(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2009-193181
(P2009-193181A)
(43)公開日　平成21年8月27日(2009.8.27)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/44	(2006.01)	G06F	9/06	620B	5B176

審査請求 未請求　請求項の数 13　ＯＬ　（全 19 頁）

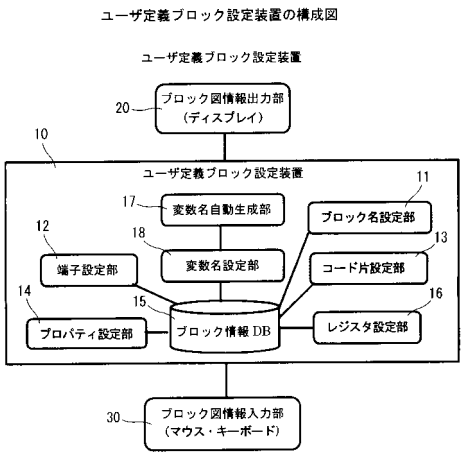
(21)出願番号　特願2008-31191 (P2008-31191)
(22)出願日　平成20年2月13日 (2008.2.13)

(71)出願人　000006105
株式会社明電舎
東京都品川区大崎２丁目１番１号
(74)代理人　100096459
弁理士　橋本　剛
(74)代理人　100104938
弁理士　鵜澤　英久
(72)発明者　宮津　章好
東京都品川区大崎２丁目１番１号　株式会社明電舎内
(72)発明者　海野　富士也
東京都品川区大崎２丁目１番１号　株式会社明電舎内

最終頁に続く

(54)【発明の名称】ソフトウェアの開発支援システム、支援方法およびこの方法のプログラム

(57)【要約】
【課題】システム組み込みソフトウェアの設計をユーザ定義ブロックの接続で行い、ブロック図からのソースコードの作成を確実、容易にする。
【解決手段】ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義するユーザ定義ブロック設定装置１０は、ブロック図で使用する変数情報や関数情報の宣言部と中間コードを組み合わせて当該ブロックのソースコードを生成する。ブロックのコード片で使用する変数を当該ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する。ブロックに設定するプロパティ値でブロックの出力を変化させる。
　この定義されたブロック図に対応させたプログラムを自動生成するプログラム自動生成装置を備える。
【選択図】図１



【特許請求の範囲】
【請求項１】
ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援システムであって、
前記ブロックを定義するユーザ定義ブロック設定装置は、
ブロックなどを置くブロック図で使用する変数情報や関数情報の宣言部と中間コードを組み合わせて当該ブロックのソースコードを設定するコード片設定部と、
ブロックに設定されるコード片の中でＣＰＵのレジスタの参照に置き換える構文を設定するレジスタ設定部と、
ブロックのコード片で使用する変数を当該ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する変数名自動生成部と、
ブロックに設定するプロパティ値でブロックの出力を変化させるプロパティ設定部と、
を備えたことを特徴とするソフトウェアの開発支援システム。
【請求項２】
ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援システムであって、
前記プログラムを自動生成するプログラム自動生成装置は、
定義されたブロックをブロック図に関連付けて表示するブロック－ブロック図関連情報設定部と、
前記ブロック間の結線を行い、さらに、この結線に対して割り当てた変数を一覧情報として設定し、接続元、接続先、割り当て変数を結線情報として設定する変数－結線関連情報設定部と、
前記ブロックと結線および変数より、コード片から中間コードを経由してソースコード（プログラム）を生成するプログラム自動生成部と、
を備えたことを特徴とするソフトウェアの開発支援システム。
【請求項３】
前記プログラム自動生成部は、変数の設定を行った後で、変数と線を関連付け、この関連つけでソースコード（プログラム）を生成することを特徴とする請求項２に記載のソフトウェアの開発支援システム。
【請求項４】
前記ブロック－ブロック図関連情報、変数－結線関連情報またはコード片に任意の文字列のコメントを入力するコメント設定部を備えたことを特徴とする請求項２に記載のソフトウェアの開発支援システム。
【請求項５】
前記ブロック－ブロック図関連情報、変数－結線関連情報はファイルなどに保存しておくブロック図情報保存部を備えたことを特徴とする請求項２～４のいずれか１項に記載のソフトウェアの開発支援システム。
【請求項６】
前記ファイルなどに保存しておくブロック－ブロック図関連情報、変数－結線関連情報をプログラムの自動生成に際して変数一覧情報、結線情報、ブロック図接続情報に戻すためのブロック図情報読込部を備えたことを特徴とする請求項２～５のいずれか１項に記載のソフトウェアの開発支援システム。
【請求項７】
ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援方法であって、
前記ブロック図を定義するユーザ定義ブロック設定装置は、

ブロックなどを置くブロック図で使用する変数情報や関数情報の宣言部と中間コードを組み合わせて当該ブロックのソースコードを設定するコード片設定ステップと、
ブロックに設定されるコード片の中でＣＰＵのレジスタの参照に置き換える構文を設定するレジスタ設定ステップと、
ブロックのコード片で使用する変数を当該ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する変数名自動生成ステップと、
ブロックに設定するプロパティ値でブロックの出力を変化させるプロパティ設定ステップと、
を備えたことを特徴とするソフトウェアの開発支援方法。
【請求項８】
ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援方法であって、
前記プログラムを自動生成するプログラム自動生成装置は、
定義されたブロックをブロック図に関連付けて表示するブロック－ブロック図関連情報設定ステップと、
前記ブロック間の結線を行い、さらに、この結線に対して割り当てた変数を一覧情報として設定し、接続元、接続先、割り当て変数を結線情報として設定する変数－結線関連情報設定ステップと、
前記ブロックと結線および変数より、コード片から中間コードを経由してソースコード（プログラム）を生成するプログラム自動生成ステップと、
を備えたことを特徴とするソフトウェアの開発支援方法。
【請求項９】
前記プログラム自動生成ステップは、変数の設定を行った後で、変数と線を関連付け、この関連つけでソースコード（プログラム）を生成することを特徴とする請求項８に記載のソフトウェアの開発支援方法。
【請求項１０】
前記ブロック－ブロック図関連情報、変数－結線関連情報またはコード片に任意の文字列のコメントを入力するコメント設定ステップを備えたことを特徴とする請求項８に記載のソフトウェアの開発支援方法。
【請求項１１】
前記ブロック－ブロック図関連情報、変数－結線関連情報はファイルなどに保存しておくブロック図情報保存ステップを備えたことを特徴とする請求項８～１０のいずれか１項に記載のソフトウェアの開発支援方法。
【請求項１２】
前記ファイルなどに保存しておくブロック－ブロック図関連情報、変数－結線関連情報をプログラムの自動生成に際して変数一覧情報、結線情報、ブロック図接続情報に戻すためのブロック図情報読込ステップを備えたことを特徴とする請求項８～１１のいずれか１項に記載のソフトウェアの開発支援方法。
【請求項１３】
請求項７～１２に記載のソフトウェアの開発支援方法における処理ステップを、コンピュータで実行可能に構成したことを特徴とするプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ソフトウェアの開発支援システムに係り、特にユーザ定義ブロックとして記述する情報処理機能に対応させたプログラムを自動生成する装置および方法に関する。
【背景技術】
【０００２】

監視制御や計測制御などのコンピュータ処理システムに組み込むアプリケーション・ソフトウェアの開発ではＵＭＬで設計図の記述を行うことが多い。このＵＭＬの設計図のうち、クラス図などからソースコードを出力し、残りのロジックに関する部分の実装を手作業で行う（例えば、非特許文献１参照）。

【０００３】

一方、組み込みソフトウェアの設計では、制御演算を司る部分の設計図については、従来からブロック図が用いられている。ブロック図の要素としてはブロックと線があり（図２参照）、これらが図面上で接続されることで、ソフトウェアとして実現したい内容を表現する。この設計図を元にして実装を行うにあたっては、実装者が設計図から設計情報を読み取りながら、手作業で実装が行われている。また、ＭＡＴＬＡＢ／Ｓｉｍｕｌｉｎｋに代表されるようなブロック図作成のためのアプリケーションでは、ブロック図を元にソースコードを出力するものもある（例えば、非特許文献２参照）。

【非特許文献１】明電時報、２００６年９・１０月、通巻３１０号、Ｎｏ．５「ソフトウェア開発総合支援システムにおける仕様作成支援機能」

【非特許文献２】ＭＡＴＬＡＢ／Ｓｉｍｕｌｉｎｋ（ｈｔｔｐ：／／ｗｗｗ．ｃｙｂｅｒｎｅｔ．ｃｏ．ｊｐ／ｍａｔｌａｂ／ｐｒｏｄｕｃｔｓ／ｐｒｏｄｕｃｔ＿ｌｉｓｔｉｎｇ／ｓｉｍｕｌｉｎｋ／）

【発明の開示】

【発明が解決しようとする課題】

【０００４】

ＭＡＴＬＡＢ／Ｓｉｍｕｌｉｎｋに代表されるような従来のブロック図作成のためのアプリケーションでは、ブロック図の配線情報を元にソースコードを出力する。しかし、組み込みソフトウェアでは、少ないリソースで所望のプログラムとして実現しなければならないため、出力されたソースコードそのものでは組み込みソフトウェアとして利用するのは困難であることが多い。

【０００５】

例えば、一般的なブロック図を用いて出力されたソースコードには、変数（グローバル変数、ローカル変数など）やレジスタなどを指定するような内容は含まれていない。そのため、ブロック図を元に実装を行う際には、手作業によりコーディング作業が行われることになる。これにより、実装時間が多くかかる。また、手作業によるコーディングではブロック図と実装が乖離しやすくなる。

【０００６】

また、ブロック図から出力されるソースコードはＣ言語であることがほとんどであり、他のプログラム言語やＣ言語でも特殊なものに対応するには、出力されたコードのすべてに対して実装者が変更の確認をする必要がある。

【０００７】

本発明の目的は、組み込みソフトウェアの設計として、ユーザ定義ブロックの接続をブロック図上で行い、ブロック図からのソースコードの作成を確実、容易にしたソフトウェアの開発支援システム、支援方法およびプログラムを提供することにある。

【課題を解決するための手段】

【０００８】

前記の課題を解決するため、本発明は以下のシステム、方法およびプログラムを特徴とする。

【０００９】

（システムの発明）

（１）ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援システムであって、

前記ブロックを定義するユーザ定義ブロック設定装置は、

ブロックなどを置くブロック図で使用する変数情報や関数情報の宣言部と中間コードを

組み合わせて当該ブロックのソースコードを設定するコード片設定部と、
ブロックに設定されるコード片の中でＣＰＵのレジスタの参照に置き換える構文を設定するレジスタ設定部と、
ブロックのコード片で使用する変数を当該ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する変数名自動生成部と、
ブロックに設定するプロパティ値でブロックの出力を変化させるプロパティ設定部と、
を備えたことを特徴とする。

【００１０】

（２）ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援システムであって、
前記プログラムを自動生成するプログラム自動生成装置は、
定義されたブロックをブロック図に関連付けて表示するブロック－ブロック図関連情報設定部と、
前記ブロック間の結線を行い、さらに、この結線に対して割り当てた変数を一覧情報として設定し、接続元、接続先、割り当て変数を結線情報として設定する変数－結線関連情報設定部と、
前記ブロックと結線および変数より、コード片から中間コードを経由してソースコード（プログラム）を生成するプログラム自動生成部と、
を備えたことを特徴とする。

【００１１】

（３）前記プログラム自動生成部は、変数の設定を行った後で、変数と線を関連付け、この関連つけでソースコード（プログラム）を生成することを特徴とする。

【００１２】

（４）前記ブロック－ブロック図関連情報、変数－結線関連情報またはコード片に任意の文字列のコメントを入力するコメント設定部を備えたことを特徴とする。

【００１３】

（５）前記ブロック－ブロック図関連情報、変数－結線関連情報はファイルなどに保存しておくブロック図情報保存部を備えたことを特徴とする。

【００１４】

（６）前記ファイルなどに保存しておくブロック－ブロック図関連情報、変数－結線関連情報をプログラムの自動生成に際して変数一覧情報、結線情報、ブロック図接続情報に戻すためのブロック図情報読込部を備えたことを特徴とする。

【００１５】

（方法の発明）

（７）ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援方法であって、
前記ブロック図を定義するユーザ定義ブロック設定装置は、
ブロックなどを置くブロック図で使用する変数情報や関数情報の宣言部と中間コードを組み合わせて当該ブロックのソースコードを設定するコード片設定ステップと、
ブロックに設定されるコード片の中でＣＰＵのレジスタの参照に置き換える構文を設定するレジスタ設定ステップと、
ブロックのコード片で使用する変数を当該ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する変数名自動生成ステップと、
ブロックに設定するプロパティ値でブロックの出力を変化させるプロパティ設定ステップと、
を備えたことを特徴とする。

【００１６】
（８）ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義し、この定義されたブロックに対応させたプログラムを自動生成するソフトウェアの開発支援方法であって、
前記プログラムを自動生成するプログラム自動生成装置は、
定義されたブロックをブロック図に関連付けて表示するブロック－ブロック図関連情報設定ステップと、
前記ブロック間の結線を行い、さらに、この結線に対して割り当てた変数を一覧情報として設定し、接続元、接続先、割り当て変数を結線情報として設定する変数－結線関連情報設定ステップと、
前記ブロックと結線および変数より、コード片から中間コードを経由してソースコード（プログラム）を生成するプログラム自動生成ステップと、
を備えたことを特徴とする。
【００１７】
（９）前記プログラム自動生成ステップは、変数の設定を行った後で、変数と線を関連付け、この関連つけでソースコード（プログラム）を生成することを特徴とする。
【００１８】
（１０）前記ブロック－ブロック図関連情報、変数－結線関連情報またはコード片に任意の文字列のコメントを入力するコメント設定ステップを備えたことを特徴とする。
【００１９】
（１１）前記ブロック－ブロック図関連情報、変数－結線関連情報はファイルなどに保存しておくブロック図情報保存ステップを備えたことを特徴とする。
【００２０】
（１２）前記ファイルなどに保存しておくブロック－ブロック図関連情報、変数－結線関連情報をプログラムの自動生成に際して変数一覧情報、結線情報、ブロック図接続情報に戻すためのブロック図情報読込ステップを備えたことを特徴とする。
【００２１】
（プログラムの発明）
（１３）上記（７）～（１２）に記載のソフトウェアの開発支援方法における処理ステップを、コンピュータで実行可能に構成したことを特徴とする。
【発明の効果】
【００２２】
以上のとおり、本発明によれば、ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロック図として定義でき、この定義されたブロックに対応させたプログラムを自動生成できるようにしたため、組み込みソフトウェアの設計として、ユーザ定義ブロックの接続をブロック図上で行い、このブロック図からのソースコードの作成が確実、容易になる。
【発明を実施するための最良の形態】
【００２３】
（実施形態１）
図１は、本実施形態を示すユーザ定義ブロック設定装置の構成図である。この装置は、ユーザがソフトウェアとして組み込もうとする任意の情報処理機能をブロックとして定義可能とし、これを「ユーザ定義ブロック」として効率よく作成できるようにする。なお、ここでは、対象となるプログラム言語はＣ言語を想定しているが、Ｃ言語以外のプログラム言語でも良い。また、このユーザ定義ブロックは、ブロック図上ではプログラム自動生成装置に固有の情報処理機能をブロックとして定義したものである基本ブロックと共通で扱えるものとして動作する。
【００２４】
図１において、ユーザ定義ブロック設定装置１０は、ユーザによって定義されたブロックのブロック名を設定するブロック名設定部１１と、ブロックの端子に名前を設定する端

子設定部１２と、ブロックにコード片（ブロックに対応する部分的なソースコード）を設定するコード片設定部１３と、ブロックのプロパティを設定するプロパティ設定部１４と、ブロック名設定部・端子設定部・コード片設定部・プロパティ設定部で設定した内容を保持するブロック情報ＤＢ（データベース）１５と、レジスタとレジスタ変数との割り当てを設定するレジスタ設定部１６と、重複を回避した変数名を生成する変数名自動生成部１７と、変数名自動生成した変数名をブロック情報ＤＢに設定する変数名設定部１８で構成する。このユーザ定義ブロック設定装置の外部入出力手段として、ブロック図およびブロックの設定情報を表示するブロック図情報出力部（デイスプレイ）２０と、ブロックの情報を入力するブロック図情報入力部（マウス・キーボード）３０を備える。

【００２５】

以下、「ユーザ定義ブロック」の作成手順を詳細に説明する。

【００２６】

（１）単純なブロックの定義

ここでは、ブロック、ブロック定義の方法、ブロックとソースコードとの関連付けについて説明する。

【００２７】

図２は単純なブロック図の定義例を示し、ブロックにはブロック名の設定と線の接続を行うことができる。ブロック図情報入力部３０にて入力されたブロック名の設定はブロック設定部１１で行い、ブロック図情報出力部２０に表示される。ブロックと線とのつなぎ目の部分を端子と呼ぶ。図２では入力端子が２つ、出力端子が１つのブロックを表している。この入力端子の一方を「ｉｎ１」、もう一方の入力端子を「ｉｎ２」、出力端子を「ｏｕｔ１」というように端子設定部１２にて各端子にブロック図情報入力部３０にて入力された名前をつけ、ブロック図情報出力部２０に表示される。

【００２８】

また、コード片（ブロックに対応する部分的なソースコード）は単体ではプログラム、またはその一部として使用することはできない。使用する変数の情報（変数宣言）、所属する関数の情報（関数宣言、実装）などと組み合わせることで、コード片は中間コード（例えば、後に示すリスト５など）に変換する。関数や変数などの宣言部と中間コードを組み合わせることでソースコードとして使用する。ブロック図情報入力部３０にて入力されたコード片はコード片設定部１３にて設定を行い、ブロック図情報出力部２０に表示される。

【００２９】

ブロックに関する情報であるブロック名、端子、コード片、プロパティ（プロパティについては後に説明する）はそれぞれ、ブロック名設定部１１、端子設定部１２、コード片設定部１３、ブロパティ設定部１４により設定され、これら情報はブロック情報ＤＢ（ファイル、メモリ上のデータなど）１５に保存される。

【００３０】

例えば、図３の（ａ）に示す乗算ブロックを作成するときは、同図の（ｂ）に示す情報設定を行う。この乗算ブロックのコード片では「＄ｉｎ１」というように、端子の名前に「＄」が付いている。これは変数との置き換えを表しており、「ｉｎ」に接続されている線の値（＝変数）を代入するという意味で用いるための記号である。ｉｎ１という端子に接続されている線には、このブロック以前に接続されている別のブロックの演算結果が入っており、その値はソースコード上では変数によって受け渡される。また、出力についても同様に、ｏｕｔ１に接続されている線に乗算ブロックの演算結果を出力する（＝変数に代入する）」という意味になる。

【００３１】

このように、ブロックに対して端子の名称とブロックに対応するコード片を設定することで、ユーザ定義ブロックとしてブロック図上で利用することができる。

【００３２】

例えば、図４のように乗算ブロックをブロック図上に置き、端子ｉｎ１に変数ＶＡＲ１

を示す線を接続する。同様に、端子ｉｎ２に変数ＶＡＲ２、端子ｏｕｔ１に変数ＲＥＳを示す線を接続する。このときにコード片から変換された中間コードは図４の（ｂ）のように、「ＲＥＳ＝ＶＡＲ１＊ＶＡＲ２；」となる。このリストは、＄ｉｎ１をＶＡＲ１に、＄ｉｎ２をＶＡＲ２に、ｏｕｔ１をＲＥＳに置き換えることでコード片から中間コードへの変換が完了する。

【００３３】

以上の手法により、ユーザ定義ブロックを定義することができる。また、ユーザ定義ブロックをブロック図上で使用することができる。

【００３４】

（２）変数を使用したブロックの定義

ここでは、レジスタの利用と、ブロック図からソースコードに変換したときに発生する変数名の重複について述べる。

【００３５】

（２－１）レジスタの利用

ブロックに設定されるコード片の中でレジスタを参照する構文を設定する。これは図１のレジスタ設定部１６にてＣＰＵ（中央演算処理装置）のレジスタとの関連付けを別途行うことにより、ＣＰＵのレジスタの参照に置き換える。同じレジスタを使用することで、他のブロックと変数を共用することができる。

【００３６】

例えば、図５の（ａ）に示すような加算ブロック（ＡＤＤ３）の定義を考える。このブロックは入力端子ｉｎ１～ｉｎ３のすべての値を足して出力端子ｏｕｔ１に出力する。同図の（ｂ）に示すリスト３にコード片を示す。リスト３にある＄ｒｅｇｉｓｔｅｒ１がレジスタ変数を表しており、＄ｒｅｇｉｓｔｅｒ１、＄ｒｅｇｉｓｔｅｒ２、…のように語尾の数字で個々のレジスタを区別する。

【００３７】

図６の（ａ）のように、ＡＤＤ３ブロックをブロック図上に置き、端子ｉｎ１に変数ＶＡＲ１、端子ｉｎ２に変数ＶＡＲ２を、端子ｉｎ３に変数ＶＡＲ３を、端子ｏｕｔ１に変数ＲＥＳを示す線を接続する。また、＄ｒｅｇｉｓｔｅｒ１はレジスタ変数の宣言文に変換する。このときにコード片から変換された中間コードは図６の（ｂ）に示すリスト４のようになる。

【００３８】

（２－２）変数名の重複

【００３９】

レジスタ変数の代わりに通常の変数を使用することもできる。これは関数内のローカル変数として使用する。各ブロックのコード片で変数を使用した場合には、変数名が重複することがある。このため、コード片を使用する際には変数名自動生成部１７にて各ブロック内で閉じた変数名として、ブロックごとにコード片の元の変数名に続いて重複しない任意の文字列を付加する。変数名自動生成部１７で生成した文字列を変数名として変数名設定部にて設定する。これにより変数名の重複を防ぎ、あるブロックで使用されている変数が他のブロックに影響を与えないようにする。

【００４０】

例えば、図７に示すリスト６（右）では、カウンタを用意して数字（０００１）を生成し、変数名のあとに文字列として付加している（ｖａｒ＿０００１）。変数名自動生成の方法についてはこれ以外でもよい。

【００４１】

変数を使用する場合、図５のＡＤＤ３ブロックのコード片は図７に示すリスト６の（左）のように変更することができる。また、変数名自動生成部により、コード片から変数名が生成された中間コードを出力した場合の例をリスト６の（右）に示す。

【００４２】

以上の手法により、コード片の中でレジスタを使用することができる。また、コード片

の中で変数を使用することができる。さらに、変数名自動生成部が他のブロックで使用している変数名と重複しないような変数名を生成するため、コード片の中で変数を使用するときに、複数のブロックの間で変数名が重複しなくなる。

【００４３】

（３）ブロックのプロパティ

ここでは、ブロックに対してプロパティが定義できることと、ブロパティへの設定値を変更することにより出力が変化することについて述べる。

【００４４】

ブロックに設定する情報をプロパティと呼ぶ。プロパティには名前とそれに設定できる値の種類（型）を指定する必要がある。値の種類には文字列、数値、２値（真偽値）、選択（複数ある値の候補の中から１つを選ぶ）のいずれかを設定することができる。ブロック名、端子、コード片はブロパティには含まれない。ブロック図情報入力部３０にて入力されたプロパティはプロパティ設定部１４で設定を行い、ブロック図情報出力部２０に表示される。

【００４５】

図８の（ａ）は加減算ブロックの定義とプロパティを示し、同図の（ｂ）に示すリスト７には加減算ブロックのコード片を示している。この加減算ブロックにおいて、ブロパティはｓｉｇｎ１、ｓｉｇｎ２の２つがあり、それぞれ真偽値を持つプロパティである。例えば、これらの値がｔｒｕｅのときには符号は“＋”、値がｆａｌｓｅのときには符号は“－”を出力するように設定する。プロパティを変更する際に、ブロック設定部１４によりＵＩ（ユーザインタフェース）を操作して変更するには、図９に示すそれぞれの型に対応したＵＩコンポーネントを使用する。例えば、文字列にはテキストボックスを、数値にはテキストボックスを、２値にはラジオボタンを、選択にはコンボボックスを使用する。

【００４６】

以下に、プロパティｓｉｇｎ１の値を“＋”から“－”に変更したときの中間コードの出力例を図１０にリスト８、リスト９として示す。プロパティｓｉｇｎ１，ｓｉｇｎ２を変更すれば、コード片を使用した場合の出力結果を変更することができる。

【００４７】

端子ｉｎ１に変数ＶＡＲ１、端子ｉｎ２に変数ＶＡＲ２、端子ｏｕｔ１に変数ＲＥＳを接続してあるものとし、図１０のリスト８ではプロパティｓｉｇｎ２は“＋”に設定されているものとする。このプロパティｓｉｇｎ１の値を“＋”から“－”に変更したときの中間コードはリスト９に示すようになる。

【００４８】

以上の手法により、ブロックにプロパティを設定することができる。また、ブロックのプロパティを変更することで、コード片から出力されるソースコードを変更することができる。

【００４９】

（４）コード片の任意プログラム言語への対応

上記の（１）～（３）では出力する内容はＣ言語を例としていた。しかし、コード片からソースコードに変換するにあたっては、ブロックに設定するコード片の内容を変更することで、任意のプログラム言語に対応することができる。図１１に示すリスト１０はＢＡＳＩＣで出力する場合のコード片の記述例である。

【００５０】

以上の手法により、コード片の記述を任意のプログラム言語に置き換えても、ユーザ定義ブロックとして有効である。

【００５１】

（実施形態２）

図１２は、本実施形態を示すプログラム自動生成装置の構成図であり、実施形態１で作成された「ユーザ定義ブロック」からのプログラムの自動生成を行う。

【００５２】

図１２において、プログラム自動生成装置４０は、変数の情報を保持する変数一覧情報ＤＢ４１と、結線の情報を保持する結線情報ＤＢ４２と、ブロックとブロック図の関連情報を保持するブロック図接続情報ＤＢ４３と、変数と結線の情報を関連付けるための変数一結線関連情報設定部４４と、ユーザが定義したブロックの情報を作成・保持するユーザ定義ブロック作成部４５と、ブロックとブロック図の関連情報を設定するためのブロック一ブロック図関連情報設定部４６と、変数一覧情報ＤＢ・結線情報ＤＢ・ブロック図接続情報ＤＢに保持されている情報を元にソースコードを生成するプログラム自動生成部４７を持つ。さらに、プログラム中にコメントを書き込むコメント設定部４８、プログラムの元になったブロック図の情報読込部４９とその情報の保存部５０を持つ。

【００５３】

ブロック図およびブロックの設定情報を表示するブロック図情報出力部２０と、ブロックの情報を入力するブロック図情報入力部３０は図１のものと共用のものである。

【００５４】

以下、プログラムの自動生成の手法を詳細に説明する。

【００５５】

（１）プログラムの自動生成（その１）

ブロック図からソースコードを生成するには、実施形態１による「ユーザ定義ブロック作成」に加えて、以下の３つの処理を行う。

【００５６】

（１－１）ブロックとブロック図との関連付け

まず、実施形態１に示すユーザ定義ブロック設定装置を使用して定義されたブロックをブロック図上に配置し、ブロック図情報出力部２０で表示する。

【００５７】

例えば、ブロック図情報入力部３０にてブロック図上に図１３の（ａ）のように配置された場合は、ブロック－ブロック図関連情報設定部４６により、ブロック図Ｘにブロック ＡとＢを関連付けて、表１に示すブロックＡとＢの情報をブロック図接続情報ＤＢ４３に設定する。このとき、ブロックの内容をブロック図情報保存部５０にて保存するために、ブロック図接続情報としてユーザ定義ブロック作成部にあるブロックの内容がブロック図接続情報ＤＢ４３に書き写される。各ブロックの情報には、実施形態１で設定したブロック名、端子、コード片などが含まれている。

【００５８】

【表１】

ブロック図接続情報

ブロック図名	ブロック
ブロック図Ｘ	ブロックＡの情報
ブロック図Ｘ	ブロックＢの情報

【００５９】

（１－２）変数一覧情報と結線との関連付け

上記の（１－１）にて図１３の（ａ）に示すブロックをブロック図上に配置した後に、ブロック図情報入力部３０にて図１３の（ｂ）ようなブロック間の結線を行う。さらに、ブロック図情報入力部３０にて結線に対して割り当てた変数の情報を表２の内容で変数一覧情報ＤＢ４１に設定する。変数一覧情報には変数ごとに型や変数種別などの層性情報を付加する。このとき、変数－結線関連情報設定部４４により表３の結線情報（接続元、接続先、割り当て変数）を結線情報ＤＢ４２に追加する。

【００６０】
【表２】

変数一覧情報

変数名	型	種別	初期値
VAR1	int	引数	ー
VAR2	int	グローバル変数	1
RES1	int	ローカル変数	0
RES2	int	戻り値	0

【００６１】
【表３】

接続情報

接続元	接続先	割り当てる変数
(VAR1)	ブロック A:in1	VAR1
(VAR2)	ブロック A:in2	VAR2
ブロック A:out1	ブロック B:in1	RES1
ブロック B:out1	(RES2)	RES2

【００６２】
（１－３）プログラム生成
「ユーザ定義ブロック」にて各ブロックにはブロックとコード片の関連付けがあらかじめ行われているので、ブロック図－ブロック（コード片）－結線－変数の流れを追うことで、コード片から中間コードを経由してソースコード（プログラム）を生成することができる。
【００６３】
例えば、上記の表３におけるブロックＡには図１４の（ａ）に示すリスト１にはブロックＡのコード片が関連付けられ、（ｂ）に示すリスト２にはブロックＢのコード片が関連付けられており、ブロック図上で図１３の（ａ）ように結線しているとする。このときに出力される中間コードは、リスト１、リスト２、表２、表３から図１５に示すリスト３のように出力する。中間コードの出力は図１３の（ｂ）の入力から出力方向へ順番に行う。
【００６４】
さらに、リスト３の中間コードに対して、プログラム自動生成部４７でＣ言語のソースコードに変換する場合、変数宣言、Ｃ言語としてコンパイルするための関数、変数宣言などを付加し、図１６に示すリスト４のソースコードを得る。
【００６５】
この場合、表２の変数種別から、ＶＡＲ１は関数の引数、ＶＡＲ２はグローバル変数、ＲＥＳ１はローカル変数、ＲＥＳ２は関数の戻り値として宣言および使用している。また、変数の種別がレジスタ変数の場合は変数宣言の方の前にキーワード“ｒｅｇｉｓｔｅｒ”を足す。関数名については、例えば文字列“ｆｕｎｃ”とシステム内に持ったカウンタで生成した数字を文字列として付加したものとする。ただし、これ以外の方法でもよい。
【００６６】
以上の手法により、変数一覧情報ＤＢ、結線情報ＤＢ、ブロック図接続情報ＤＢの情報に基づいて、ソースコードを自動生成することができる。
【００６７】

（２）プログラムの自動生成（その２）

プログラムの自動生成（その１）での変数を設定する機能では、その場で設定した変数を線に割り当てた。本例では、既に定義されている変数を線に設定する。

【００６８】

前記の図１３（ａ）にあるブロックをブロック図上に配置する前に、ブロック図情報入力部３０により変数の情報を前記の表２の内容で変数一覧情報ＤＢ４１に設定する。次にブロック図情報入力部３０にて図１３の（ｂ）のような結線を行い、既に定義している変数を線に対して設定する。このとき、変数－結線関連情報設定部４４により前記の表３の情報を結線情報ＤＢ４２に追加する。

【００６９】

したがって、変数の設定を行った後で、変数と線を関連付けることができる。

【００７０】

（３）コメントを付加したプログラムの生成

ブロック図上でブロックに対してコメントを付加するために、下記の表４のようにブロック図接続情報にコメントの項目を増やす。ブロック図情報入力部３０から入力を行い、コメント設定部４８によりブロック図接続情報のコメントの項目について任意の文字列を設定する。また、結線情報についても下記の表５のようにコメントの項目を増やし、ブロック図接続情報と同様にコメント設定部４８にてコメントを設定する。

【００７１】

【表４】

コメントを付加したブロック図接続情報

ブロック図名	ブロック	コメント
ブロック図Ｘ	ブロックＡ	ブロックＡのコメント
ブロック図Ｘ	ブロックＢ	ブロックＢのコメント

【００７２】

【表５】

コメントを付加した結線情報

接続元	接続先	割り当てる変数	コメント
(VAR1)	ブロック A:in1	VAR1	線 VAR1－ブロック A:in1 のコメント
(VAR2)	ブロック A:in2	VAR2	線 VAR2－ブロック A:in2 のコメント
ブロック A:out1	ブロック B:in1	RES1	線ブロック A:out1－ブロック B:in1 のコメント
ブロック B:out1	(RES2)	RES2	線ブロック B:out1－RES2 のコメント

【００７３】

プログラム生成の際に、上記で追加したブロック図接続情報ＤＢ４３に設定したコメント、結線情報ＤＢ４２に設定したコメント、およびブロックに関連付けられたコード片に埋め込まれたコメントを出力する。ブロック図接続情報ＤＢ４３のコメントはそのブロックに対応する行の上に付加する。結線情報ＤＢ４２のコメントは変数宣言に続いて付加する。ただし、引数は宣言の前の行に追加する。

【００７４】

例として、図１６に示すリスト４に対して上配の表４、表５のようなコメントを付加した場合のソースコードを図１７にリスト５で示す。例えば、ＶＡＲ２はグローバル変数なので、変数宣言の横にコメントが追加される。ＶＡＲ１は引数なので、関数宣言の前の行にコメントが追加される。ブロックＡについてはリスト１にあるコード片の前の行にコメントが追加される。

【００７５】

また、コード片に埋め込まれたコメントについては、コード片から中間コードに変換する際に行う端子名の展開を適用する。例えば、リスト１のコード片を図１８の（ａ）に示すリスト６のように変更した場合、リスト６のコード片を中間コードに変換したものを図１８の（ｂ）にリスト７で示す。

【００７６】

したがって、ブロックまたは変数にコメントをつけたコメントつきのソースコードを自動生成することができる。

【００７７】

（４）ブロック図の情報保存

上記の（１）から（３）までによって、プログラム自動生成は行うことができるが、組み込みソフトウェア開発支援システムを終了すると、それまでに作成したブロック図が破棄されてしまう。作成したブロック図の情報（ブロック図接続情報ＤＢ、結線情報ＤＢ、変数一覧情報ＤＢ）はブロック図情報保存部５０でファイルなどに保存する。処理の手順を図１９の（ａ）に示す。保存したファイルなどはブロック図情報読込部４９にてブロック図情報として読み込まれる。処理の手順を図１９の（ｂ）に示す。

【００７８】

したがって、ブロック図をファイルなどに保存することができ、このファイルなどに保存されているブロック図をプログラムの自動生成に際して読み込むことができる。

【００７９】

なお、本発明は、図１や図１２に示した装置の一部又は全部の処理機能をプログラムとして構成してコンピュータに実行させることができる。

【図面の簡単な説明】

【００８０】

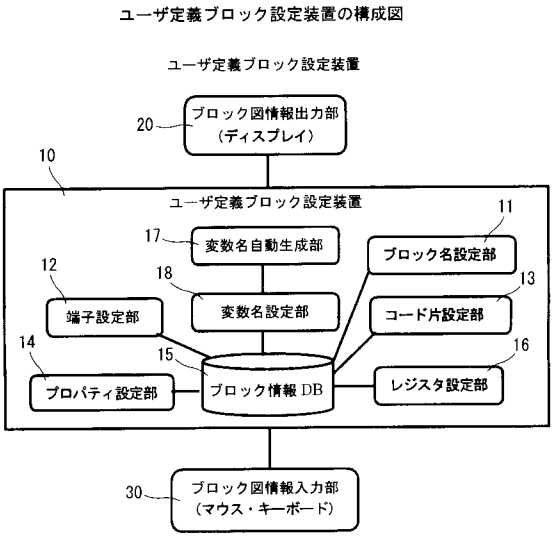
【図１】本発明の実施形態を示すユーザ定義ブロック設定装置の構成図。
【図２】ブロック図の定義例。
【図３】乗算ブロックの定義とコード片の例。
【図４】乗算ブロックの入出力とコードへの変換例。
【図５】ＡＤＤ３ブロックの定義とコード片の例。
【図６】ＡＤＤ３ブロックの中間コードの例。
【図７】ＡＤＤ３ブロックのコード片と中間コードの例。
【図８】加減算ブロックの定義とコード片の例。
【図９】ユーザインタフェースコンポーネントの例。
【図１０】中間コードの出力例。
【図１１】出力結果がＢＡＳＩＣの例。
【図１２】プログラム自動生成装置の構成図。
【図１３】ブロック図と結線後のブロック図の例。
【図１４】ブロックＡ、Ｂのコード片の例。
【図１５】ブロックＡ、Ｂの中間コードの例。
【図１６】ブロックＡ、Ｂのソースコードの例。
【図１７】コメント付きソースコードの例。
【図１８】コメント付きのコード片と中間コードの例。
【図１９】ブロック情報の保存／読み込み処理手順。

【符号の説明】

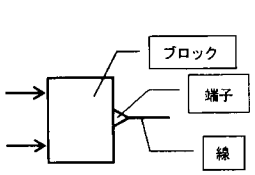
【００８１】

１０　ユーザ定義ブロック設定装置
２０　ブロック図情報出力部
３０　ブロック図情報入力部
４０　プログラム自動生成装置

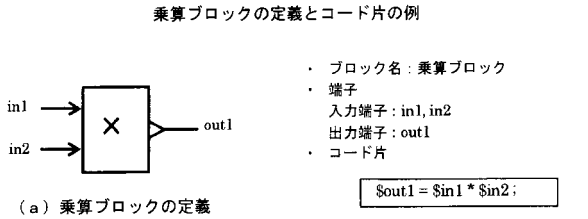
【図１】



【図２】



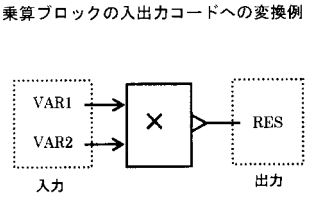
【図３】



（ａ）乗算ブロックの定義

（ｂ）リスト１．乗算ブロックのコード片

【図４】



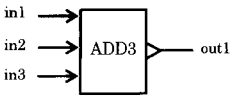
（ａ）乗算ブロックの入出力

```
RES = VAR1 * VAR2;
```

（ｂ）リスト２．ブロック使用によるコード片からのコードへの変換例

【図５】

ＡＤＤ３ブロックの定義とコード片の例



（ａ）ＡＤＤ３ブロックの定義

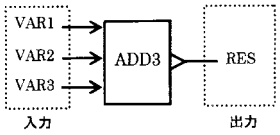
- ブロック名：ＡＤＤ３ブロック
- 端子
  入力端子：in1, in2, in3
  出力端子：out1
- コード片

```
$register1 long;
$register1 = $in1 + $in2 ;
$out1 = $resigter1 + $in3 ;
```

（ｂ）リスト３．レジスタ使用のＡＤＤ３ブロックのコード片

【図６】

ＡＤＤ３ブロックの中間コードの例



（ａ）ＡＤＤ３ブロックの入出力

```
register long REG1;
REG1 = VAR1 + VAR2 ;
RES = REG1 + VAR3 ;
```

（ｂ）リスト４．ＡＤＤ３ブロックの中間コード

【図７】

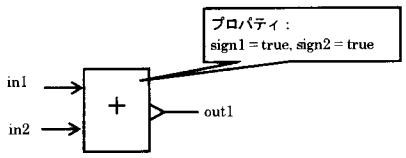
ＡＤＤ３ブロックのコード片と中間コードの例

```
long var;
var = $in1 + $in2 ;
$out1 = var + $in3 ;
```

```
long var_0001;
var_0001 = VAR1 + VAR2 ;
RES = var_0001 + VAR3 ;
```

【図８】

加減算ブロックの定義とコード片の例



（ａ）加減算ブロックの定義

- ブロック名：加減算ブロック
- 端子
  入力端子：in1, in2,
  出力端子：out1
- プロパティ：
  - 名前：sign1、型：２値、値：true
  - 名前：sign2、型：２値、値：true
- コード片

```
$out1 = $sign1 $in1 + $sign2 $in2 ;
```

（ｂ）リスト７．加減算ブロックのコード片

【図１０】

中間コードの出力例

・sign1 ＝“＋”

```
RES = + VAR1 + + VAR2 ;
```

（ａ）リスト８．sign1が“＋”の場合の加減算ブロックの中間コード

・sign1 ＝”－”

```
RES = － VAR1 + + VAR2 ;
```

（ｂ）リスト９．sign1が“－”の場合の加減算ブロックの中間コード

【図１１】

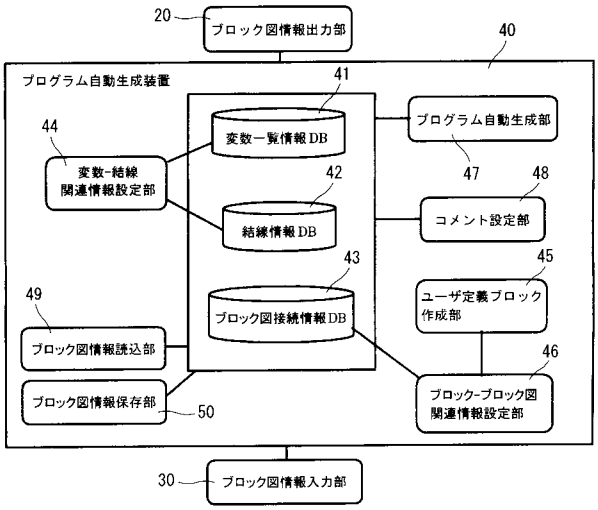
出力結果がBASICの例

```
10 var = $in1 + $in2
20 $out1 = var + $in3
```

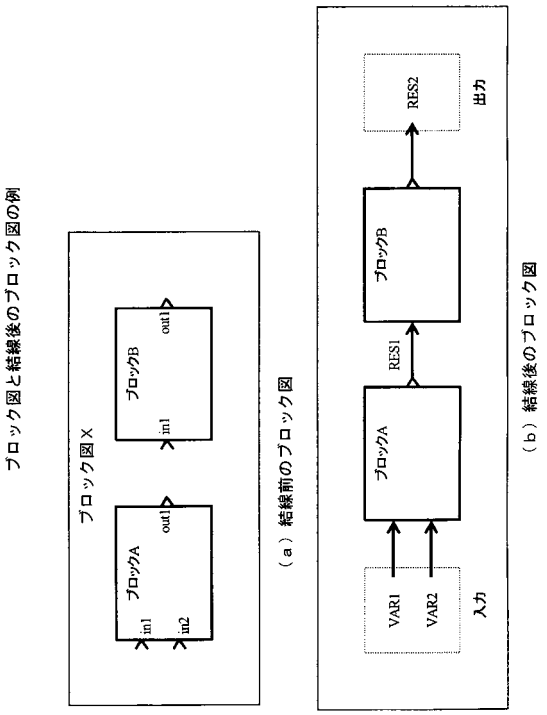
リスト１０．出力結果がBASICの場合

【図１２】

プログラム自動生成装置の構成図



【図１３】



【図１４】

ブロックＡ，Ｂのコード片の例

```
$out1 = $in1 * $in2;
```

（ａ）リスト１．ブロックＡのコード片

```
if($in1<0){
  $out1 = $in1 * (-1);
}else{
  $out1 = $in1;
}
```

（ｂ）リスト２．ブロックＢのコード片

【図１５】

ブロックＡ，Ｂの中間コードの例

```
RES1 = VAR1 * VAR2;
if(RES1<0){
  RES2 = RES1 * (-1);
}else{
  RES2 = RES1;
}
```

リスト３．中間コード

【図１６】

ブロックＡ，Ｂのソースコードの例

```
int VAR2 = 1;

 ...

int  func01(int VAR1){
  int  RES1 = 0
  int  RES2 = 0;

  RES1 = VAR1 * VAR2;
  if(RES1<0){
    RES2 = RES1 * (-1);
  }else{
    RES2 = RES1;
  }
  return RES2;
}
```

【図１７】

コメント付きソースコードの例

```
int VAR2 = 1 ;  /* 線 VAR2－ブロック A:in2 のコメント */

 ・・・


/* VAR1 : 線 VAR1－ブロック A:in1 のコメント */
int  func01 ( int VAR1){
  int  RES1 = 0 ; /*線ブロック A:out1－ブロック B:in1 のコメント */
  int  RES2 = 0 ; /*線ブロック B:out1－RES2 のコメント */

  /* ブロック A のコメント */
  RES1 = VAR1 * VAR2;
  /* ブロック B のコメント */
  if ( RES1<0 ) {
    RES2 = RES1 * (-1) ;
  } else {
    RES2 = RES1 ;
  }

  return RES2 ;
}
```

【図１８】

コメント付きのコード片と中間コードの例

```
$out1 = $in1 * $in2 ;  /* $in1 と $in2 */
```

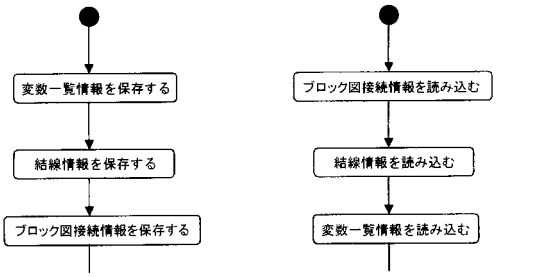
（ａ）リスト６．コメント付きのコード片

```
RES1 = VAR1 * VAR2 ;  /* VAR1 と VAR2 */
```

（ｂ）コメント付きの中間コード

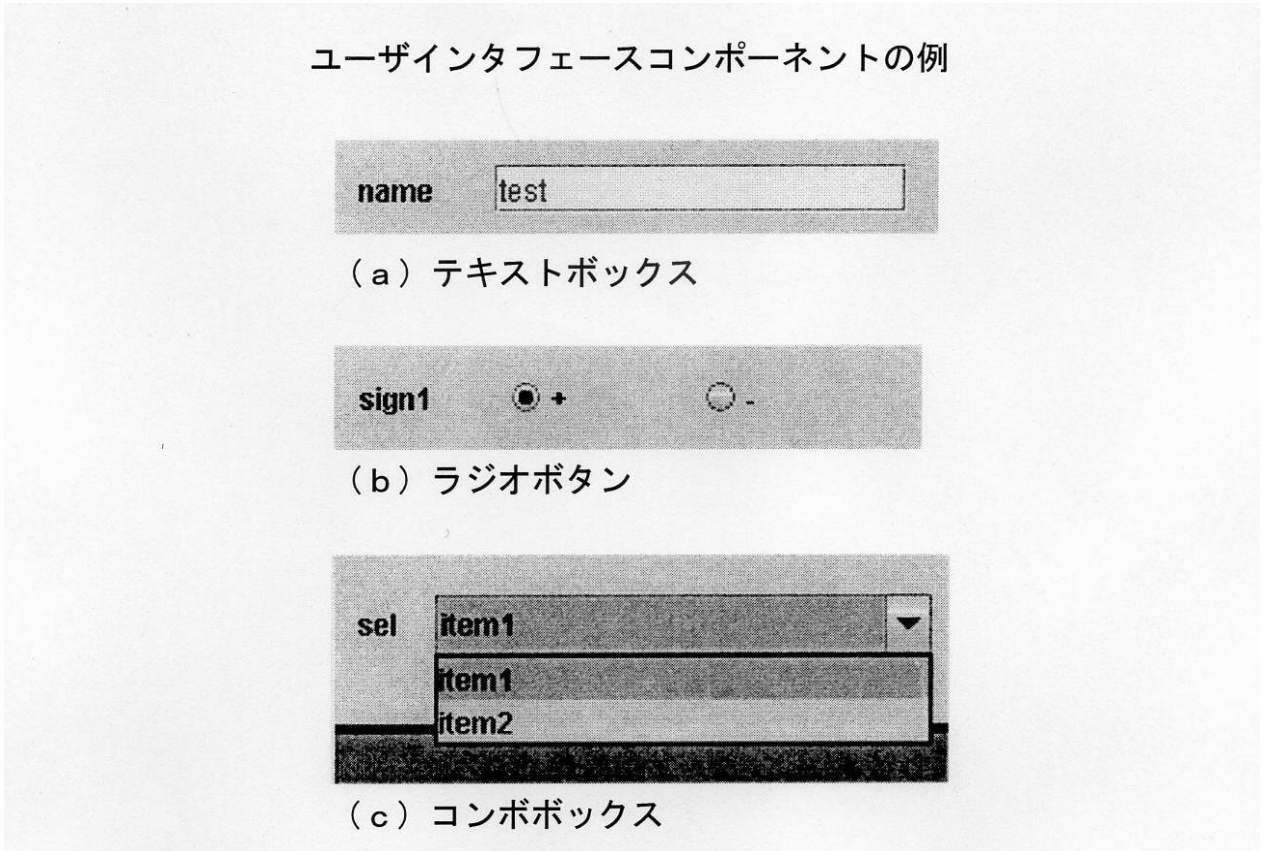
【図１９】

ブロック情報の保存/読み込み処理手順



（ａ）ブロック情報の保存処理　（ｂ）ブロック情報の読み込み処理

【図９】



フロントページの続き

(72)発明者　船津　有
東京都品川区大崎２丁目１番１号　株式会社明電舎内
(72)発明者　外山　達斎
東京都品川区大崎２丁目１番１号　株式会社明電舎内

Ｆターム(参考) 5B176 DA04　DD04　DE00　EC07