(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 December 2000 (07.12.2000)

PCT

(10) International Publication Number
WO 00/73885 A1

(51) International Patent Classification[7]: G06F 3/00

(21) International Application Number: PCT/US00/13635

(22) International Filing Date: 17 May 2000 (17.05.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/137,030 1 June 1999 (01.06.1999) US
60/155,751 23 September 1999 (23.09.1999) US

(71) Applicant: THE FOXBORO COMPANY [US/US]; 33 Commercial Street, Foxboro, MA 02035 (US).

(72) Inventor: LINSCOTT, Richard, L.; 2 Oakridge Drive, Plainville, MA 02762 (US).

(74) Agents: POWSNER, David, J. et al.; Nutter, McClennen & Fish, LLP, One International Place, Boston, MA 02110-2699 (US).

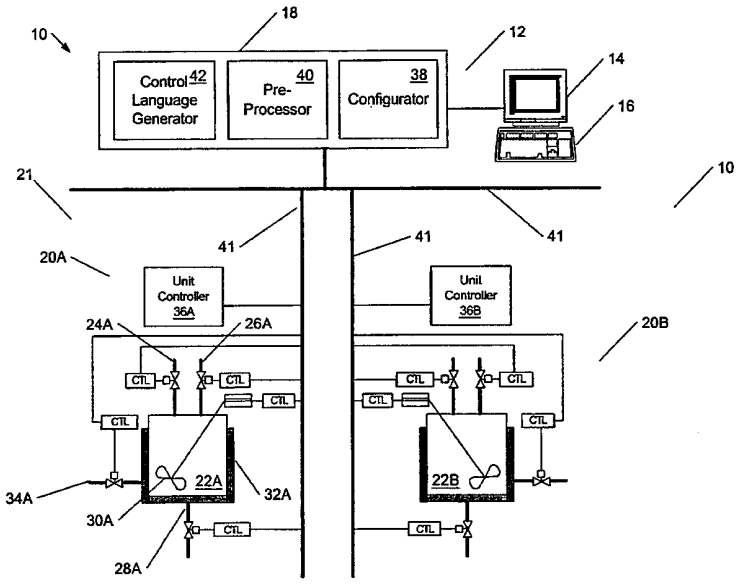
(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *With international search report.*
— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

(54) Title: SYSTEMS AND METHODS FOR LINKING PARAMETERS FOR THE CONFIGURATION OF CONTROL SYSTEMS




(57) Abstract: Improved methods and apparatus for process, environmental, industrial and other control systems extend the scope of parameters (scope of parameters), variables or other constructs (collectively, "parameters") beyond the objects in which they are declared. This is effected by providing a common scope for the naming of parameters in selected branches of a hierarchical model used to configure (38) the systems. This facilitates sharing information among the objects and, thereby, configuring (38) the controlled system.

WO 00/73885 A1

# SYSTEMS AND METHODS FOR LINKING PARAMETERS FOR THE CONFIGURATION OF CONTROL SYSTEMS

## Reference to Related Applications

This application claims the benefit of priority of U.S.S.N. 60/137,030, filed June 1, 1999, entitled LANGUAGE PARAMETER REFERENCES AND AUTO-LINKAGES and U.S.S.N. 60/155,751, filed September 23, 1999, entitled LANGUAGE PARAMETER REFERENCES AND AUTO-LINKAGES, the teachings of which are incorporated herein by reference.

This application is related to copending, commonly assigned U.S.S.N. 09/448,845, filed November 23, 1999, entitled METHODS AND APPARATUS FOR CONTROLLING OBJECT APPEARANCE IN A PROCESS CONTROL CONFIGURATION SYSTEM (Attorney Docket: 102314-50), U.S.S.N. 09/448,223, filed November 23, 1999 entitled PROCESS CONTROL CONFIGURATION SYSTEM WITH CONNECTION VALIDATION AND CONFIGURATION (Attorney Docket: 102314-54), and U.S.S.N. 09/448,374, filed November 23, 1999, entitled PROCESS CONTROL CONFIGURATION SYSTEM WITH PARAMETERIZED OBJECTS (Attorney Docket: 102314-46), the teachings of which are incorporated herein by reference.

## Background of the Invention

The invention pertains to control and, more particularly, to methods and apparatus for configuring control systems.

The terms "control" and "control systems" refer to the control of devices or systems by monitoring their characteristics. This is used to insure that output, processing, quality and/or efficiency remain within desired parameters over the course of time. In many control systems, digital data processing or other automated apparatus monitor a device, process or system and automatically adjust its operational parameters. In other control systems, such

apparatus monitor the device, process or system and display alarms or other indicia of its characteristics, leaving responsibility for adjustment to the operator.

Control is used in a number of fields. Process control, for example, is typically employed in the manufacturing sector for process, repetitive and discrete manufactures, though, it also has wide application in utility and other service industries. Environmental control finds application in residential, commercial, institutional and industrial settings, where temperature and other environmental factors must be properly maintained. Control is also used in articles of manufacture, from toasters to aircraft, to monitor and control device operation.

Modern day control systems typically include a combination of field devices, control devices, and controllers, the functions of which may overlap or be combined. Field devices include temperature, flow and other sensors that measure characteristics of the device, process or system being controlled. Control devices include valves, actuators, and the like, that control the device, process or system itself. Controllers generate settings for the control devices based on measurements from the field devices. Controller operation is typically based on a "control algorithm" that maintains a controlled system at a desired level, or drives it to that level, by minimizing differences between the values measured by the sensors and, for example, a setpoint defined by the operator.

Since they are designed to execute control strategies, controllers typically have neither the computing power nor user interfaces required to facilitate the design of a control algorithm. Instead, the art has developed configurators. These are typically general purpose computers (e.g., workstations) running software that permit an engineer or operator to graphically model a device, process or system and the desired strategy for controlling it. This includes enumerating field devices, control devices, controllers and other apparatus that will be used for control, specifying their interrelationships and the information that will be transferred among them, as well as detailing the calculations and methodology they will apply for purposes of control. Once modeling is complete and tested, the control algorithm is downloaded to the components of the control system.

One well known process control system configurator is that provided with the I/A Series® (hereinafter, "IAS" or "I/A") systems, marketed by the assignee hereof. That configurator has a graphical interface permitting an engineer to model a process hierarchically and to define a control algorithm from that hierarchy.

Though prior art process control configuration systems, particularly, for example, those sold by the assignee hereof, have met wide acceptance in the industry, there remains room for improvement. Such is the case, for example, with respect to defining parameters and other data within the modules of a process control hierarchy.

Traditionally, configurators have treated such modules as independent units, or "black boxes." Apart from parameters defined for explicit exchange between individual modules (for example, measurements passed from a sensor to a control block), user-defined parameters are not shared within the system. This is particularly true of those parameters that are downloaded to the controllers and other components of the control system.

To the extent that sharing occurs, the prior art requires explicit declaration of parameters that are to be accessed by multiple modules. Thus, for example, a sensor module declares for output measurement parameters that are passed to an associated control module. The latter, in turn, declares those measurements as inputs. Alternatively, or in addition, the prior art requires that path names or other identifying information be appended to externally accessed parameters. The control module, in the example, might access a measurement parameter by appending to its name that of the sensor module in which it is generated.

In this context, an object of the present invention is to provide improved methods and apparatus for control and, particularly, for configuring control systems. A related object of the invention is to provide such methods and apparatus for configuring process control systems.

A further object of the invention is to provide such methods and apparatus as facilitate defining parameters and other data values used to model and configure field devices, control devices, controllers and other components of a control system.

A still further object is to provide such methods and apparatus as are adaptable to a range of hierarchical and other control models.

Yet a still further object of the invention is to provide such methods and apparatus as can be implemented in a variety of digital data processing environments.

**Summary of the Invention**

The foregoing are among the objects attained by the invention, which provides improved methods and apparatus for modeling process, environmental, industrial and other control systems with hierarchies of modules, blocks, objects, or other data and/or programming constructs (collectively, "objects"). The improvement, according to one aspect of the invention, extends the effect of parameters, variables or other constructs (collectively, "parameters") by providing a common scope for naming them in selected branches, rather than merely in individual objects, of the modeling hierarchy. This facilitates sharing information among the objects and, thereby, configuring the controlled system.

According to these aspects of the invention, a branch of the hierarchy can be designated to support extended scope parameters, e.g., by a dialog box or other option. A parameter named or declared in one object in that branch can be referred to and shared by other objects in that branch, without redeclaration in those other objects and without explicit reference to the object in which the parameter was initially declared. For example, a parameter ABC101 can be declared in a first object, with its value set as the sum of two values. A second object in the same branch of the hierarchy can refer to ABC101 without redeclaring it. This facilitates, for example, calculations based on ABC101 in that second object, without the requirement that the user identify or declare the source of that parameter.

Related aspects of the invention provide improved apparatus as described above in which the parameters for which scope is extended are of the type downloaded to the process, environmental, industrial or other control system. Those parameters, moreover, can be user-defined parameters of the type used in source code that defines the behavior or other attributes of the objects and/or the entities they represent.

To continue the above example, the parameter ABC101 can be a user-defined parameter declared in source code contained in a unit operation-level block that is downloaded to a corresponding programmable controller to control runtime operation at that level. This same parameter can be accessed in source code contained in a control module-

level block that descends from the unit operations block. Unlike the prior art, which requires redeclaration of ABC101 in the control module block (or naming of the unit operation-level block as its source), no such declaration is required in a system according to this aspect of the invention.

Further aspects of the invention provide methods, as described above, in which the objects that form the hierarchical model represent entities within the any of (i) the control system, (ii) a system controlled by the control system, (iii) a control level hierarchy, and (iv) the apparatus for configuring the control system. The aforementioned parameters can, correspondingly, relate to characteristics associated with entities represented by the respective objects.

Further objects of the invention provide improved apparatus as described above specifically directed to process control. The control hierarchy modeled by objects of such an apparatus can include two or more levels from the industry-standard SP88 modeling hierarchy, e.g., an area level, a process cell level, a unit operation level, an equipment level and/or a control module level.

Still further objects of the invention provide an apparatus for configuring a control system which includes the a configurator that generates the aforementioned models. It can also include a control language generator that downloads, to the control system, control language codes that are based on the model. The improvements described above can be included in a preprocessor that works in conjunction with the configurator and/or the control language generator to extend parameter scope in the manner discussed.

Yet still further objects of the invention provide methods for configuring process and other control systems paralleling the foregoing.

These and other objects of the invention are evident in the drawings and in the description that follows.

**Brief Description of the Drawings**

A more complete understanding of the invention may be attained by reference to the drawings, in which:

Figure 1 depicts a controlled system, a control system and a configuration system according to the invention;

Figure 2 depicts a model generated by a configurator in a system according to the invention;

Figures 3 and 6 illustrate parameter scope extension in a system according to the invention;

Figure 4 depicts a method of operation of a preprocessor in a system according to the invention; and

Figure 5 depicts a preprocessor and its interaction with other system components in a system according to the invention.

**Detailed Description of the Illustrated Embodiment**

Figure 1 depicts a process control system 10 of the type in which apparatus and methods according to the invention are practiced. Though illustrated and described below for use in process control, those skilled in the art will appreciate that apparatus and methods according to the invention can be used in connection any industrial, manufacturing, service, environmental or other process, device or system amenable to monitoring or control (hereinafter, collectively, "control").

The system of Figure 1 includes a digital data processor 12 having monitor 14 (and/or other output devices), keyboard 16 (and/or other input devices), and computer 18 having a central processing unit, storage and input/output functionality (not shown). Digital data processor 12 represents an engineering workstation, personal computer, mainframe computer or other digital data processing device suitable for modeling a control system and generating control language for the configuration thereof in accord with the teachings herein. Though illustrated as being carried out on a physically distinct digital data processor 12, those skilled in the art will appreciate that the modeling and configuration functions described herein can be executed on suitably configured controller or other process control equipment.

The digital data processor 12 is coupled to, and arranged for controlling, a process 21 for mixing materials in units 20A, 20B. In this regard, though described mainly hereinafter as generating models and configuring the process 21, the unit 12 also forms part of that system, e.g., carrying out runtime control functions in support of the unit, field, and/or other controllers described below. Unit 20A comprises mixing chamber 22A, fluid inlets 24A, 26A, fluid outlet 28A, paddle 30A, cooler 32A, cooler inlet 34A. Inlet and outlet valves are provided as illustrated, all operating under control of field device controllers, labeled CTL, also as illustrated. Unit 20B is similarly constructed, as illustrated. Unit controllers 36A and 36B are provided, as illustrated, for monitoring and controlling operations of the respective units 20A, 20B, e.g., via their respective field device controllers.

Unit controllers 36A, 36B and field device controllers CTL are constructed and operated in the conventional manner known in the art. These can constitute microprocessors or other digital data processing units housed independently of the field devices (e.g., valves) and/or they can constitute integral portions of those field devices (e.g., "smart field devices"). Alternatively, or in addition, the controllers can execute from within the same computer 18 that is utilized for system configuration.

Digital data processor 12, unit controllers 36A, 36B and field devices controllers CTL (and/or smart field devices) are coupled for communication via network 41. This communications medium permits downloading of control language codes that implement control strategies and other configuration information to controllers 36A, 36B and CTL, e.g., from digital data processor 12. It also provides a medium for uploading information from those controllers to digital data processor 12. Still further, it can provide a medium for communications, real-time or otherwise, between the controllers. Though illustrated to represent a LAN, WAN, or global network (Internet), those skilled in the art will appreciate that element 41 may represent any medium or mechanism through which control strategies and other information may be transported, electronically, physically or otherwise, to and from controllers 36A, 36B and CTL.

Executing on digital data processor 12 and, more particularly, on computer 18 are a configurator 38, preprocessor 40 and control language generator 42. Configurator 38 operates in the manner of similarly named functionality known in the art. Thus, configurator 38 models the illustrated process 21 and the desired algorithm for controlling it. This includes enumerating field devices, control devices, controllers and other apparatus used for control, specifying their interrelationships and the information transferred among them, as well as detailing the calculations and methodology they will apply for purposes of control. In embodiments of the invention adapted for process control, such modeling is provided through graphical or other manipulation of objects representing, by way of non-limiting example, field devices, control devices, control processors, blocks, loops, compounds, historians, object type categories, object connections, parameter connections, display placeholders, graphical display entities, and reports.

A preferred configurator 38 operates in the manner of the IDA Control Algorithm Configurator disclosed in copending, commonly assigned United States Patent Application Serial No. 09/448,223, for PROCESS CONTROL CONFIGURATION SYSTEM WITH CONNECTION VALIDATION AND CONFIGURATION, filed November 23, 1999 (Attorney Docket No. 102314-54), the teachings of which are incorporated herein by reference.

Control language generator 42 generates control codes (or control language) for downloading to controllers 36A, 36B and CTL. These codes are based on the aforementioned model in the general manner known in the art, as adapted in accord with the teachings hereof. The format of the codes vary depending on the type and makeup of controllers, all as known in the art. A preferred control language generator 42 operates in the manner of the Download Manager discussed in aforementioned, incorporated-by-reference United States Patent Application Serial No. 09/448,223. Another preferred control language generator operates in the manner of such functionality provided with the aforementioned commercially available I/A Series® systems.

Preprocessor 40 performs initial processing on the configurator model, prior to its being passed to the control language generator. In the illustrated embodiment, this initial processing entails resolving undeclared symbols or names assigned to parameters, variables or other constructs (collectively, "parameters") particularly, for example, within structured text or other source-code contained within, or otherwise associated with, the objects. Preprocessor 40 can perform additional functions, e.g., of the type common for such functionality in the art.

Figure 2 depicts objects in a model generated by a configurator 38 in a system according to the invention. These objects model aspects of the controlled system 21, including, for example, chambers 22A, inlets 24A, 26A, outlet 28A, paddle 30A, cooler 32A, cooler inlet 34A and corresponding equipment associated with unit 20A. They also model the control system, including, for example, unit controllers 36A, 36B and field device controllers CTL. The objects can also model aspects of the control environment.

While not showing a complete model generated by the configurator, Figure 2 depicts a control level hierarchy according to which at least some of the objects are organized. That hierarchy is the so-called SP88 industry standard taught, for example, in Batch Control Part 1: Models and Terminology, ISA-S88.01-1995, approved February 28, 1995 (the "SP88 specification"), the teachings of which are incorporated herein by reference. Illustrated in the drawing are levels of the SP88 hierarchy, including area 44, process cell 46, unit operation 48, equipment module 50, and control module 52. Not illustrated, though optionally included in embodiments of the invention are the enterprise and site layers of that hierarchy. Other hierarchies, industry-standard or otherwise, may be used as a means of organizing objects as an alternative to, or in addition to, the SP88 hierarchy.

The illustrated objects can be implemented with object-oriented programming (OOP) objects, as well as with other programming constructs such as, by non-limiting example, records, structs, arrays, and tables (collectively, "objects"). In the illustrated embodiment, the objects represent blocks, which are conventional "black box" units of functionality for the respective hierarchical levels.

By way of example, blocks 44A, 44B, in the area layer, can model physical, geographical, or logical groupings of process cells, units, equipment modules, and control modules, e.g., based on organizational or business criteria in accord with the SP88 specification. In further accord with that specification, these blocks may serve as "parents," "grand-parents" and so forth, to blocks representing process cells, units, equipment modules, and control modules, which descend therefrom.

Blocks 46A, 46B, in the process cell level, can model logical groupings of equipment required for production of one or more batches. In accord with the SP88 specification, a process cell block descends from, and is a "child" to, an area block. In turn, unit, equipment module, and control blocks required to make one or more batches can descend from the respective process cell block.

Illustrated blocks 48A, 48B model units, e.g., portions of the system configured to carry out specific tasks. Each unit block descends from a process cell block and, in turn, is the parent, grandparent, etc., of blocks representing equipment and control modules. In the illustrated embodiment, the unit blocks 48A, 48B can model unit controllers 36A, 36B, respectively. These and/or additional blocks (not illustrated) can model digital data processing apparatus (or other devices) in which those controllers are embodied, as well as other aspects of the control system, the controlled system, at the unit level.

Blocks 50A, 50B model the system at the equipment layer. According to the SP88 specification, such blocks are usually centered on a piece of processing equipment and are defined by the finite tasks such equipment is designed to carry out. Each equipment-level block is the child of a unit block or of another equipment-level block, and is the parent of a control module block or another equipment-level block.

In the illustrated embodiment, block 50A can represent cooling equipment used for temperature control of mixing chamber 22A in the unit controlled by controller 36A and modeled by unit block 48A. Likewise, block 50B can represent cooling equipment used for temperature control of mixing chamber 22B in the unit controlled by controller 36B and modeled by unit block 48B.

Blocks 52A, 52B model the system at the control module level. This typically includes sensors, actuators, other devices, and associated processing equipment that are operated as a single entity from a control perspective. Each control module block is the child of an equipment-level block or of another control module block. A control module block can be an end-node (e.g., leaf) in the hierarchy or the parent of a another control module block.

In the illustrated embodiment, block 52A can represent an actuator used in the cooling equipment modeled by block 50A and used for temperature control of mixing chamber 22A in the unit controlled by controller 36A and modeled by unit block 48A. Likewise, block 52B can represent an actuator used in the cooling equipment modeled by block 50B and used for

temperature control of mixing chamber 22 in the unit controlled by controller 36A and modeled by unit block 48B.

The illustrated blocks include names or other identifiers by which they are identified. They typically also include parameters that define characteristics of each object and, more particularly, of the element or entity the object represents. Depending on the type of object, these include inputs, outputs, alarm limits, control functions and display characteristics, among others. Each parameter itself has a name (e.g., ABC101) or other symbol by which it is referenced, and other attributes, e.g., data type, value, permissible ranges, formulaic definition, etc.

Blocks and objects, including parameters, utilized in the illustrated embodiment of the invention are preferably structured and utilized in the manner of those described in aforementioned incorporated-by-reference patent application Serial No. 09/448,223. Other block and/or object constructions known in the art can also be employed in systems according to the invention.

Whether characterized as a parameter, or otherwise, each block can additionally include an instruction sequence or source code (collectively, "code") defining further aspects of the modeled entity. That code can, more particularly, by way of non-limiting example, define the behavior of the modeled entity, e.g., in response to various inputs or other conditions. The code can be written in an English language-like structured text format or any format known in the art. That code is preferably amenable for compilation or interpretation as a control language for downloading and execution in the control system. Formats of structured text and other code used to define block operation are known in the art, e.g., in process control configuration systems of the type available from the assignee hereof under the I/A Series® mark.

An excerpt of structured text code of the type incorporated, e.g., in block 48A, is shown below. This block defines input parameters A52 and B9, along with output parameter ABC101. It, further, assigns the latter a value equal to the sum of the former two.

```
VAR_INPUT
        A52     REAL_OUT;
        B9      REAL_OUT;
END-VAR

VAR_OUTPUT
        ABC101      REAL_OUT;
END_VAR

        . . .

ABC101 := A52 + B9;

        . . .
```

Though each of the blocks shown in Figure 2 are shown as including source code (denominated "CODE"), those skilled in the art will appreciate that some (or all) objects may include no code, while other objects may include more than one code section.

Figure 6 is an alternate view showing the effect of extended parameter scope on a model with one area block (AREA 1), one processing cell block (PROC CELL 1), two unit blocks (UNIT OP 1 and UNIT OP 2), and their attendant equipment and control module blocks, all as illustrated. Here, extended parameter scope is defined in multiple branches as indicated by the shaded rectangles around branches stemming from the hierarchy at blocks UNIT OP 1, EQUIP 3, and EQUIP 4, respectively.

In the prior art, the naming scope of user-defined parameters is limited to blocks in which those parameters are defined. Thus, in the example above, parameters A52, B9 and ABC101 would be known only to the area block 48A in which they are declared. To overcome this, the prior art requires explicit declaration of parameters accessed by multiple blocks, e.g., by way of explicit input parameter and output parameter declarations and/or by

appending to the names of "shared" parameters those of blocks from which those parameters are sourced.

The illustrated embodiment adds further flexibility in the sharing of parameters by extending their scope to selected branches of the object hierarchy. Returning to Figure 2, the branch at the unit operation layer is designated for such extension. For this purpose, a parameter labeled "Extend Scope" is activated (via a dialog box, user request, or otherwise) in the unit blocks, indicating that all blocks hierarchically descending from each of them share a respective common scope for parameter names and, hence, do not require explicit parameter declaration (whether by source code declarations or name-appending) to effect parameter sharing.

As a result of such parameter scope extension, hierarchically descending blocks 48A, 50A, and 52A -- modeling controller 36A, cooling equipment for temperature control of mixing chamber 22A, and an actuator in that cooling equipment, respectively -- can share parameters without redeclaration and/or appending source block names. Blocks 48B, 50B and 52B can likewise share parameters among themselves. In each of the cases, the parameters shared within a branch are independent of those shared within another branch, absent, redeclaration and/or the appending source block names in the manner of the prior art.

The effect of parameter scope extension is shown in Figure 3. There, a parameter ABC101 is declared as a real (or floating point) value in unit block 48A, with its value set as the sum of two other parameters (A52 and B9). Descendant control module block 52A, which resides in the same branch of the hierarchy, refers to parameter ABC101 without redeclaring it. This permits, for example, a calculation of additional parameters in block 52A that are dependent on the value of ABC101 set in block 48A, e.g., without the need for explicit declarations of the source of that parameter.

The extension of parameter scope in accord with the invention is not limited to branches that begin at the unit level of hierarchy. Thus, embodiments of the invention permit

common scope to be set via activation of an Extend Scope parameter (or the like) beginning at other levels of the hierarchy.

In operation, a user (e.g., operator or process control engineer) configures the control system 21 (Figure 1) to create a model using a hierarchical model of the type shown in Figure 2. Upon creating a unit operations block (e.g., 48A), the user is presented with a dialog box (or the like) permitting selection of an option to extend parameter scope. If selected, block 48A and all blocks descending therefrom have a common scope for all user-defined parameters. Accordingly, all such parameters within that branch must be unique within that scope. It will be appreciated that any variety of user interface techniques known in the art may be utilized to permit the selection of parameter scope extension from individual blocks (e.g., block 48A) or from all blocks of like type (e.g., all unit operation blocks).

Referring back, Figure 2 illustrates the storage of information to support parameter extension in an embodiment of the invention. Each individual block, e.g., 44A - 52B, in the model has a dedicated store (an exemplary one of which is labeled Parameter File in the drawing) for parameters defined within that block. In a branch where parameter extension is activated, a store (here, labeled Scope Parameters) is created for the branch as a whole. This is loaded with a complete list of all parameters within the branch. The storage format of the Scope Parameters can be identical to that of the individual parameter file stores that contribute parameters.

Figure 4 depicts a method of operation of preprocessor 40 (Figure 1) for interpreting and extending the scope of user-defined parameters contained in blocks 44A - 52B in a system according to the invention.

In step 60, the preprocessor reads the source code contained in, or otherwise associated with, a first block, e.g., unit block 48A. The preprocessor identifies input and output parameters declared in that code.

In step 62, those input and output parameters are put into the Parameter File associated with block 48A. The preprocessor also parses the source code to identify any undeclared parameter references, of which there are none in this example.

In step 64, parameters from the Parameter File for block 48A are added to the Scope Parameter store associated with the branch, for which parameter extension is selected. That store can be maintained, for example, as a parameter of the root block of the branch, in this case, block 48a, or otherwise. The store need not need be downloaded into the processor controller, since its principal purpose is for use in the configuration environment.

In step 66, the preprocessor reads the source code contained in, or otherwise associated with, a second block of the same branch, e.g., control module block 52A. The preprocessor identifies input and output parameters declared in that code.

In step 68, those input and output parameters are put into the Parameter File associated with block 52A. The preprocessor identifies any undeclared parameter references. ABC101 is identified in this example.

In step 70, the preprocessor reads the Scope Parameters store for the branch to locate a parameter (here, ABC101 from block 48A) matching the undeclared reference. When found, the preprocessor generates any necessary declarations to link the otherwise undeclared parameter in the second block to the like-named declared parameter in the first block. The Parameter File for block 48A is extended to include any automatically generated parameters. Those parameters are also added to the Extended Scope store.

In step 72, the preprocessor generates a list of block-to-block connections based on the parameters in the source file. This represents autolinkage of a parameter in the extended scope. A further appreciation of such block-to-block connections (or "interblock" connections) may be attained by reference to United States Patent Application Serial No. 09/448,223, entitled PROCESS CONTROL CONFIGURATION SYSTEM WITH CONNECTION VALIDATION AND CONFIGURATION (Attorney Docket: 102314-54).

Figure 5 depicts a preprocessor 40 and its interaction with other system components in a further embodiment of the invention. As shown in the drawing, the preprocessor includes a preprocessor control 80, a directive expander 84, an error resolver 86, and a parameter expander 88. The preprocessor control 80 is responsible for controlling conversion of expanded source code received from a structured text source code editor 82 into executable form for downloading to the runtime control system. The process consists of: preprocessing the source code to expand any preprocessor directives; syntax analysis and code generation; error collection and resolution; updating of the model generated by the configurator 38; reporting of results and/or errors to the structured text source code editor 82.

The preprocessor control 80 receives a syntax check or syntax check and translate command from the structured text source code editor 82, along with the location of the expanded source code block (here, labeled "Block") to process. The preprocessor control 80 then invokes the directive expander 84 to expand any preprocessor directives. If the directive expander 84 returns errors, the preprocessor control 80 calls the error resolver 86 to associate all errors with the correct line in the original source code. The error resolver 86 then creates and returns an error message to the structured text source code editor 82.

If the directive expander 84 finds no errors, the preprocessor control 80 calls a control language build utility 90. This performs syntax checking and translation, compilation, and links (if required) the source code into an executable for downloading. If there are errors in the translation, compilation, or linking, the control language build utility 90 returns an error message to the preprocessor control 80, which in turn calls the error resolver 86 to generate the appropriate error message for the structured text source code editor 82.

If no errors are reported by the control language build utility 90, the preprocessor control 80 checks to see if any parameters needed by other blocks have been deleted since the last compile. The preprocessor control 80 does not allow the model generated by the configurator 38 to be updated if any such parameters have been detected and an error is returned to the structured text source code editor 82.

If there were no errors reported by the control language build utility 90 and no parameters needed by other blocks were deleted, the preprocessor control 80 updates the source code block and return a success status to the structured text source code editor 82.

In addition to controlling conversion of source code to executable code, the preprocessor control 80, upon request of the structured text source code editor 82, retrieves a list of all available parameters in the extended parameter scope of the current block. This information is returned to the structured text source code editor 82.

The directive expander 84 is invoked from the preprocessor control 80. This expands all #includes that refer to text files included in the model generated by configurator 38, as well as other preprocessor directives. The preprocessor is responsible for inserting a version string into the control language block. This version string is used by the control language build utility 90 to mark the dynamic source code file for version control.

#Include directives are expanded only if the referenced object is found in the model generated by configurator 38. Otherwise an error is signaled. Syntax errors in expanded preprocessor directives are reported as occurring at the source code line containing the preprocessor directive. Tracking of the expanded #include line numbers is the responsibility of the directive expander 84, all other line numbers are tracked by the control language build utility 90.

The parameter expander 88 resolves all parameters referenced in the source code. There are three types of parameters: input, input-output, output, and external. The control language build utility 90 invokes the parameter expander 88 the first time it encounters any parameter while processing the source code. For all parameter types, the attribute information is passed from the control language build utility 90 to the parameter expander 88 and is stored in the control language block, portions of which will be downloaded to the control system. In addition the parameter expander 88 is responsible for receiving from the control language build utility 90 and storing information about the configuration parameters of legacy blocks.

An input parameter is sent to the parameter expander 88 when its declaration is found in the source code. The input parameter may be declared and initialized (linked to a parameter of another block) in the source code, declared in the source code and linked by an external program (e.g., a configuration editor), or declared only (no link). If the input parameter is declared and linked in the source code, the parameter that it is linked to must be checked by the parameter expander 88 to ensure that it is a valid parameter for another block. The complete path in the block hierarchy to the parameter is passed from the control language build utility 90 to the parameter expander 88. If the input parameter is declared as safety parameter it may only be connected to a safety parameter. In all cases a user parameter in the control language block is created for each input parameter. Likewise, an entry into the Extended Scope store (see Figure 4) is created if extended parameter scope exists and the input parameter name is unique within the extended parameter scope and is not Limited. If the parameter name is not unique the parameter expander 88 issues an error to the control language build utility 90. The preprocessor control 80 makes all entries into the control language block after successful compilation and linking.

In the subject embodiment, the Extended Scope store is implemented as an object-oriented programming object and referred to as an Extended Parameter Scope Object. That object is not be impacted by a name change for a block contained within the object; is able to handle the deletion of a block; is able to handle the relocation of a block; returns a list of parameters available in the current block's extended parameter scope upon request; is able to handle the renaming of a compound within the compound hierarchy structure; and returns a list of parameters available in the current block's extended parameter scope upon request.

The parameter expander 88 is called for each output parameter when its declaration is found in the source code. The parameter expander 88 must check the extended parameter scope to ensure that the output parameter has a unique name. If the name is unique, entries are made for the output parameter in the control language block and the Extended Parameter Scope Object. If the output parameter is identified as limited an entry is not be made in the Extended Parameter Scope Object nor is any examination of the object be made.

The parameter expander 88 is called for each input-output parameter when its declaration is found in the source code. The input-output parameter may be declared and initialized (linked to a parameter of another block) in the source code, declared in the source code and linked by an external program (e.g., a configuration editor), or declared only (no link). If the input-output parameter is declared and linked in the source code, the parameter that it is linked to must be checked by the parameter expander 88 to ensure that it is a valid parameter for another block. If the input-output parameter is declared and linked by an external program, the parameter expander 88 must verify that the link is to a valid parameter and returns the complete path in the block hierarchy to that parameter. If the input parameter is declared as safety parameter in may only be connected to a safety parameter. In all cases a user parameter in the control language block is created for each input-output parameter. Likewise, an entry into the Extended Parameter Scope Object is created if extended parameter scope exists and the input-output parameter name is unique within the extended parameter scope and is not limited. The preprocessor control 80 enters the control language block and Extended Parameter Scope Object entries into the control language block and Extended Parameter Scope Object after successful compilation and linking.

The parameter expander 88 is called for each external parameter in the source code the first time the parameter is referenced in the source code. The external parameter may contain a scope parameter, a full path name, or a relative path name.

If the parameter is a scope parameter, the parameter expander 88 must check the extended parameter scope to determine if the parameter exists. Error messages are returned if the parameter is not found. If the parameter is found within extended parameter scope, the parameter is copied from the control language block where the parameter is defined, to the current control language block. An inter-block connection is then made between the parameter and the copy of the parameter.

If the parameter is a full path name or a relative path name, the parameter expander 88 determines if the parameter exists. If it does not exist and a data type is provided, the parameter expander 88 creates a standard parameter on the control language block and return

a warning message. If the parameter does exist, the parameter expander 88 creates a standard parameter on the control language block and create an inter-block connection between the parameter that was found and the standard parameter.

The Extended Parameter Scope Object is accessed by the parameter expander 88 to resolve parameters and to add parameters to the Extended Parameter Scope. If a block or compound changes name, is moved within the hierarchy, or is deleted from the hierarchy the Extended Parameter Scope Object makes the required corrections to the data base to reflect these changes in the hierarchy.

The error resolver 86 is invoked from the preprocessor control 80. The error resolver 86 relates all errors found in the preprocessor expanded control language block to their line in the original control language block that was sent by the structured text source code editor 82. The error resolver 86 ensures that all errors found in text that was inserted by the directive expander 84 are related to the preprocessor directive in the original source code. The error resolver 86 returns a status to the preprocessor control 80 and update the compilation status parameter in the source code block when it is finished.

Referring to Figure 6, extended parameter scope may be created or deleted for a block through a graphical interface and corresponding OOP objects (or other programming constructs), hereinafter referred to as the "tree," that allow the user to select a box that enables or disables extended parameter scope.

The tree creates or deletes the extended parameter scope objects. If extended parameter scope for the requested block already exists, it cannot be created again. Likewise deletion of a non-existent extended parameter scope is ignored. An extended parameter scope is not be deleted if any of the parameters in the extended parameter scope are being referenced in a compiled block that is part of the extended parameter scope.

The example shown in Figure 6 demonstrates extended parameter scopes that have been defined in the UNIT OP 1, EQUIP 3, and EQUIP 4 components. The extended

parameter scope parameters are visible to all of the components and blocks within each of the extended parameter scope areas defined as Ext Scope 1, Ext Scope 2, Ext Scope 3.

If, during a syntax check and translation of a source code in a block at the Control Module level in the CTRL MOD 1 component, an undefined local parameter is discovered, a search for an extended parameter scope is conducted. First the Control Module level is searched followed by the Equipment and Unit Operations levels in turn. There is an extended parameter scope defined in the Unit Operations level in the UNIT OP 1 component. Because there can only be one extended parameter scope per branch in the illustrated embodiment the search stops here. The parameter expander 88 searches the list of parameters in the UNIT OP 1 Extended Parameter Scope Object (Ext Scope 1) for the unresolved parameter. If it is found, the parameter expander 88 passes the parameter location to the control language build utility 90. If it is not found, the parameter expander 88 passes an error indicator to the control language build utility 90.

The Extended Parameter Scope Object encapsulates the database storage of scope information. It contains a parameter that points to the compound with which it is associated. It also contains copies of all output parameters defined by blocks within the scope of the object. Note that it only contains references to control language blocks that have been successfully compiled, and is not maintained as the user modifies code within the source code editor 82.

The Extended Parameter Scope Object's primary functions are to provide the source code editor 82 with information about the parameters in the extended parameter scope and to allow the control language build utility 90 to resolve parameters it encounters during the translation process. In addition, it provides interfaces to allow the creation and deletion of itself from the tree.

The source code editor 82 accesses the Extended Parameter Scope Object via the Get Parameter interface. The source code editor 82 may request a list of all extended parameter scope parameters from the Extended Parameter Scope Object.

During the translation of the Structured Text the control language build utility 90 calls the parameter expander 88 the first time it encounters an input, input-output, output, or external parameter in the Structured Text to resolve the parameter. The control language build utility 90 sends the parameter name along with a flag indicating if the parameter is an input, an input-output, an output, or an external parameter. The parameter expander 88 accesses the Extended Parameter Scope Object as required to resolve the parameter in question.

The Extended Parameter Scope Object supports the creation and deletion of extended parameter scope. The aforementioned tree initiates the creation and deletion of the extended parameter scope at the desired hierarchy level by creating or destroying the Extended Parameter Scope Object.

Described above are systems and methods meeting the desired objects. Those skilled in the art will of course appreciate that the embodiments described herein are merely examples of the invention and that other embodiments incorporating modifications thereto fall within the scope of the claims. By way of non-limiting example it will be appreciated that the invention may be utilized in connection with hierarchical models of any type, regardless of whether defined by industry standard (such as SP88), proprietary or otherwise. It will further be appreciated that the scope of parameters can be extended regardless of whether they are user-defined and regardless of whether they are utilized in source code. In view thereof, what I claim is:

1. In an apparatus for configuring a control system that is modeled by a hierarchy of objects, the improvement wherein a plurality of objects in a selected branch of the hierarchy have a common scope with respect to naming of at least selected parameters, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

2. In the apparatus according to claim 1, the further improvement wherein the selected parameters are user-defined parameters.

3. In the apparatus according to claim 1, the further improvement wherein the objects represent entities within the any of (i) the control system, (ii) a system controlled by the control system, (iii) a control level hierarchy, and (iv) the apparatus for configuring the control system.

4. In the apparatus according to claim 3, the further improvement wherein the parameters relate to characteristics associated with entities represented by the respective objects.

5. In an apparatus for configuring a control system that is modeled by a hierarchy of objects, one or more of which include names, the improvement wherein a plurality of objects in a selected branch of the hierarchy have a common scope with respect to resolution of undeclared names, that scope being independent of a scope of resolution of undeclared names used by objects in any other branch of the hierarchy.

6. In the apparatus according to claim 5, the further improvement wherein the undeclared names are user-defined.

7. In the apparatus according to claim 5, the improvement wherein the undeclared names pertain to control parameters.

8. In an apparatus for configuring a control system that is modeled by objects that represent entities in a control hierarchy, one or more objects including source code with

symbols that reference one or more parameters, the improvement wherein symbols for at least selected parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy are resolved for downloading into the control system in accord with a common scope that is independent of a scope within which symbols for parameters in any other branch of the hierarchy are resolved.

9. In the apparatus according to claim 8, the further improvement wherein the selected undeclared parameters are user-defined parameters.

10. In an apparatus for configuring a process control system wherein the control system is modeled by a hierarchy of objects that have parameters, the improvement wherein at least a selected plurality of objects in a selected branch of the hierarchy have a common scope with respect to naming of at least selected parameters downloaded into the process control system, that scope being independent of the scope within which parameters are named by objects in any other branch of the hierarchy.

11. In the apparatus according to claim 10, the further improvement wherein the selected parameters are user-defined parameters.

12. In the apparatus according to claim 10, the further improvement wherein the objects represent entities in a control hierarchy that includes two or more levels defined in an SP88 industry standard hierarchy.

13. In the apparatus according to claim 12, the further improvement wherein the parameters relate to characteristics associated with entities represented by the respective objects.

14. In an apparatus for configuring a process control system that is modeled by objects that represent entities in a control hierarchy, one or more objects including source code with symbols that reference one or more parameters, the improvement wherein symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected

branch of the hierarchy are resolved for downloading into the control system in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved.

15. In the apparatus according to claim 14, the further improvement wherein the selected undeclared parameters are user-defined parameters.

16. In the apparatus according to claim 14, the further improvement wherein the control hierarchy two or more of an enterprise level, a site level, an area level, a process cell level, a unit operation level, an equipment level and a control module level.

17. In an apparatus for configuring a process control system that is modeled by objects that represent entities in a control hierarchy, one or more objects having source code defining a behavior of an entity modeled by the object, the improvement wherein

symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy are resolved for downloading into the control system in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved,

a first object in the selected branch declares a first symbol as referencing a first parameter,

the first symbol being interpreted as referencing the first parameter in a second object that is in the selected branch and that uses the first symbol without declaration thereof to the contrary.

18. In the apparatus according to claim 17, the further improvement the first symbol is not interpreted as referencing the first parameter in a third object that is not in the selected branch, unless the first symbol is explicitly declared otherwise.

19. In the apparatus according to claim 17, the further improvement wherein the selected symbols, including the first symbol, identify process control parameters.

20. In the apparatus according to claim 17, the further improvement wherein the selected symbols, including the first parameter, identify user-defined process control parameters.

21. In the apparatus according to claim 19, the further improvement wherein all objects in the selected branch have common scope with respect to the selected parameters.

22. Apparatus for configuring a control system

a configurator that models the control system with a hierarchy of objects,

a control language generator that generates codes based on the model for downloading to the control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor providing a common scope for naming at least selected parameters that are used by a plurality of objects within a selected branch of the hierarchy and that are to be downloaded to the control system, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

23. Apparatus according to claim 22, wherein the selected parameters are user-defined parameters.

24. Apparatus according to claim 22, wherein the objects represent entities within the any of (i) the control system, (ii) a system controlled by the control system, (iii) a control level hierarchy, and (iv) the apparatus for configuring the control system.

25. Apparatus according to claim 24, wherein the parameters relate to characteristics associated with entities represented by the respective objects.

26. Apparatus for configuring a control system

a configurator that models the control system with a hierarchy of objects, one or more of which include names,

a control language generator that generates codes based on the model for any of configuring and controlling the control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

27. Apparatus according to claim 26, wherein the names are user-defined.

28. Apparatus according to claim 26, wherein the names pertain to control parameters.

29. Apparatus for configuring a control system

a configurator that models the control system with a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

a control language generator that generates codes based on the model for any of configuring and controlling the control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor resolving for downloading into the control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is independent of a

scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved.

30. Apparatus according to claim 29, wherein the selected undeclared parameters are user-defined parameters.

31. Apparatus for configuring a process control system

a configurator that models the process control system with a hierarchy of objects,

a control language generator that generates codes based on the model for any of configuring and controlling the process control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor providing a common scope for naming of at least selected parameters by a plurality of objects within a selected branch of the hierarchy, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

32. Apparatus according to claim 31, wherein the selected parameters are user-defined parameters.

33. Apparatus according to claim 31, wherein the objects represent entities in a control hierarchy that includes two or more of an enterprise level, a site level, an area level, a process cell level, a unit operation level, an equipment level and a control module level.

34. Apparatus according to claim 33, wherein the parameters relate to characteristics associated with entities represented by the respective objects.

35. Apparatus for configuring a process control system

a configurator that models the process control system with a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

a control language generator that generates codes based on the model for any of configuring and controlling the process control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

36. Apparatus according to claim 35, wherein the selected parameters are user-defined parameters.

37. Apparatus according to claim 35, wherein the control hierarchy two or more levels selected from the levels of an SP88 industry standard hierarchy.

38. Apparatus for configuring a process control system

a configurator that models the process control system with a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

a control language generator that generates codes based on the model for any of configuring and controlling the process control system,

a preprocessor that is coupled to the configurator and to the control language generator, the preprocessor resolving for downloading into the process control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is

independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved,

the preprocessor responding to a declaration in a first object that associates a first symbol with a first parameter, by referencing the first parameter in response to the first symbol in a second object in the selected branch, absent an explicit declaration to the contrary.

39. Apparatus according to claim 38, wherein the preprocessor does not associate the first symbol with the first parameter in a third object that is not in the selected branch absent an explicit definition in the third object establishing such an association.

40. Apparatus according to claim 38, wherein the selected symbols, including the first symbol, identify process control parameters.

41. Apparatus according to claim 38, wherein the selected parameters, including the first parameter are user-defined process control parameters.

42. Apparatus according to claim 40, wherein all objects in the selected branch have common scope with respect to the selected parameters.

43. In a method for configuring a control system that is modeled by a hierarchy of objects, the improvement comprising providing a common scope for naming at least selected parameters that are used by a plurality of objects within a selected branch of the hierarchy and that are to be downloaded to the control system, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

44. In the method of claim 43, the further improvement wherein the selected parameters are user-defined parameters.

45. In the method of claim 43, the further improvement wherein the objects represent entities
within the any of (i) the control system, (ii) a system controlled by the control system, (iii) a control level hierarchy, and (iv) the apparatus for configuring the control system.

46. In the method of claim 45, the further improvement wherein the parameters relate to characteristics associated with entities represented by the respective objects.

47. In a method for configuring a control system that is modeled by a hierarchy of objects, one or more of which include names, the improvement comprising providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

48. In the method of claim 47, the further improvement wherein the names are user-defined.

49. In the method of claim 47, the further improvement wherein the names pertain to control parameters.

50. In a method for configuring a control system that is modeled by a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters, the improvement comprising resolving for downloading into the control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved.

51. In the method of claim 50, the further improvement wherein the selected undeclared parameters are user-defined parameters.

52. In a method for configuring a process control system that is modeled with a hierarchy of objects, the improvement comprising providing a common scope for naming of at least selected parameters by a plurality of objects within a selected branch of the hierarchy, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

53. In the method of claim 52, the further improvement wherein the selected parameters are user-defined parameters.

54. In the method of claim 52, the further improvement wherein the objects represent entities in a control hierarchy that includes two or more of an area level, a process cell level, a unit operation level, an equipment level and a control module level.

55. In the method of claim 54, the further improvement wherein the parameters relate to characteristics associated with entities represented by the respective objects.

56. In a method for configuring a process control system that is modeled by a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters, the improvement comprising providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

57. In the method of claim 56, the further improvement wherein the selected parameters are user-defined parameters.

58. In the method of claim 56, the further improvement wherein the control hierarchy two or more of an enterprise level, a site level, an area level, a process cell level, a unit operation level, an equipment level and a control module level.

59. In a method for configuring a process control system that is modeled by a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters, the improvement comprising the steps of

resolving for downloading into the process control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved,

the resolving step including responding to a declaration in a first object that associates a first symbol with a first parameter, by referencing the first parameter in response to the first symbol in a second object in the selected branch, absent an explicit declaration to the contrary.

60. In the method of claim 59, the further improvement wherein the preprocessor does not associate the first symbol with the first parameter in a third object that is not in the selected branch absent an explicit definition in the third object establishing such an association.

61. In the method of claim 59, the further improvement wherein the selected symbols, including the first symbol, identify process control parameters.

62. In the method of claim 59, the further improvement wherein the selected parameters, including the first parameter, identify user-defined process control parameters.

63. In the method of claim 61, the further improvement wherein all objects in the selected branch have common scope with respect to the selected parameters.

64. A method for configuring a control system, comprising the steps of

generating a model with a hierarchy of objects,

generating codes based on the model for downloading to the control system, and

providing a common scope for naming at least selected parameters that are used by a plurality of objects within a selected branch of the hierarchy and that are to be downloaded to the control system, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

65. The method of claim 64, wherein the selected parameters are user-defined parameters.

66. The method of claim 64, wherein the model-generating step includes generating a model with objects that represent entities within the any of (i) the control system, (ii) a system controlled by the control system, (iii) a control level hierarchy, and (iv) the apparatus for configuring the control system.

67. The method of claim 66, wherein the parameters relate to characteristics associated with entities represented by the respective objects.

68. A method for configuring a control system,

generating a model with a hierarchy of objects, one or more of which include names,

generating codes based on the model for downloading to the control system, and

providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

69. The method of claim 68, wherein the names are user-defined.

70. The method of claim 68, wherein the names pertain to control parameters.

71. A method for configuring a control system,

generating a model with a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

generating codes based on the model for downloading to the control system, and

resolving for downloading into the control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved.

72. The method of claim 71, wherein the selected undeclared parameters are user-defined parameters.

73. A method for configuring a process control system comprising the steps of

generating a model with a hierarchy of objects,

generating codes based on the model for downloading to the control system, and

providing a common scope for naming of at least selected parameters by a plurality of objects within a selected branch of the hierarchy, that scope being independent of a scope within which parameters are named by objects in any other branch of the hierarchy.

74. The method of claim 73, wherein the selected parameters are user-defined parameters.

75. The method of claim 73, wherein the model-generating step includes generating a model with objects that represent entities in a control hierarchy that includes two or more of an area level, a process cell level, a unit operation level, an equipment level and a control module level.

76. The method of claim 75, wherein the parameters relate to characteristics associated with entities represented by the respective objects.

77. A method for configuring a process control system, the method comprising

generating a model with hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

generating codes based on the model for downloading to the process control system, and

providing a common scope with respect to the naming of undeclared names used in any of a plurality of objects in a selected branch of the hierarchy, that scope being independent of a scope of resolution of undeclared names in any other branch of the hierarchy.

78. The method of claim 77, wherein the selected parameters are user-defined parameters.

79. The method of claim 77, wherein the control hierarchy two or more of an area level, a process cell level, a unit operation level, an equipment level and a control module level.

80. A method for configuring a process control system, comprising

generating a model with a hierarchy of objects, one or more objects including source code with symbols that reference one or more parameters,

generating codes based on the model for downloading to the control system, and

resolving for downloading into the process control system symbols for at least selected undeclared parameters in the source code of any of a plurality of objects in a selected branch of the hierarchy in accord with a common scope that is independent of a scope within which symbols for undeclared parameters in any other branch of the hierarchy are resolved,

the resolving step including responding to a declaration in a first object that associates a first symbol with a first parameter, by referencing the first parameter in response to the first symbol in a second object in the selected branch, absent an explicit declaration to the contrary.

81. The method of claim 80, wherein the preprocessor does not associate the first symbol with the first parameter in a third object that is not in the selected branch absent an explicit definition in the third object establishing such an association.

82. The method of claim 80, wherein the selected symbols, including the first symbol, identify process control parameters.

83. The method of claim 80, wherein the selected parameters, including the first parameter, identify user-defined process control parameters.

84. The method of claim 82, wherein all objects in the selected branch have common scope with respect to the selected parameters.

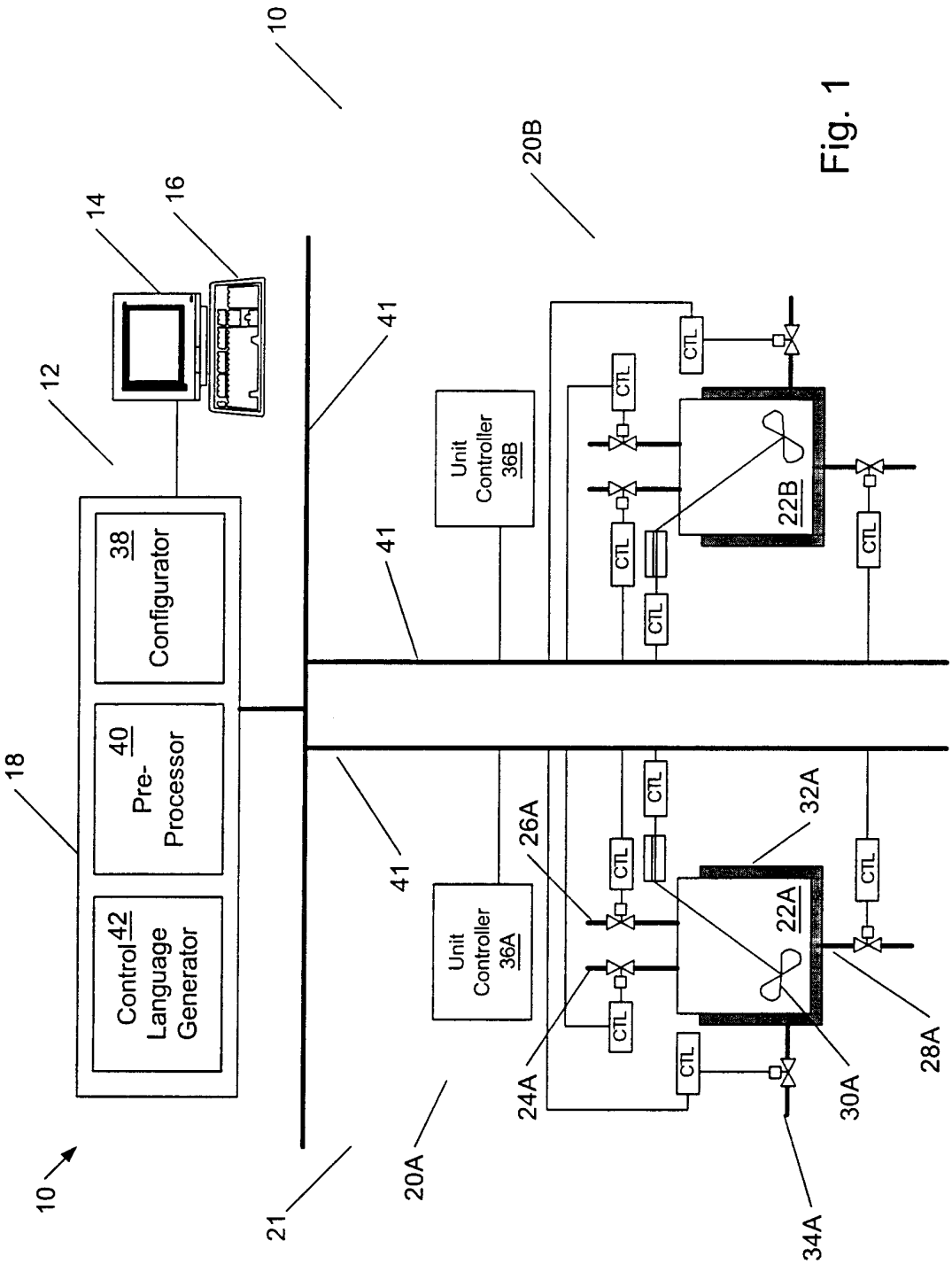

10
14
16
12
18
38 Configurator
40 Pre-Processor
42 Control Language Generator
10
21
20A
20B
41
41
41
Unit Controller 36A
Unit Controller 36B
CTL
22A
22B
24A
26A
28A
30A
32A
34A

Fig. 1

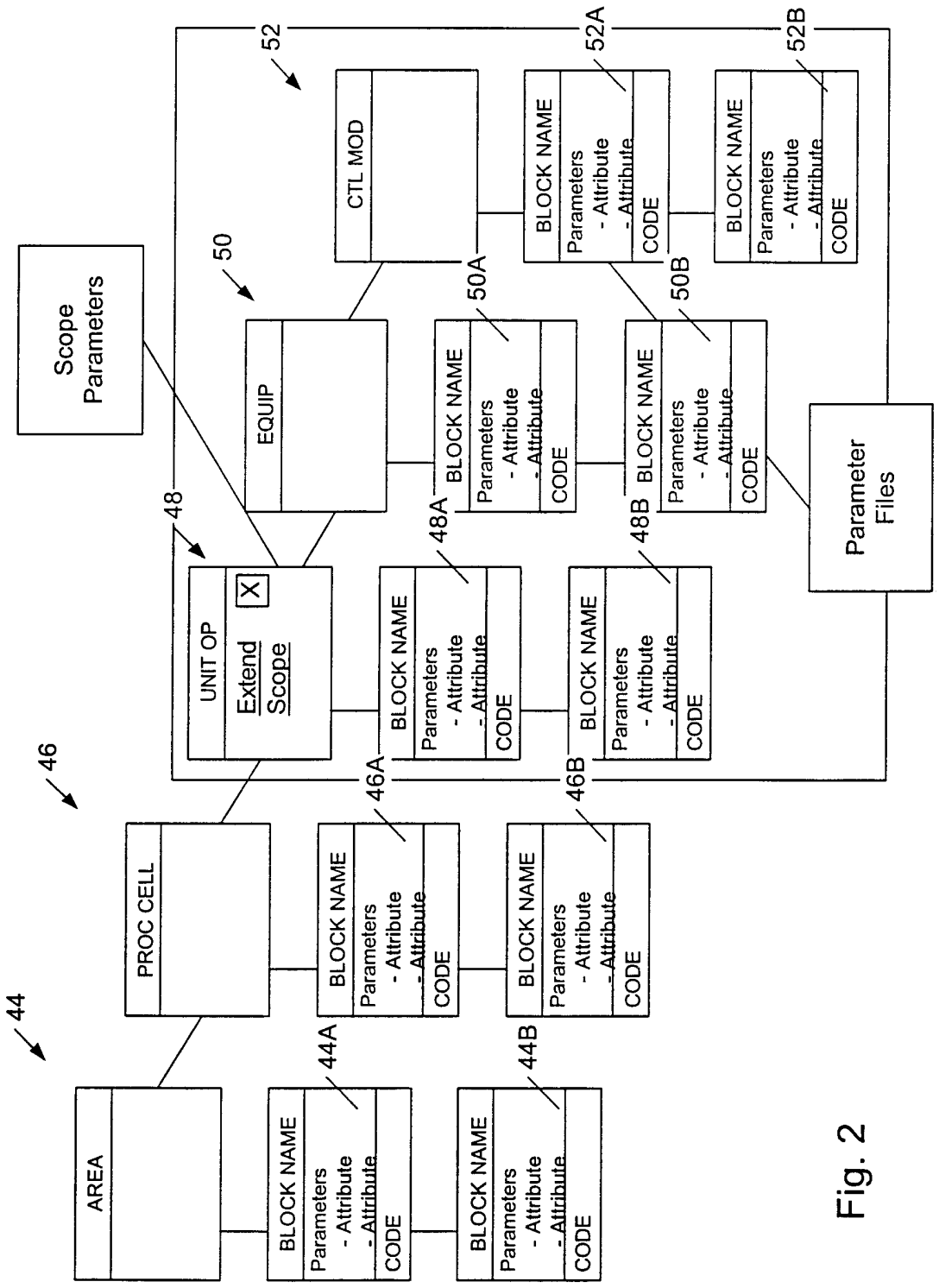

Scope Parameters
52
52A
52B
CTL MOD
BLOCK NAME
Parameters
- Attribute
- Attribute
CODE
50
50A
50B
EQUIP
48
48A
48B
UNIT OP
X
Extend Scope
46
46A
46B
PROC CELL
44
44A
44B
AREA
Parameter Files

Fig. 2

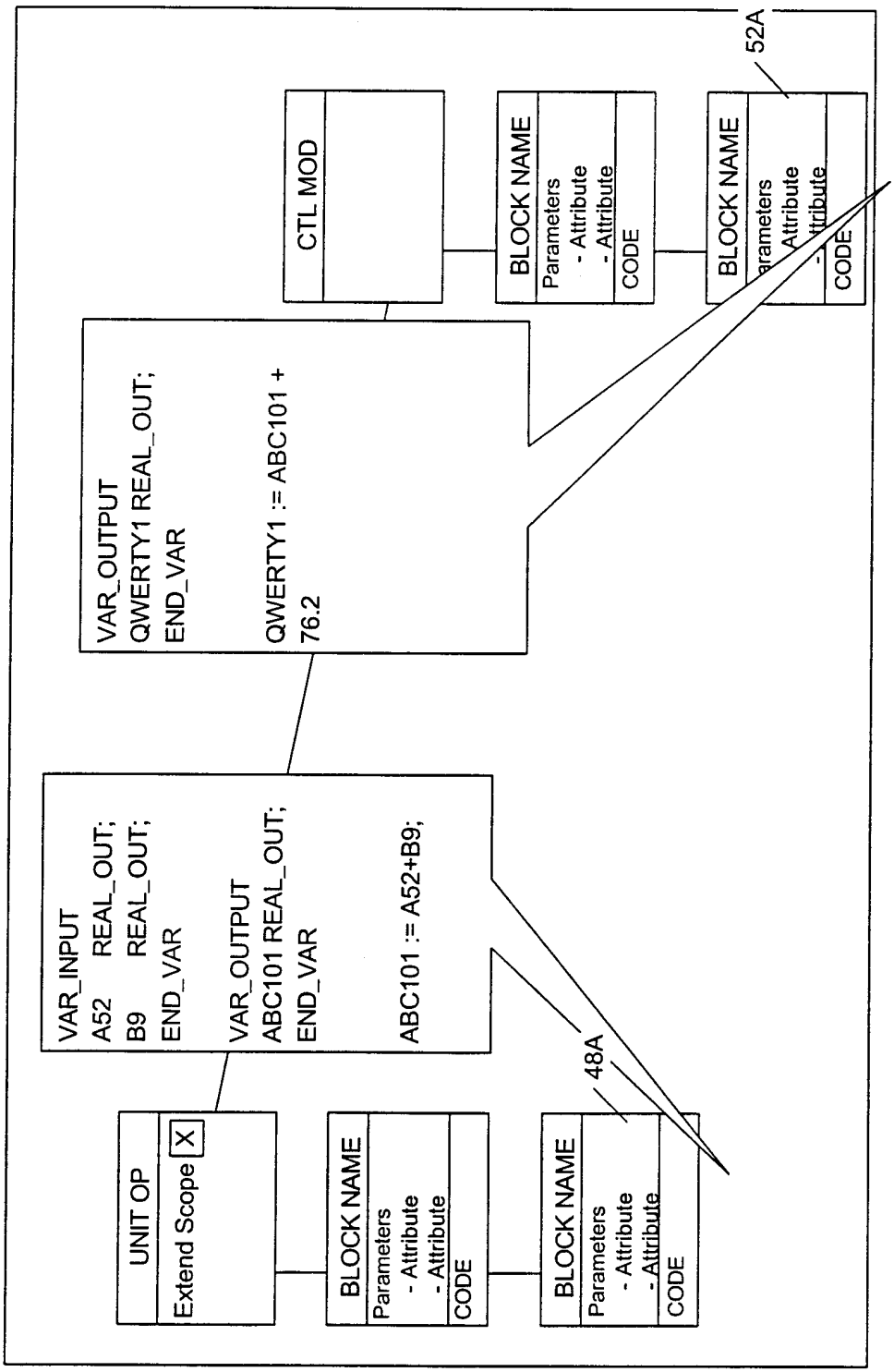

UNIT OP
Extend Scope X
BLOCK NAME
Parameters
- Attribute
- Attribute
CODE
BLOCK NAME
Parameters
- Attribute
- Attribute
CODE
48A
VAR_INPUT
A52 REAL_OUT;
B9 REAL_OUT;
END_VAR
VAR_OUTPUT
ABC101 REAL_OUT;
END_VAR
ABC101 := A52+B9;
VAR_OUTPUT
QWERTY1 REAL_OUT;
END_VAR
QWERTY1 := ABC101 +
76.2
CTL MOD
BLOCK NAME
Parameters
- Attribute
- Attribute
CODE
BLOCK NAME
arameters
Attribute
ttribute
CODE
52A

Fig. 3

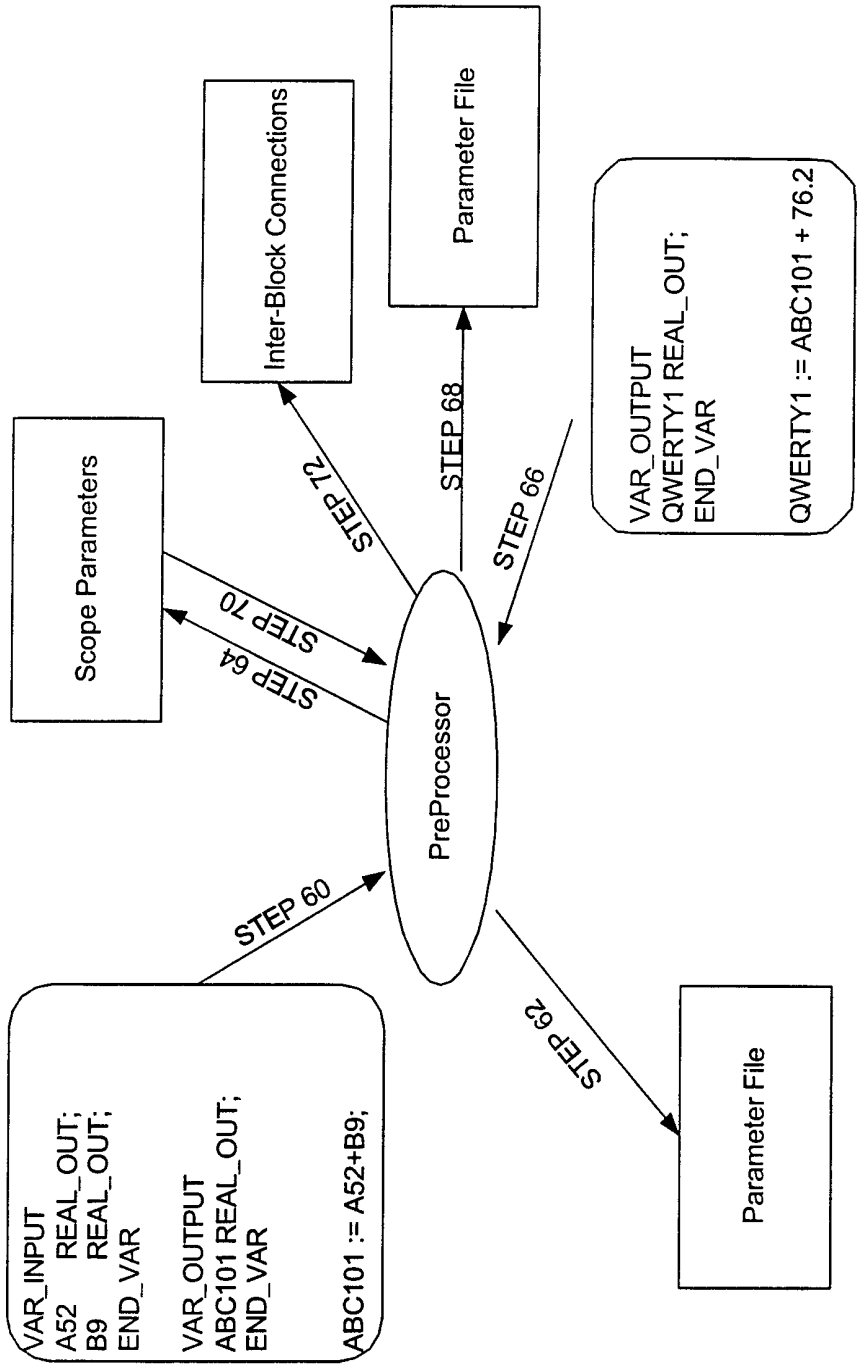

Inter-Block Connections
Parameter File
Scope Parameters
STEP 72
STEP 68
STEP 66
STEP 70
STEP 64
PreProcessor
VAR_OUTPUT
QWERTY1 REAL_OUT;
END_VAR
QWERTY1 := ABC101 + 76.2
STEP 60
STEP 62
VAR_INPUT
A52 REAL_OUT;
B9 REAL_OUT;
END_VAR
VAR_OUTPUT
ABC101 REAL_OUT;
END_VAR
ABC101 := A52+B9;
Parameter File

Fig. 4

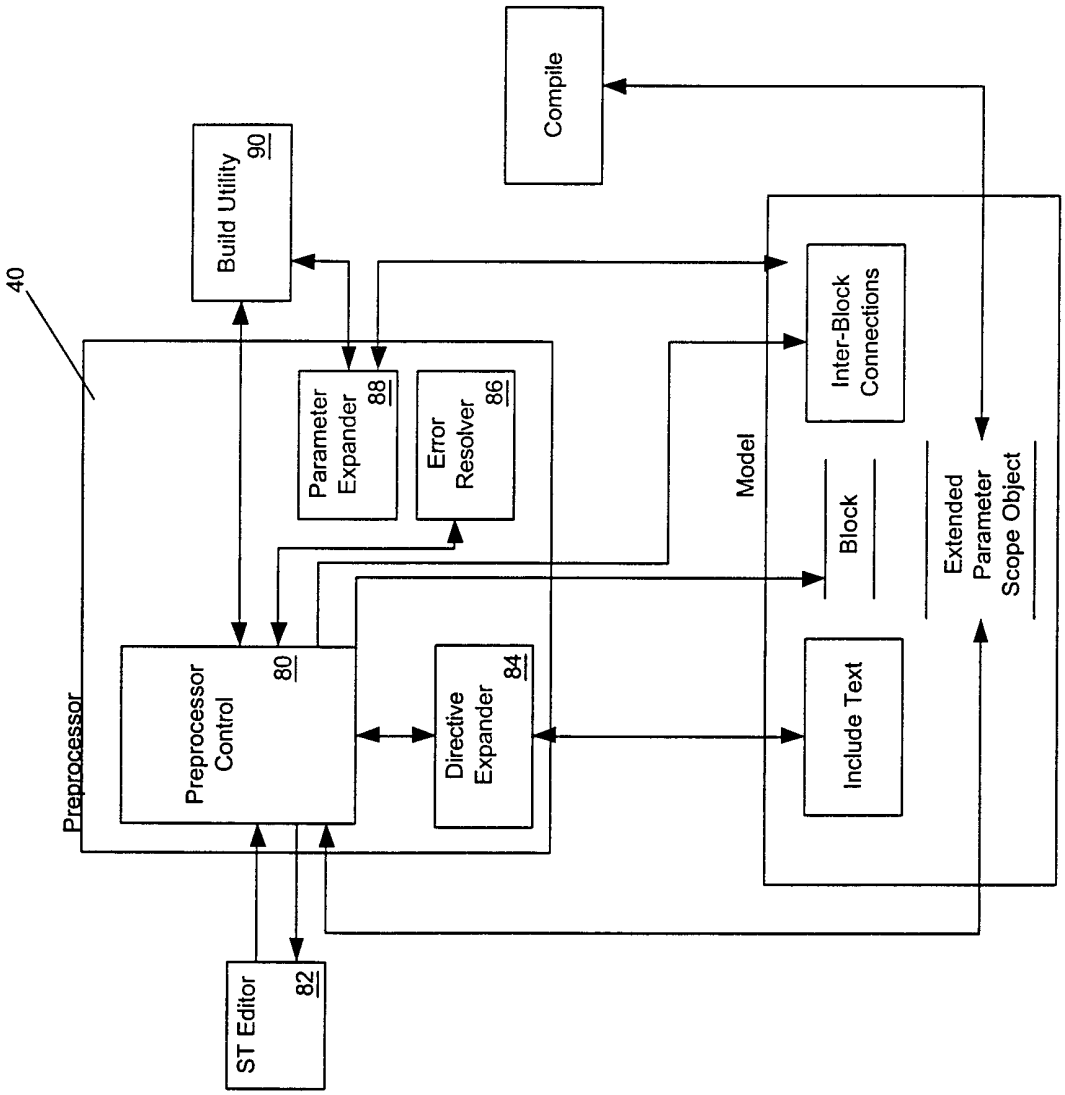

40
Compile
Build Utility
90
Parameter Expander
88
Error Resolver
86
Inter-Block Connections
Model
Block
Extended Parameter Scope Object
Preprocessor
Preprocessor Control
80
Directive Expander
84
Include Text
ST Editor
82

Fig. 5

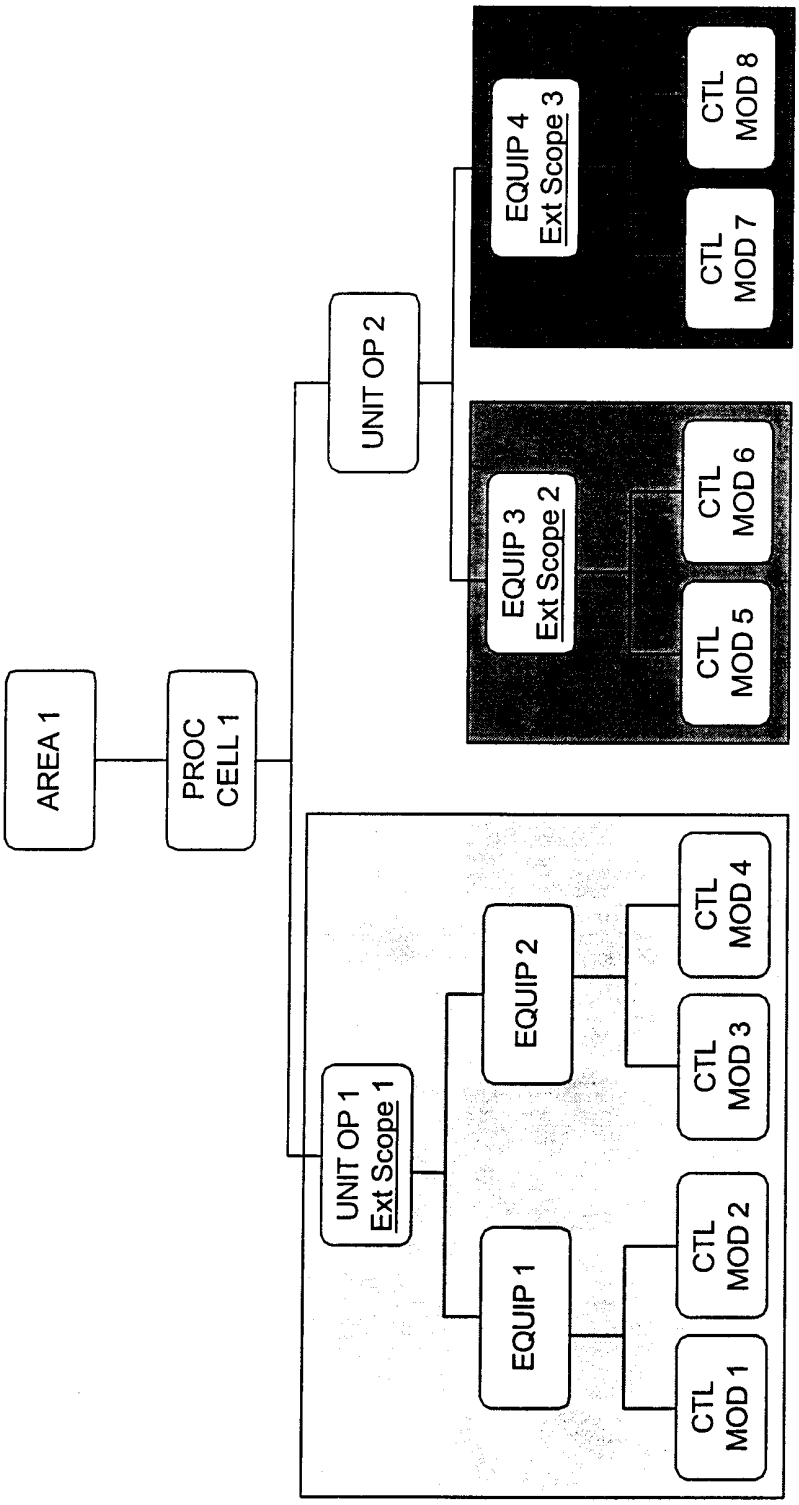

AREA 1
PROC CELL 1
UNIT OP 1 Ext Scope 1
EQUIP 1
CTL MOD 1
CTL MOD 2
EQUIP 2
CTL MOD 3
CTL MOD 4
UNIT OP 2
EQUIP 3 Ext Scope 2
CTL MOD 5
CTL MOD 6
EQUIP 4 Ext Scope 3
CTL MOD 7
CTL MOD 8

Fig. 6

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/13635

**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(7) :G 06 F 3/00
US CL : 345/333; 364/131,188

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 345/333; 364/131,188

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

West, IEEE, ACM, Dialog

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | US 5,598,566 A (PASCUCCI et al.) 28 January 1997, col 25-col 39 | 1-84 |
| Y | US 5,838,563 A (DOVE et al.) 17 November 1998, col 2-col 18, line 20 | 1-21, 44-63 |
| Y, P | US 5,940,294 A (DOVE) 17 August 1999, col 2-col 7 | 22-42, 64-84 |
| Y,P | US 5,980,078 A (KRIVOSHEIN et al.) 09 November 1999, col 5, line 66-col 25, line 27 | 1-84 |
| Y,P | US 6,078,320 A (DOVE et al.) 20 June 2000, col 2-col 8, line 28 | 1-84 |

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

| | | | |
|---|---|---|---|
| * | Special categories of cited documents: | "T" | later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "A" | document defining the general state of the art which is not considered to be of particular relevance | "X" | document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "E" | earlier document published on or after the international filing date | "Y" | document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| "L" | document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) | "&" | document member of the same patent family |
| "O" | document referring to an oral disclosure, use, exhibition or other means | | |
| "P" | document published prior to the international filing date but later than the priority date claimed | | |

| | |
|---|---|
| Date of the actual completion of the international search<br>02 OCTOBER 2000 | Date of mailing of the international search report<br>14 NOV 2000 |
| Name and mailing address of the ISA/US<br>Commissioner of Patents and Trademarks<br>Box PCT<br>Washington, D.C. 20231<br>Facsimile No. (703) 305-3230 | Authorized officer<br>Alvin Oberley James R. Matthews<br>Telephone No. (703) 305-9716 |

Form PCT/ISA/210 (second sheet) (July 1998)★