUnit *ptrProcUnit, RenumInfo *ptrRenumInfo)
{
        // 演算式の更新                                                    ~173

        ptrRenumInfo->RenumExpression(ptrProcUnit, 0);

        // ユニットの更新

        SETUPDATA    *ptrSetupData = ptrProcUnit->GetSetupData();

        ptrSetupData->targetUnitNo = ptrRenumInfo->RenumUnitNo(ptrSetup
Data->targetUnitNo);

        return(NORMAL);
}
```

【図３９】

```
// ********************************************************************************
//                                  (C) Copyright OMRON CORPORATION 2005-2007
//                                                  All Rights Reserved
// --------------------------------------------------------------------------------
//                                                  FZ シリーズ処理項目
// ********************************************************************************
// ファイル名 : UnitData.cpp
// 処 理 概 要 : 処理ユニットデータ設定／取得関連の処理
// ********************************************************************************

// ************************************
//          インクルードファイル
// ************************************

#include        "StdAfx.h"
#include        <FZ-Include.h>
#include        "ItemDefs.h"

// ************************************
//          マ ク ロ 定 義
// ************************************

// ************************************
//          外 部 宣 言
// ************************************

// ************************************
//          静 的 変 数
// ************************************

// ************************************
//          外 部 変 数
// ************************************

// ************************************
//          関 数 参 照
// ************************************

// ********************************************************************************
// 機 能 名 称 : 処理ユニットデータ設定処理
// 関 数 名 : int SetUnitData(ptrProcUnit, dataNo, data)
// 引       数 : ProcUnit *ptrProcUnit;          処理ユニット情報へのポインタ
//                                  int     dataNo;             データ番号
//                                  ANYTYPE  *data;                         設定データ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外    : 異常終了
// 機       能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************
```

【図４０】

```
int CLASSNAME::SetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称 : 処理ユニットデータ設定処理
// 関 数 名 : int SetUnitData(ptrProcUnit, dataIdent, data)
// 引       数 : ProcUnit *ptrProcUnit;          処理ユニット情報へのポインタ
//                                  TCHAR   *dataIdent;   データ識別名
//                                  ANYTYPE  *data;                         設定データ
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外    : 異常終了
// 機       能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::SetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE *data)
{
        return(NORMAL);
}

// ********************************************************************************
// 機 能 名 称 : 処理ユニットデータ取得処理
// 関 数 名 : int GetUnitData(ptrProcUnit, dataNo, data)
// 引       数 : ProcUnit *ptrProcUnit;          処理ユニット情報へのポインタ
//                                  int     dataNo;             データ番号
//                                  ANYTYPE  *data;
取得データ格納領域
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外    : 異常終了
// 機       能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::GetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        return(NORMAL);
}
```

【図４１】

```
// ********************************************************************************
// 機 能 名 称 : 処理ユニットデータ取得処理
// 関 数 名 : int GetUnitData(ptrProcUnit, dataIdent, data)
// 引       数 : ProcUnit *ptrProcUnit;          処理ユニット情報へのポインタ
//                                  TCHAR   *dataIdent;   データ識別名
//                                  ANYTYPE  *data;
取得データ格納領域
// 戻 り 値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外    : 異常終了
// 機       能 :
// 日付                作成者                  内容
// ---------- ---------- ----------------------------------------------------------
// 2007/??/??   AST3 ○○○○   新規作成
// ********************************************************************************

int CLASSNAME::GetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE *data)
{
        return(NORMAL);
}
```

【図４２】

```
// ***************************************************************************
//                                   (C) Copyright OMRON CORPORATION 2005-2007
//                                                         All Rights Reserved
// ---------------------------------------------------------------------------
//                                                          FZ シリーズ処理項目
// ***************************************************************************
// ファイル名 : UnitData.cpp
// 処 理 概 要 : 処理ユニットデータ設定／取得関連の処理
// ***************************************************************************

// *************************************
//          インクルードファイル
// *************************************

#include     "StdAfx.h"
#include     <FZ-Include.h>
#include     "ItemDefs.h"

// *************************************
//           マ ク ロ 定 義
// *************************************
                                                      174
// 処理項目固有定義

#define DATA_NAME_MAX_STRSIZE 16 // 項目データ名のサイズ

// 参照項目データ構造体

typedef struct _REFERENCE_DATA_STRUCT {
        int              unitDataNo;
                         // 項目番号
        TCHAR unitDataName[DATA_NAME_MAX_STRSIZE];   // 項目名
} REFERENCE_DATA_STRUCT;

// *************************************
//           外 部 宣 言
// *************************************

// *************************************
//           静 的 変 数
// *************************************
```

【図４３】

```
// 参照項目テーブル                                                    175

const REFERENCE_DATA_STRUCT unitReferenceData[] = {
        {0,          _T("JG")},
        {0,          _T("judge")},
        {5,          _T("DT")},
        {5,          _T("resultData")},
        {6,          _T("DT1")},
        {6,          _T("resultData1")},
        {7,          _T("DT2")},
        {7,          _T("resultData2")},
        {8,          _T("DT3")},
        {8,          _T("resultData3")},
        {120,   _T("targetUnitNo")},
        {121,   _T("targetDataNo")},
        {122,   _T("targetData1No")},
        {123,   _T("targetData2No")},
        {124,   _T("targetData3No")},
        {125,   _T("setupData")},
        {126,   _T("setupData1")},
        {127,   _T("setupData2")},
        {128,   _T("setupData3")},
        {-1, NULL}
};

// *************************************
//           外 部 変 数
// *************************************

// *************************************
//           関 数 参 照
// *************************************

// ローカル関数プロトタイプ                                            176

static int setData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *getData);
static int getData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *getData);

// ***************************************************************************
// 機 能 名 称 : 処理ユニットデータ設定処理
// 関   数   名 : int SetUnitData(ptrProcUnit, dataNo, data)
// 引        数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                             int      dataNo;          データ番号
//                             ANYTYPE  *data;                          設定データ
// 戻  り  値 : エラーコード
//                             NORMAL(0) : 正常終了
//                             それ以外   : 異常終了
// 機        能 :
// 日付                  作成者  178              内容
// ----------- ------------------------------------------------
// 2007/01/16  AST3 △△△△   新規作成
// ***************************************************************************
177
```

【図４４】

```
int CLASSNAME::SetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        // データ設定                                        179

        if (0 != setData(ptrProcUnit, dataNo, data)) {
                // 項目は存在しない

                return(-1);
        }
        return(NORMAL);
}

// ***************************************************************************
// 機 能 名 称 : 処理ユニットデータ設定処理
// 関   数   名 : int SetUnitData(ptrProcUnit, dataIdent, data)
// 引        数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                             TCHAR    *dataIdent;  データ識別名
//                             ANYTYPE  *data;                          設定データ
// 戻  り  値 : エラーコード
//                             NORMAL(0) : 正常終了
//                             それ以外   : 異常終了
// 機        能 :
// 日付                  作成者  181              内容
// ----------- ------------------------------------------------
// 2007/01/16  AST3 △△△△   新規作成
// ***************************************************************************
180
int CLASSNAME::SetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE *data)
{
        int dataNo=0;                                                  182

        // テーブル検索

        while(1){
                if (-1 == unitReferenceData[dataNo].unitDataNo) {
                        // 項目は存在しない

                        return(-1);
                }
                if (0 == _tcscmp((TCHAR *)unitReferenceData[dataNo].unitDataName,
                                (TCHAR *)dataIdent)) {
                        break;
                }
                dataNo+=1;
        }

        // データ設定

        if (0 != setData(ptrProcUnit, unitReferenceData[dataNo].unitDataNo, data)) {
                // 項目は存在しない
                return(-1);
        }
        return(NORMAL);
}
```

【図４５】

```
// ***************************************************************************
// 機 能 名 称 : 処理ユニットデータ取得処理
// 関   数   名 : int GetUnitData(ptrProcUnit, dataNo, data)
// 引        数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                             int      dataNo;          データ番号
//                             ANYTYPE  *data;
取得データ格納領域
// 戻  り  値 : エラーコード
//                             NORMAL(0) : 正常終了
//                             それ以外   : 異常終了
// 機        能 :
// 日付                  作成者  184              内容
// ----------- ------------------------------------------------
// 2007/01/16  AST3 △△△△   新規作成
// ***************************************************************************
183
int CLASSNAME::GetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        // データ取得                                        185

        if (0 != getData(ptrProcUnit, dataNo, data)) {
                // 項目は存在しない

                return(-1);
        }
        return(NORMAL);
}

// ***************************************************************************
// 機 能 名 称 : 処理ユニットデータ取得処理
// 関   数   名 : int GetUnitData(ptrProcUnit, dataIdent, data)
// 引        数 : ProcUnit *ptrProcUnit;      処理ユニット情報へのポインタ
//                             TCHAR    *dataIdent;  データ識別名
//                             ANYTYPE  *data;
取得データ格納領域
// 戻  り  値 : エラーコード
//                             NORMAL(0) : 正常終了
//                             それ以外   : 異常終了
// 機        能 :
// 日付                  作成者  187              内容
// ----------- ------------------------------------------------
// 2007/01/16  AST3 △△△△   新規作成
// ***************************************************************************
186
```

【図４６】

```
int CLASSNAME::GetUnitData(ProcUnit *ptrProcUnit, TCHAR *dataIdent, ANYTYPE *data)
{
        int dataNo=0;                                                          ~188

        // テーブル検索

        while(1){
                if (-1 == unitReferenceData[dataNo].unitDataNo) {
                        // 項目は存在しない

                        return(-1);
                }
                if (0 == _tcscmp((TCHAR *)unitReferenceData[dataNo].unitDataName,
                                        (TCHAR *)dataIdent)) {
                        break;
                }
                dataNo+=1;
        }

        // データ取得

        if (0 != getData(ptrProcUnit, unitReferenceData[dataNo].unitDataNo, data)) {
                // 項目は存在しない

                return(-1);
        }
        return(NORMAL);
}


// ********************************************************************************
// 機 能 名 称 : ユニットデータ設定処理部
// 関　数　名 : static int setDataProc(
//                                         ProcUnit *ptrProcUnit, int dataNo,
ANYTYPE *setData)
// 引　　数 : ProcUnit *ptrProcUnit;       処理ユニット情報へのポインタ
//                                 int     dataNo;         データ番号
//                                 ANYTYPE  *setData;                      設定データ
// 戻　り　値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外　: 異常終了
// 機　　能 : ユニットデータの設定処理実行処理部
// 日付                 作成者                          内容
// ---------- ---------------- ------------------------------------------------
// 2007/01/16   AST3 △△△△   新規作成
// ********************************************************************************
```

【図４７】

```
static int setData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *setData)
{
        SETUPDATA *ptrSetupData = ptrProcUnit->GetSetupData();
        int retVal=NORMAL;

        switch (dataNo) {
        case 120:
                ptrSetupData->targetUnitNo = (int)setData->dval;        // 対象ユニット No.
                break;
        case 121:
                ptrSetupData->targetDataNo = (int)setData->dval;        // 対象データ No.
                break;
        case 122:
                ptrSetupData->targetData1No = (int)setData->dval;       // 対象データ No.
                break;
        case 123:
                ptrSetupData->targetData2No = (int)setData->dval;       // 対象データ No.
                break;
        case 124:
                ptrSetupData->targetData3No = (int)setData->dval;       // 対象データ No.
                break;
        case 125:
                // 演算式
                if (T_STRING == setData->type) {
                        long strLen = _tcslen(setData->string)+1;       // 文字数

                        ptrProcUnit->ModelDataAlloc(0, strLen * sizeof(TCHAR));
                        TCHAR *ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(0);

                        _tcscpy(ptrExpData, setData->string);
                }
                else {
                        retVal = -1;
                }
        case 126:
                // 演算式
                if (T_STRING == setData->type) {
                        long strLen = _tcslen(setData->string)+1;       // 文字数

                        ptrProcUnit->ModelDataAlloc(1, strLen * sizeof(TCHAR));
                        TCHAR *ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(1);

                        _tcscpy(ptrExpData, setData->string);
                }
                else {
                        retVal = -1;
                }
```

【図４８】

```
        case 127:
                // 演算式
                if (T_STRING == setData->type) {
                        long strLen = _tcslen(setData->string)+1;       // 文字数

                        ptrProcUnit->ModelDataAlloc(2, strLen * sizeof(TCHAR));
                        TCHAR *ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(2);

                        _tcscpy(ptrExpData, setData->string);
                }
                else {
                        retVal = -1;
                }
                break;
        case 128:
                // 演算式
                if (T_STRING == setData->type) {
                        long strLen = _tcslen(setData->string)+1;
// 文字数＋終端コード

                        ptrProcUnit->ModelDataAlloc(3, strLen * sizeof(TCHAR));
                        TCHAR *ptrExpData = (TCHAR *)ptrProcUnit->GetModelData(3);

                        _tcscpy(ptrExpData, setData->string);
                }
                else {
                        retVal = -1;
                }
                break;
        default:
                retVal = -1;    // 該当項目無し
        }
        return(retVal);
}


// ********************************************************************************
// 機 能 名 称 : ユニットデータ取得処理部
// 関　数　名 : static int getDataProc(
//                                         ProcUnit *ptrProcUnit, int dataNo,
ANYTYPE *setData)
// 引　　数 : ProcUnit *ptrProcUnit;       処理ユニット情報へのポインタ
//                                 int     dataNo;         データ番号
//                                 ANYTYPE  *setData;                      設定データ
// 戻　り　値 : エラーコード
//                                  NORMAL(0) : 正常終了
//                                  それ以外　: 異常終了
// 機　　能 : ユニットデータの取得処理実行処理部
// 日付                 作成者                          内容
// ---------- ---------------- ------------------------------------------------
// 2007/01/16   AST3 △△△△   新規作成
// ********************************************************************************
```

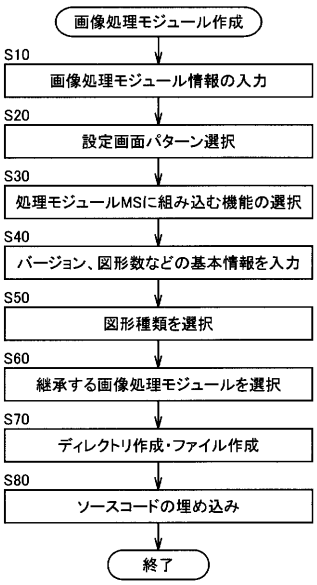
【図４９】

```
static int getData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *getData)
{
        SETUPDATA *ptrSetupData = ptrProcUnit->GetSetupData();
        MEASUREDATA *ptrMeasData = ptrProcUnit->GetMeasureData();
        int retVal=NORMAL;

        switch (dataNo) {
        case 0:
                // 判定
                getData->ival = ptrProcUnit->GetUnitJudge();
                getData->type = T_INTEGER;
                break;
        case 5:
                getData->dval = ptrMeasData->resultData;
                getData->type = T_DOUBLE;
                break;
        case 6:
                getData->dval = ptrMeasData->resultData1;
                getData->type = T_DOUBLE;
                break;
        case 7:
                getData->dval = ptrMeasData->resultData2;
                getData->type = T_DOUBLE;
                break;
        case 8:
                getData->dval = ptrMeasData->resultData3;
                getData->type = T_DOUBLE;
                break;
        case 120:
                getData->ival = ptrSetupData->targetUnitNo;
                getData->type = T_INTEGER;
                break;
        case 121:
                getData->ival = ptrSetupData->targetDataNo;
                getData->type = T_INTEGER;
                break;
        case 122:
                getData->ival = ptrSetupData->targetData1No;
                getData->type = T_INTEGER;
                break;
        case 123:
                getData->ival = ptrSetupData->targetData2No;
                getData->type = T_INTEGER;
                break;
        case 124:
                getData->ival = ptrSetupData->targetData3No;
                getData->type = T_INTEGER;
                break;
        case 125:
                // 演算式
                {
                        getData->string = (TCHAR *)ptrProcUnit->GetModelData(0);
                        getData->type = T_STRING;
                }
                break;
```

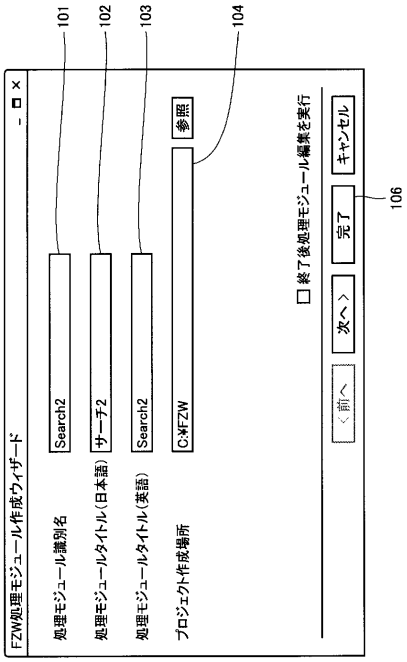
【図５０】

```
        case 126:
                // 演算式
                {
                        getData->string = (TCHAR *)ptrProcUnit->GetModelData(1);
                        getData->type = T_STRING;
                }
                break;
        case 127:
                // 演算式
                {
                        getData->string = (TCHAR *)ptrProcUnit->GetModelData(2);
                        getData->type = T_STRING;
                }
                break;
        case 128:
                // 演算式
                {
                        getData->string = (TCHAR *)ptrProcUnit->GetModelData(3);
                        getData->type = T_STRING;
                }
                break;
        default:
                retVal = -1;    // 該当項目無し
                break;
        }
        return(retVal);
}
```

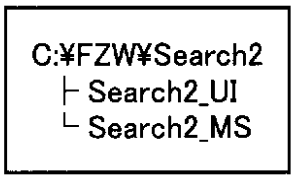
【図５１】



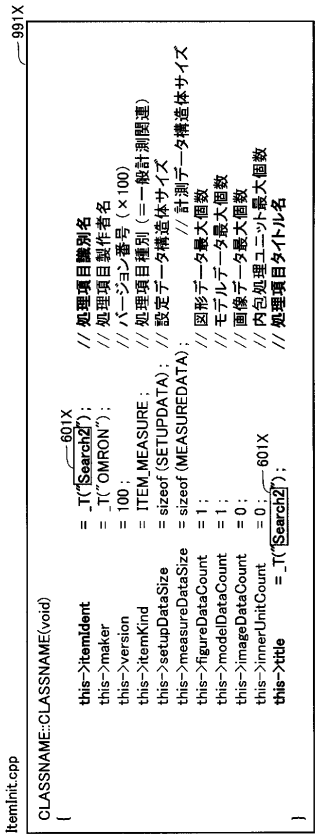
【図５２】



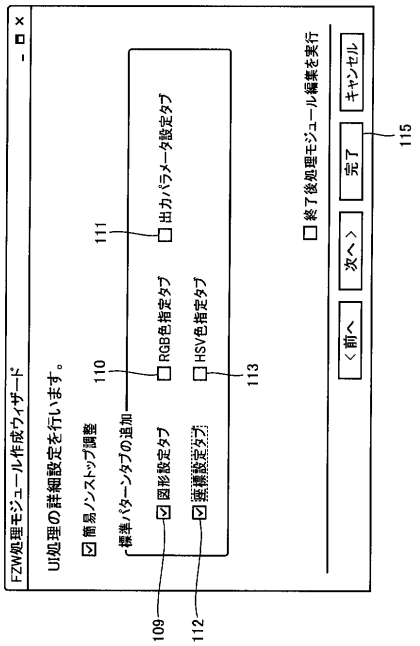
【図５３】



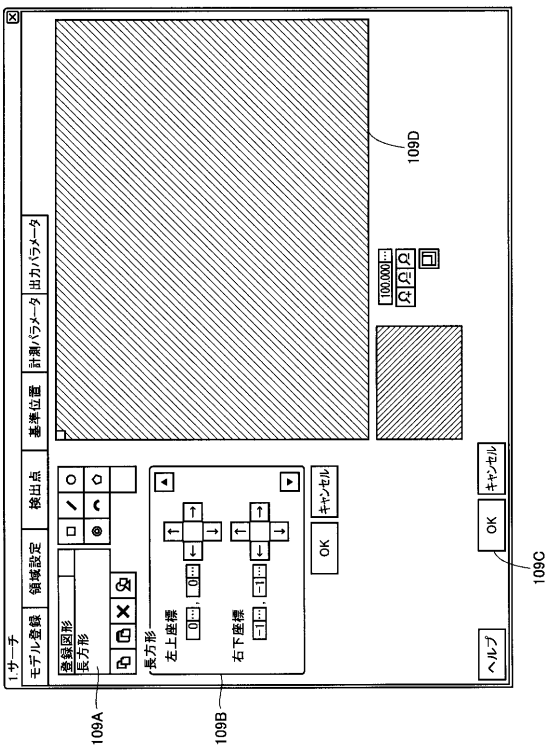
【図５４】



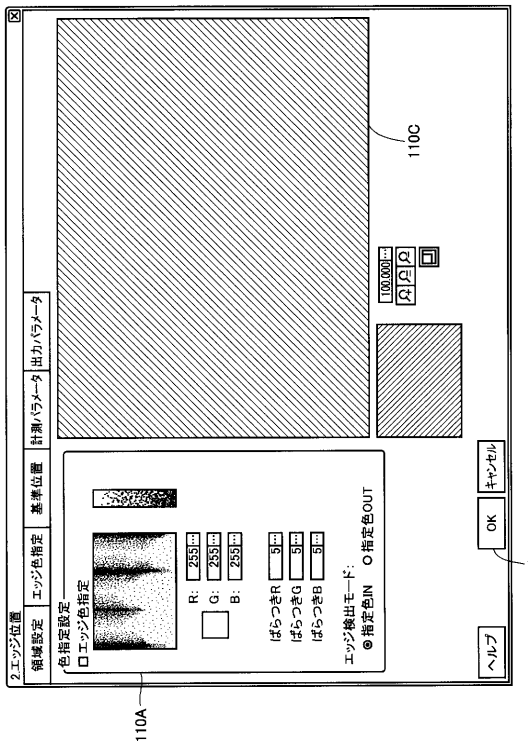
【図５５】



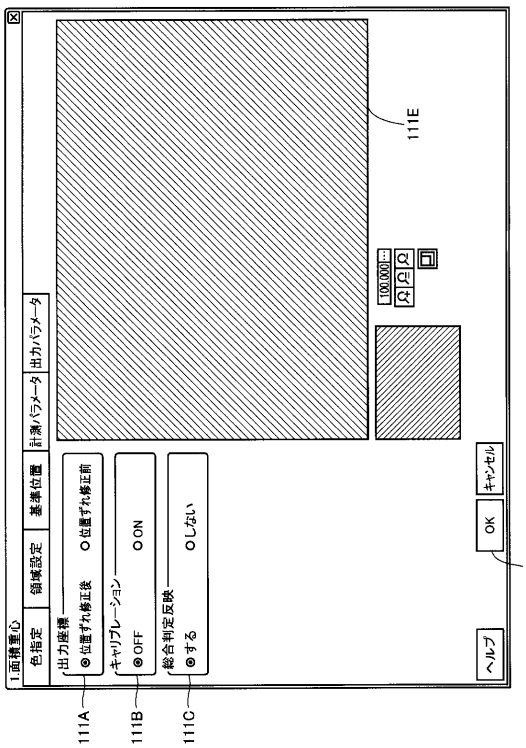
【図５６】



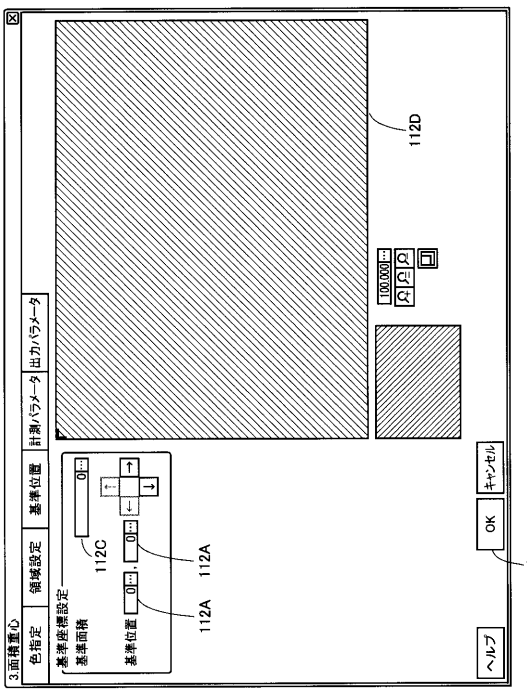
【図５７】



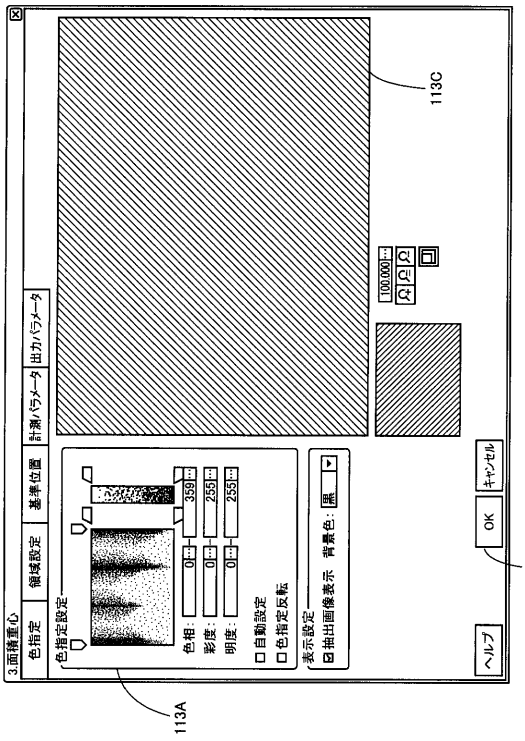
【図５８】



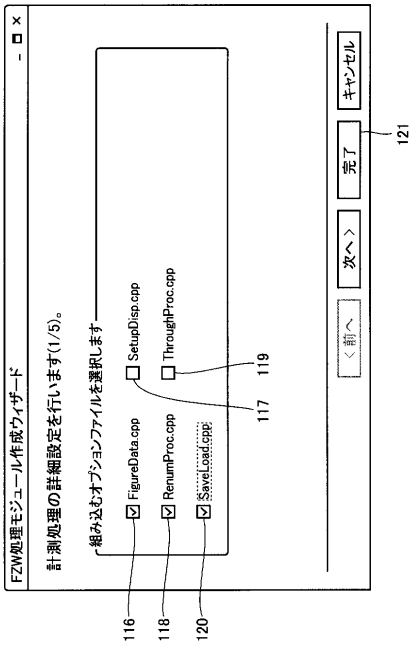
【図５９】



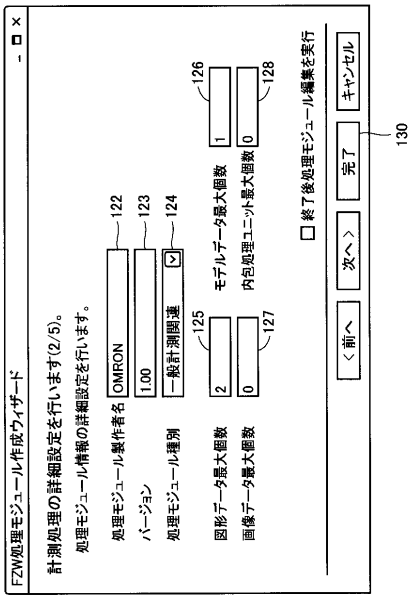
【図６０】



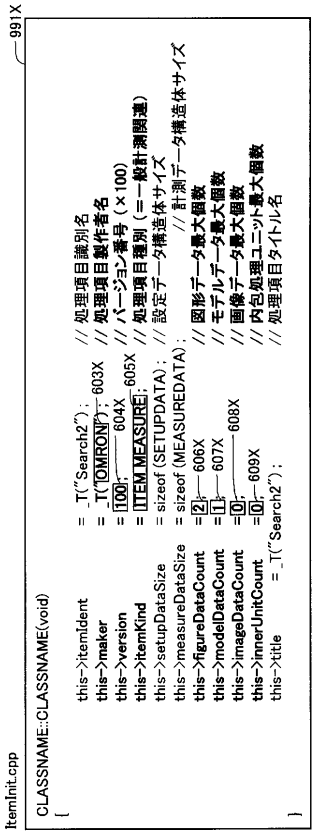
【図６１】



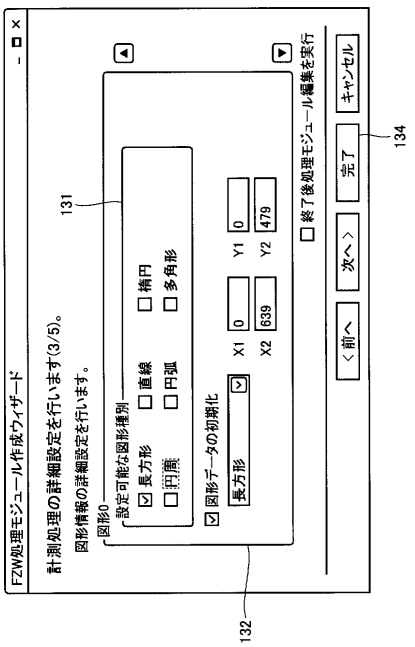
【図６２】



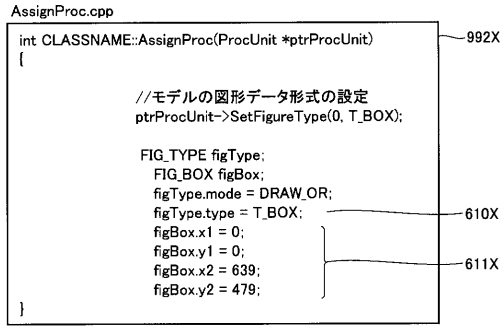
【図６３】



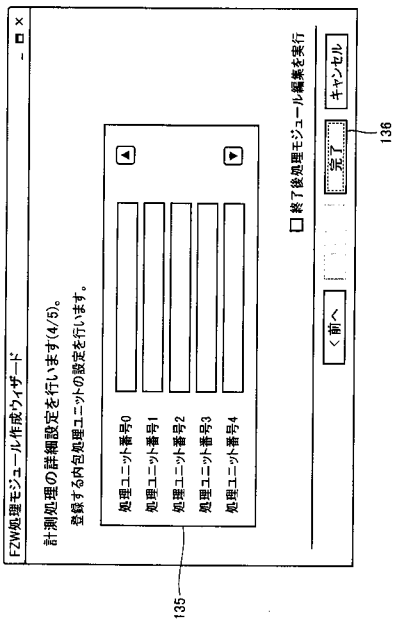
【図６４】



【図６５】



【図６６】



【図６７】

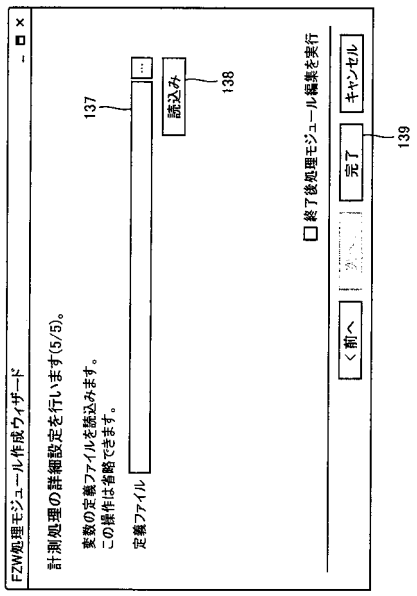
AssignProc.cpp : 処理ユニット番号0に"Calculation"を記入した場合

```
int CLASSNAME::AssignProc(ProcUnit *ptrProcUnit)
{

        //内包ユニットの設定
        ptrProcUnit->AssignInnerUnit(0, _T("Calculation"));
}
```

992X

612X

【図６８】



【図６９】

変数名	型	識別子	初期値	下限値	上限値	外部参照
判定	整数	Judge	0	-1	1	出力
基準座標X	実数	StandardX	320	0	9999	入出力
基準座標Y	実数	StandardY	240	0	9999	入出力
分散値	実数	Stddev	0	0	100	なし
…						

140 141 142 143 144 145 146

【図７０】

<読込むファイル：CSV形式>

```
変数名,型,識別子,初期値,下限値,上限値,外部参照
判定,整数,Judge,0,-1,1,出力
基準座標X,実数,StandardX,320,0,9999,入出力
基準座標Y,実数,StandardY,240,0,9999,入出力
分散値,実数,Stddev,0,0,100,なし
```

【図７１】

<ItemDefs.h：定義>

```
// 計測データ構造体定義
struct MEASUREDATA {
        int             Judge           // 判定
};

// 設定データ構造体定義
struct SETUPDATA {
        double          StandardX;      // 基準座標X
        double          StandardY;      // 基準座標Y
        double          Stddev;         // 分散値
};

※出力のみのパラメータは、計測データに振り分ける
  他は設定データとする
```

【図７２】

<AssignProc.cpp：初期化>

```
int CLASSNAME::AssignProc(ProcUnit *ptrProcUnit)
{
        SETUPDATA       *ptrSetupData;
        MEASUREDATA                     *ptrMeasureData;

        // 設定データの初期化
        ptrMeasureData = ptrProcUnit->GetMeasureData();
        ptrMesureData->Judge = 0;       // 判定

        ptrSetupData = ptrProcUnit->GetSetupData();
        ptrSetupData->StandardX = 320;                  // 基準座標X
        ptrSetupData->StandardY = 240;                  // 基準座標Y
        ptrSetupData->Stddev = 0.0;     // 分散値
}
```

【図７３】

<UnitData.cpp：外部からの変更>

```
typedef struct{
        int             no;
        TCHAR           *ident;
        double          low;
        double          high;
} TABLE;

static const TABLE nameTable[] = {
        // 計測データ
        {0,                             _T("Judge"),-1, 1},

        // 設定データ
        {120,                           _T("StandardX"), 0, 9999},
        {121,                           _T("StandardY"), 0, 9999},

        {-1, NULL, 0, 0}
};

int CLASSNAME::SetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        // 設定値の範囲チェック

        int errflg = 1;
        for (int i = 0; nameTable[i].no!= -1; ++i){
                if (dataNo == nameTable[i].no
                 && data->GetDVal() >= nameTable[i].low
                 && data->GetDVal() <= nameTable[i].high){
                        errflg = 0;
                        break;
                }
        }
        if (errflg== 1) return-1;

        // 設定データ設定

        SETUPDATA *ptrSetupData = ptrProcUnit->GetSetupData();

        switch (dataNo) {
        case 120:       // 基準座標X
                        ptrSetupData->StandardX = data->GetIVal();
                        break;
        case 121:       // 基準座標Y
                        ptrSetupData->StandardY = data->GetIVal();
                        break;
        default:
                        ret = -1;
                        break;
        }
        return(ret);
}
```

【図７４】

```
int CLASSNAME::GetUnitData(ProcUnit *ptrProcUnit, int dataNo, ANYTYPE *data)
{
        int ret = NORMAL;

        // 計測データ設定

        MEASUREDATA *ptrMeasureData = ptrProcUnit->GetMeasureData();

        // 設定データ設定

        SETUPDATA *ptrSetupData = ptrProcUnit->GetSetupData();

        switch (dataNo) {
        case 0:         // 判定
                        data->SetVal(ptrMeasureData->Judge);
                        break;

        case 120:       // 基準座標X
                        data->SetVal(ptrSetupData->StandardX);
                        break;
        case 121:       // 基準座標Y
                        data->SetVal(ptrSetupData->StandardY);
                        break;
        default:
                        ret = -1;
                        break;
        }
        return(ret);
}
```

フロントページの続き

(74)代理人 100109162
弁理士 酒井 將行
(74)代理人 100111246
弁理士 荒川 伸夫
(72)発明者 鈴木 勇治
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地 オムロン株式会社内
(72)発明者 井尻 隆史
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地 オムロン株式会社内
(72)発明者 吉浦 豪
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地 オムロン株式会社内
(72)発明者 満塩 孝史
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地 オムロン株式会社内

審査官 稲垣 良一

(56)参考文献 特開２０００－２００１９１（ＪＰ，Ａ）
特開２００４－３８８４４（ＪＰ，Ａ）
特開２００１－１５４８３４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／４４
Ｇ０６Ｆ ３／０４８
Ｇ０６Ｔ １／００