US 20060248540A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2006/0248540 A1

Stemer et al. (43) Pub. Date: Nov. 2, 2006

(54) AUTOMATIC SOURCE CODE GENERATION

(76) Inventors: **Peter Stemer**, Waldbronn (DE); **Klaus Prasse**, Karlsruhe (DE); **Herbert Anderer**, Waldbronn (DE)

Correspondence Address:
PERMAN & GREEN
425 POST ROAD
FAIRFIELD, CT 06824 (US)

(21) Appl. No.: **11/384,974**

(22) Filed: **Mar. 20, 2006**

(30) **Foreign Application Priority Data**

Apr. 14, 2005 (EP) EP05102938.7

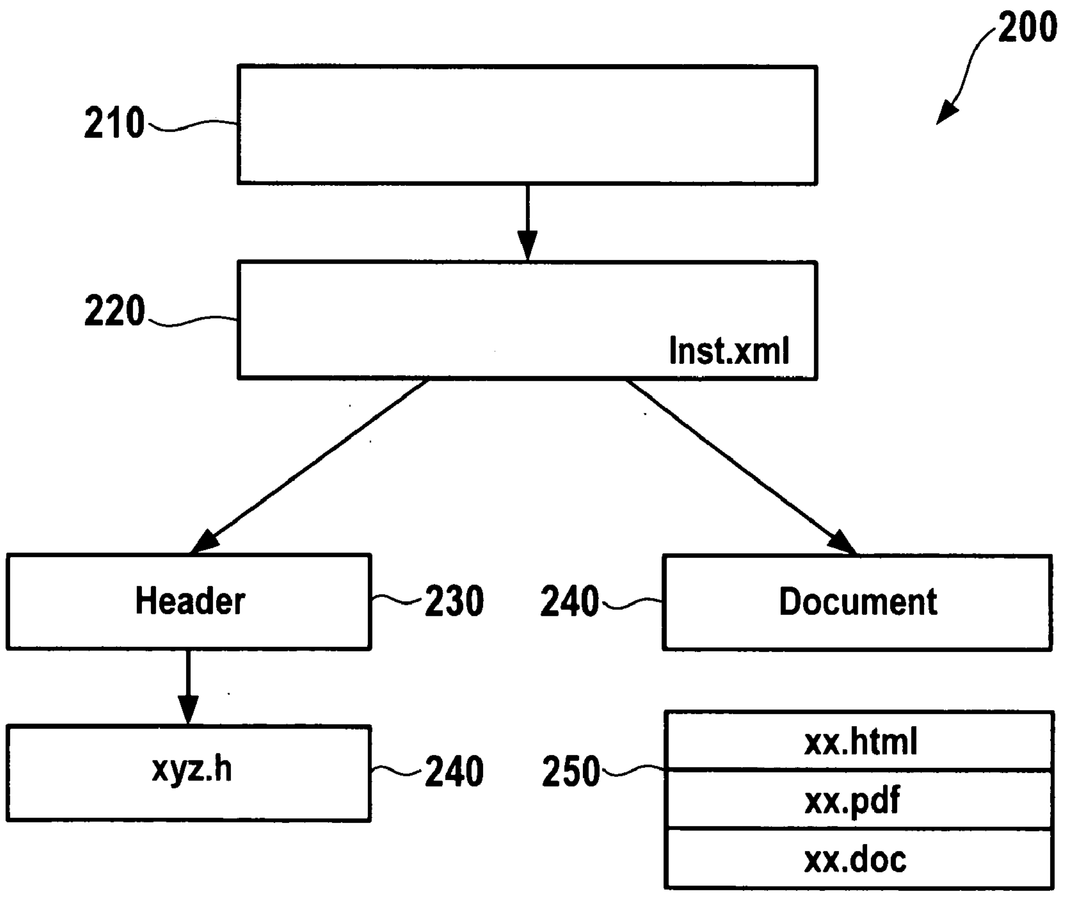
Publication Classification

(51) **Int. Cl.**
G06F 9/46 (2006.01)
G06F 9/44 (2006.01)

(52) **U.S. Cl.** **719/321**

(57) **ABSTRACT**

A data processing device comprising a first generation unit for generating, based on generic data defining a functional interface of an apparatus in general terms, a formalized description of the functional interface of the apparatus, and a second generation unit for generating, based on the formalized description, source code for realizing the functional interface of the apparatus.



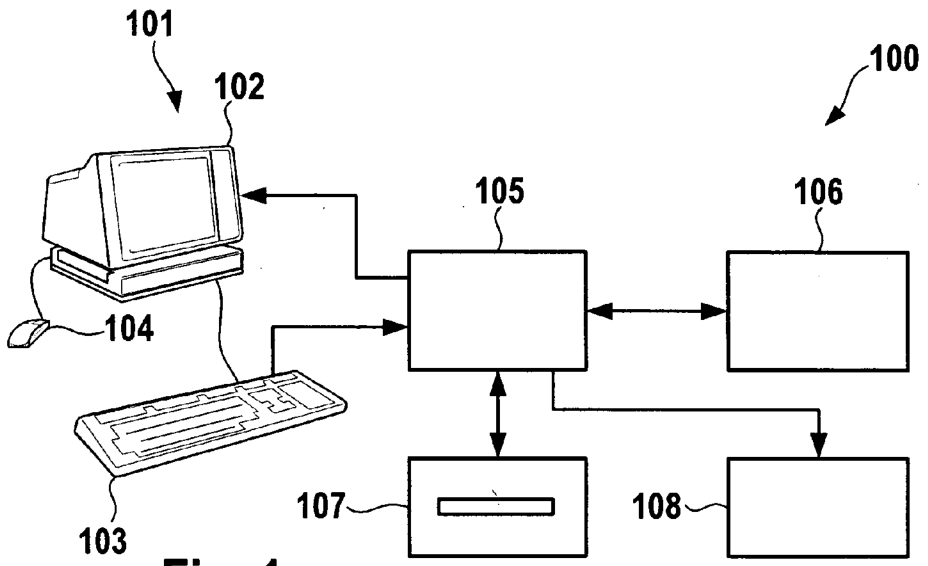

101
102
100
105
106
104
103
107
108

Fig. 1

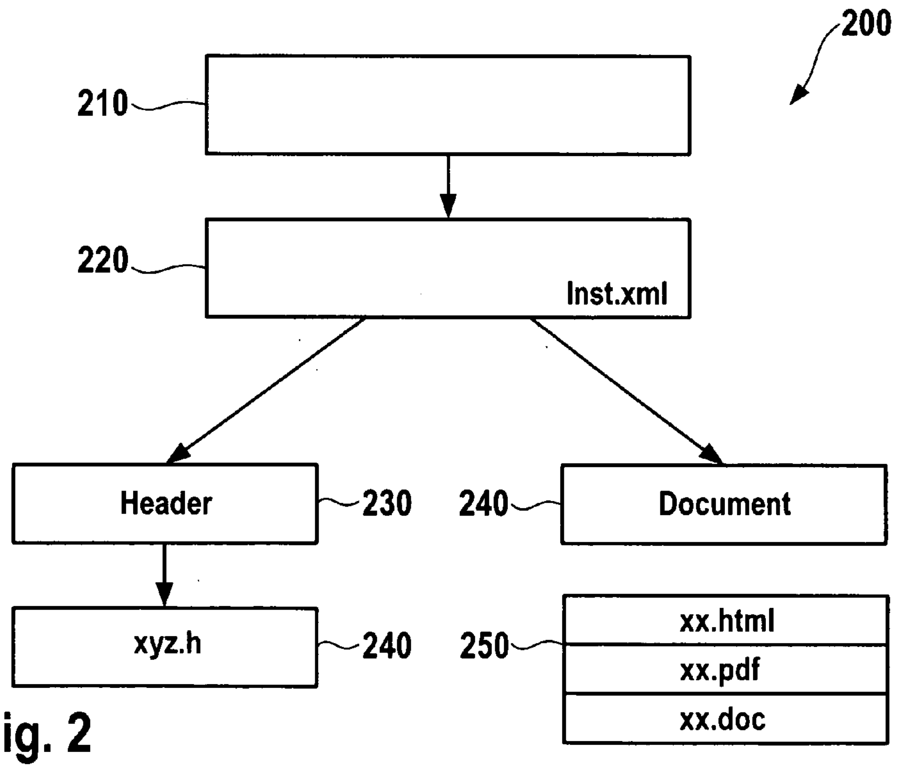

200
210
220
Inst.xml
Header
230
240
Document
xyz.h
240
250
xx.html
xx.pdf
xx.doc

Fig. 2

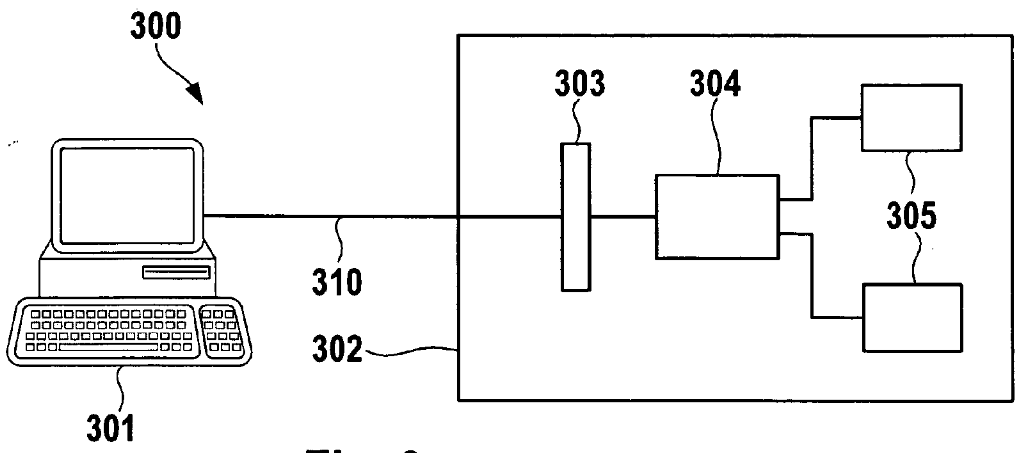
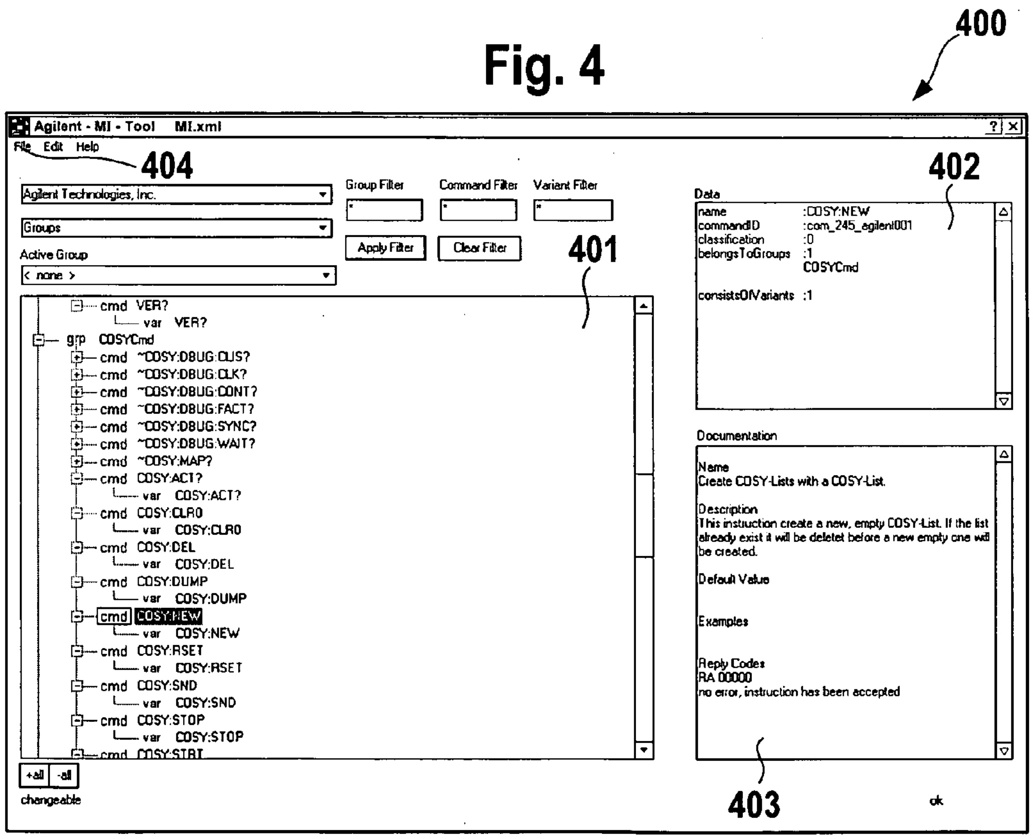


Fig. 3

Fig. 4



```
⊟···· cmd  VER?
      └── var  VER?
⊟── grp  COSYCmd
  ⊞── cmd  ~COSY:DBUG:CLUS?
  ⊞── cmd  ~COSY:DBUG:CLK?
  ⊞── cmd  ~COSY:DBUG:CONT?
  ⊞── cmd  ~COSY:DBUG:FACT?
  ⊞── cmd  ~COSY:DBUG:SYNC?
  ⊞── cmd  ~COSY:DBUG:WAIT?
  ⊞── cmd  ~COSY:MAP?
  ⊟── cmd  COSY:ACT?
        └── var  COSY:ACT?
  ⊟── cmd  COSY:CLR0
        └── var  COSY:CLR0
  ⊟── cmd  COSY:DEL
        └── var  COSY:DEL
  ⊟── cmd  COSY:DUMP
        └── var  COSY:DUMP
  ⊟── cmd  COSY:NEW
        └── var  COSY:NEW
  ⊟── cmd  COSY:RSET
        └── var  COSY:RSET
  ⊟── cmd  COSY:SND
        └── var  COSY:SND
  ⊟── cmd  COSY:STOP
        └── var  COSY:STOP
  ⊟── cmd  COSY:STRT
```

```
name               :COSY:NEW
commandID          :com_245_agilent001
classification     :0
belongsToGroups    :1
                   COSYCmd

consistsOfVariants :1
```

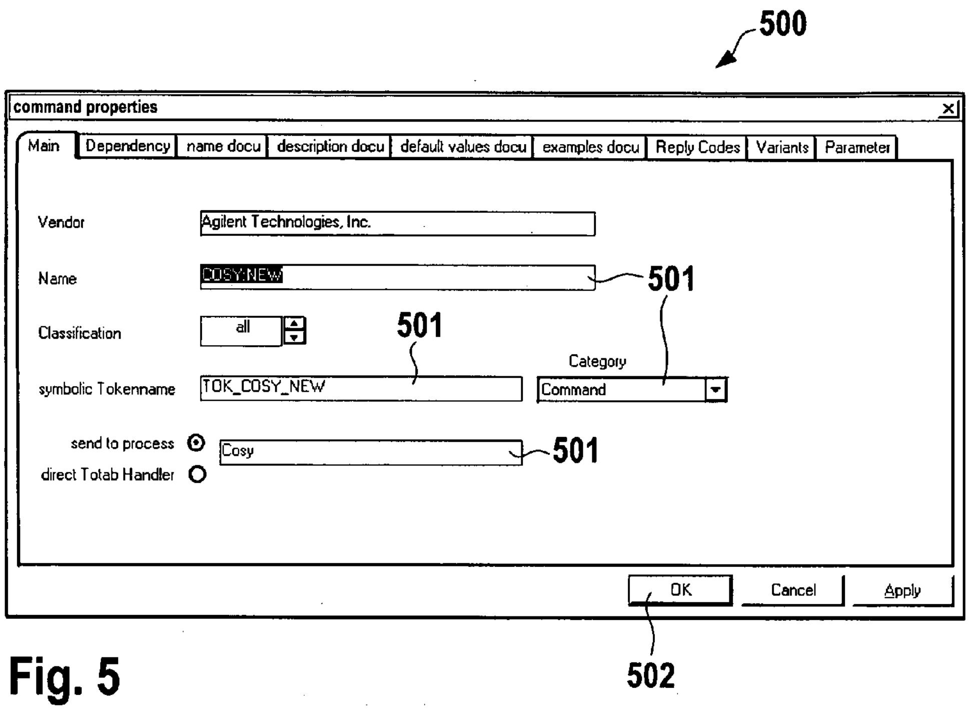
Name
Create COSY-Lists with a COSY-List.

Description
This instuction create a new, empty COSY-List. If the list already exist it will be deletet before a new empty one will be created.

Default Value

Examples

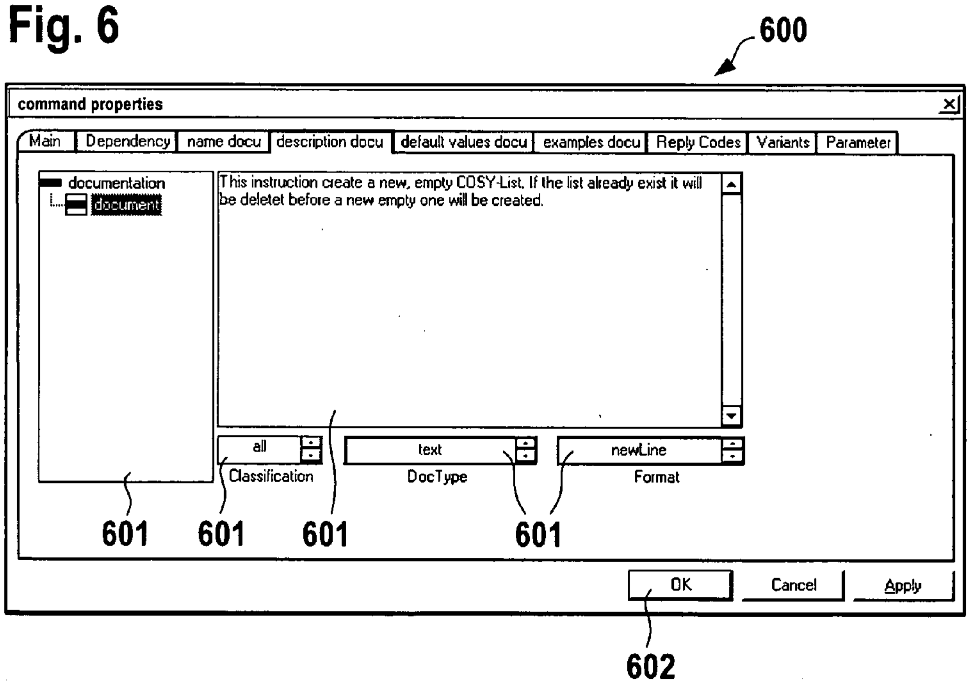
Reply Codes
RA 00000
no error, instruction has been accepted



Vendor: Agilent Technologies, Inc.
Name: COSY:NEW
Classification: all
symbolic Tokenname: TOK_COSY_NEW
Category: Command
send to process [✓] Cosy
direct Totab Handler

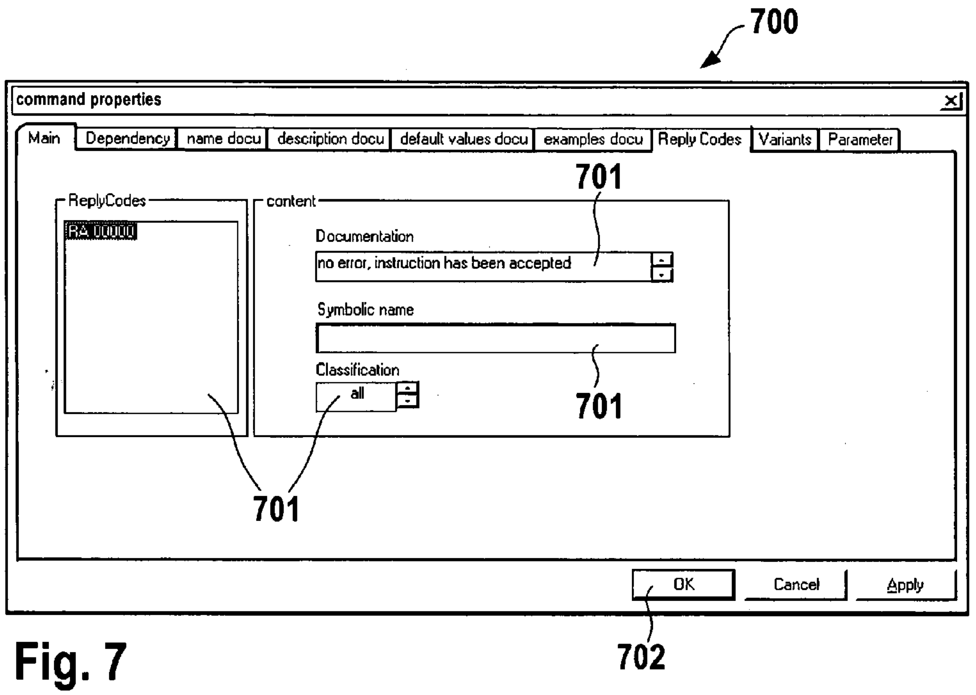
Fig. 5

Fig. 6



This instruction create a new, empty COSY-List. If the list already exist it will be deletet before a new empty one will be created.

Classification: all
DocType: text
Format: newLine



RA 00000

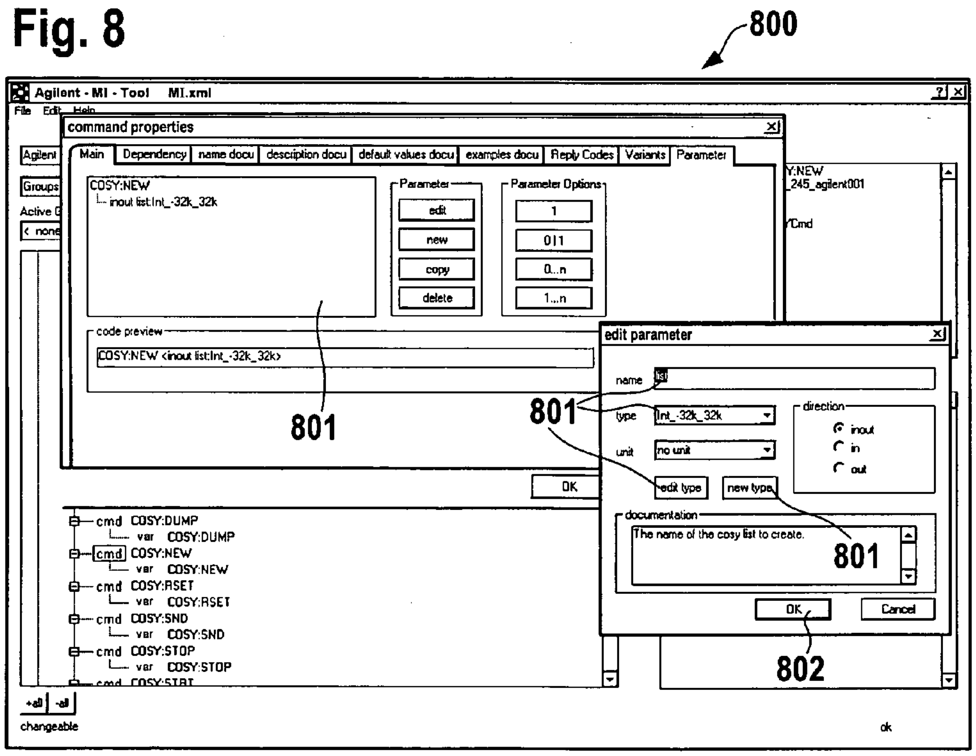
Documentation: no error, instruction has been accepted

Symbolic name:

Classification: all

Fig. 7

Fig. 8



```
COSY:NEW
└ inout list:Int_-32k_32k
```

COSY:NEW <inout list:Int_-32k_32k>

Y:NEW
_245_agilent001
/Cmd

name: list

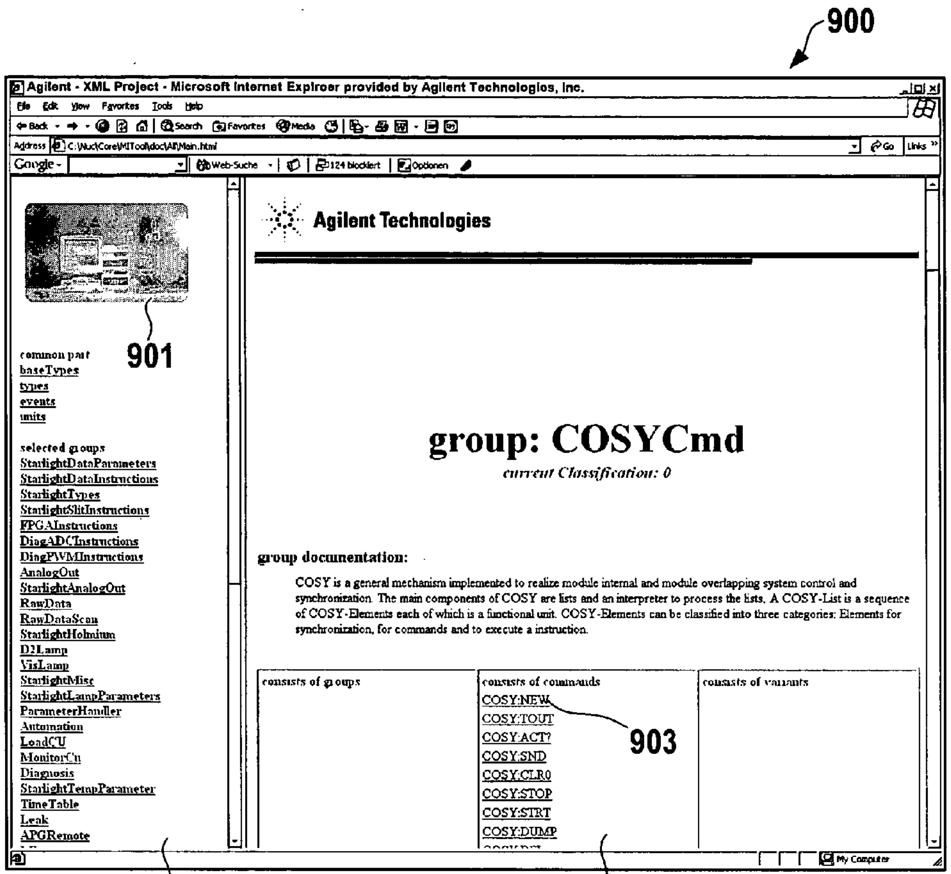
type: Int_-32k_32k

unit: no unit

direction: [✓ inout] in out

documentation: The name of the cosy list to create.

```
⊟—cmd  COSY:DUMP
   └— var   COSY:DUMP
⊟—cmd  COSY:NEW
   └— var   COSY:NEW
⊟—cmd  COSY:RSET
   └— var   COSY:RSET
⊟—cmd  COSY:SND
   └— var   COSY:SND
⊟—cmd  COSY:STOP
   └— var   COSY:STOP
⊟—cmd  COSY:STRT
```


900
Agilent - XML Project - Microsoft Internet Explroer provided by Agilent Technologies, Inc.
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Nuc\Core\MITool\doc\All\Main.html
Go Links
Google Web-Suche 124 blockiert Optionen
Agilent Technologies
901
902
903
My Computer

common part
baseTypes
types
events
units

selected groups
StarlightDataParameters
StarlightDataInstructions
StarlightTypes
StarlightSlitInstructions
FPGAInstructions
DiagADCInstructions
DiagPWMInstructions
AnalogOut
StarlightAnalogOut
RawData
RawDataScan
StarlightHolmium
D2Lamp
VisLamp
StarlightMisc
StarlightLampParameters
ParameterHandler
Automation
LoadCU
MonitorCu
Diagnosis
StarlightTempParameter
TimeTable
Leak
APGRemote

group: COSYCmd

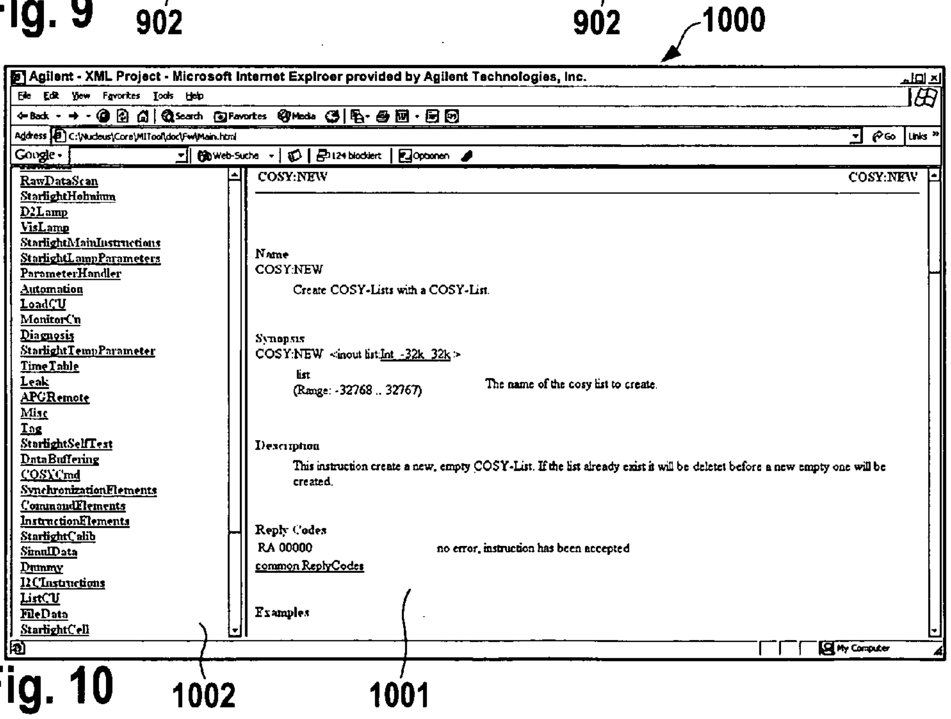
current Classification: 0

group documentation:

COSY is a general mechanism implemented to realize module internal and module overlapping system control and synchronization. The main components of COSY are lists and an interpreter to process the lists. A COSY-List is a sequence of COSY-Elements each of which is a functional unit. COSY-Elements can be classified into three categories: Elements for synchronization, for commands and to execute a instruction.

consists of groups	consists of commands	consists of variants
	COSY:NEW	
	COSY:TOUT	
	COSY:ACT?	
	COSY:SND	
	COSY:CLR0	
	COSY:STOP	
	COSY:STRT	
	COSY:DUMP	

Fig. 9


1000
Agilent - XML Project - Microsoft Internet Explroer provided by Agilent Technologies, Inc.
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Nucleus\Core\MITool\doc\Fw\Main.html
Go Links
Google Web-Suche 124 blockiert Optionen
1002
1001
My Computer

RawDataScan
StarlightHolmium
D2Lamp
VisLamp
StarlightMainInstructions
StarlightLampParameters
ParameterHandler
Automation
LoadCU
MonitorCu
Diagnosis
StarlightTempParameter
TimeTable
Leak
APGRemote
Misc
Tag
StarlightSelfTest
DataBuffering
COSYCmd
SynchronizationElements
CommandElements
InstructionElements
StarlightCalib
SimulData
Dummy
I2CInstructions
ListCU
FileData
StarlightCell

COSY:NEW COSY:NEW

Name
COSY:NEW
Create COSY-Lists with a COSY-List.

Synopsis
COSY:NEW <inout list:Int -32k 32k>

list (Range: -32768 .. 32767)	The name of the cosy list to create.

Description
This instruction create a new, empty COSY-List. If the list already exist it will be deletet before a new empty one will be created.

Reply Codes

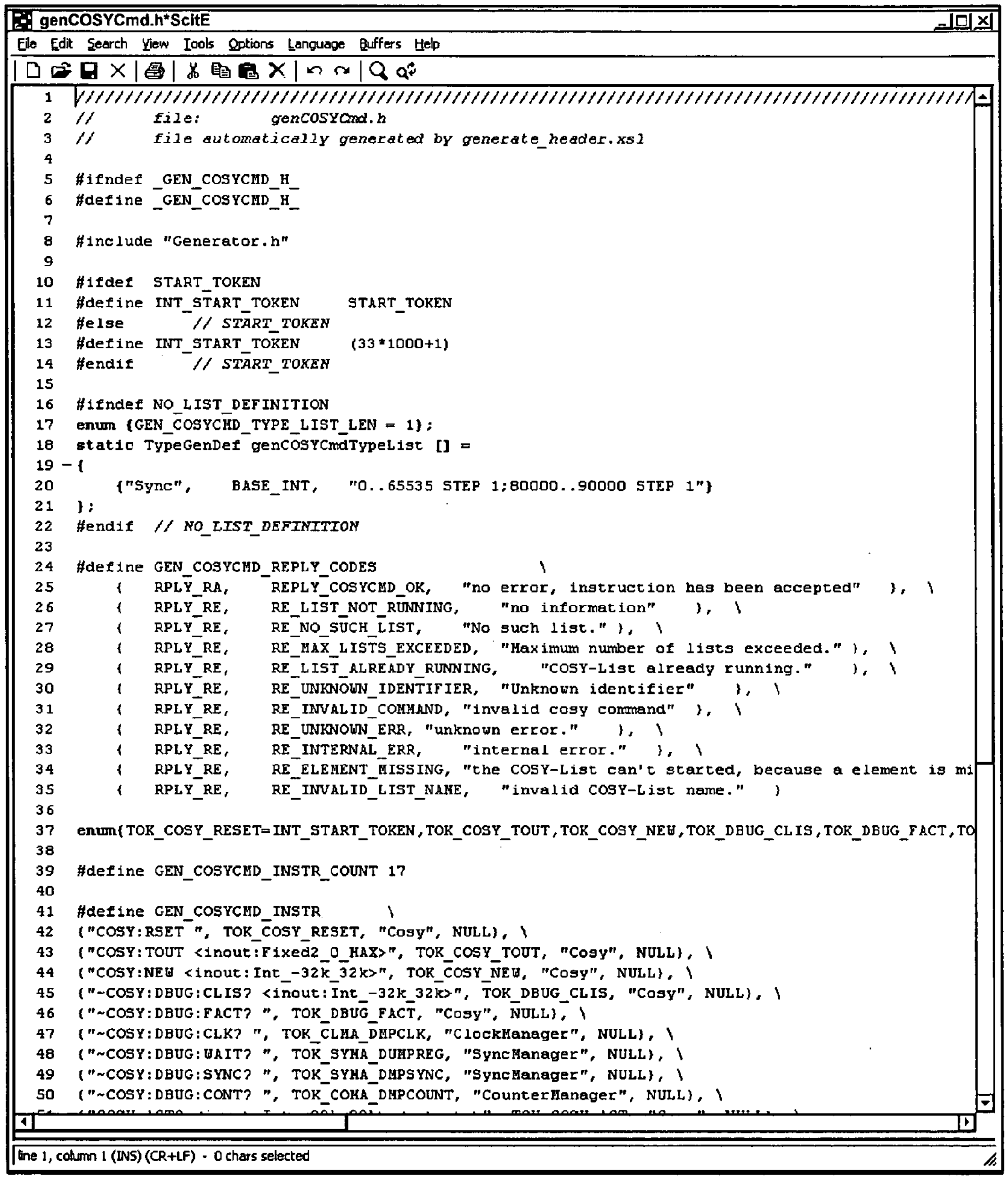
RA 00000	no error, instruction has been accepted

common ReplyCodes

Examples

Fig. 10

1100



```
//////////////////////////////////////////////////////////////////////////////
//       file:        genCOSYCmd.h
//       file automatically generated by generate_header.xsl

#ifndef _GEN_COSYCMD_H_
#define _GEN_COSYCMD_H_

#include "Generator.h"

#ifdef  START_TOKEN
#define INT_START_TOKEN     START_TOKEN
#else        // START_TOKEN
#define INT_START_TOKEN     (33*1000+1)
#endif       // START_TOKEN

#ifndef NO_LIST_DEFINITION
enum {GEN_COSYCMD_TYPE_LIST_LEN = 1};
static TypeGenDef genCOSYCmdTypeList [] =
{
    {"Sync",    BASE_INT,   "0..65535 STEP 1;80000..90000 STEP 1"}
};
#endif  // NO_LIST_DEFINITION

#define GEN_COSYCMD_REPLY_CODES                  \
    {   RPLY_RA,    REPLY_COSYCMD_OK,   "no error, instruction has been accepted"   },  \
    {   RPLY_RE,    RE_LIST_NOT_RUNNING,    "no information"    },  \
    {   RPLY_RE,    RE_NO_SUCH_LIST,    "No such list." },  \
    {   RPLY_RE,    RE_MAX_LISTS_EXCEEDED,  "Maximum number of lists exceeded." },  \
    {   RPLY_RE,    RE_LIST_ALREADY_RUNNING,    "COSY-List already running."    },  \
    {   RPLY_RE,    RE_UNKNOWN_IDENTIFIER,  "Unknown identifier"    },  \
    {   RPLY_RE,    RE_INVALID_COMMAND, "invalid cosy command"  },  \
    {   RPLY_RE,    RE_UNKNOWN_ERR, "unknown error."    },  \
    {   RPLY_RE,    RE_INTERNAL_ERR,    "internal error."   },  \
    {   RPLY_RE,    RE_ELEMENT_MISSING, "the COSY-List can't started, because a element is mi
    {   RPLY_RE,    RE_INVALID_LIST_NAME,   "invalid COSY-List name."   }

enum{TOK_COSY_RESET=INT_START_TOKEN,TOK_COSY_TOUT,TOK_COSY_NEW,TOK_DBUG_CLIS,TOK_DBUG_FACT,TO

#define GEN_COSYCMD_INSTR_COUNT 17

#define GEN_COSYCMD_INSTR       \
{"COSY:RSET ", TOK_COSY_RESET, "Cosy", NULL}, \
{"COSY:TOUT <inout:Fixed2_0_MAX>", TOK_COSY_TOUT, "Cosy", NULL}, \
{"COSY:NEW <inout:Int_-32k_32k>", TOK_COSY_NEW, "Cosy", NULL}, \
{"~COSY:DBUG:CLIS? <inout:Int_-32k_32k>", TOK_DBUG_CLIS, "Cosy", NULL}, \
{"~COSY:DBUG:FACT? ", TOK_DBUG_FACT, "Cosy", NULL}, \
{"~COSY:DBUG:CLK? ", TOK_CLMA_DMPCLK, "ClockManager", NULL}, \
{"~COSY:DBUG:WAIT? ", TOK_SYMA_DUMPREG, "SyncManager", NULL}, \
{"~COSY:DBUG:SYNC? ", TOK_SYMA_DMPSYNC, "SyncManager", NULL}, \
{"~COSY:DBUG:CONT? ", TOK_COMA_DMPCOUNT, "CounterManager", NULL}, \
```

Fig. 11

AUTOMATIC SOURCE CODE GENERATION

BACKGROUND ART

[0001] The present invention relates to automatic source code generation.

[0002] When a new laboratory apparatus, for instance a fluid separation system or a biochemical analysis device, is being developed, source code has to be developed by a human programmer. After a compilation of the source code, it can be executed in the developed apparatus for providing and controlling the functionality of the apparatus. Apart from this, a user manual is written describing the functionality and the instruction interface of the apparatus. With the help of such a user manual, a human user is enabled to operate the apparatus or to adjust the apparatus to her or his preferences.

DISCLOSURE

[0003] It is an object of the invention to provide an improved data processing. The object may be solved by the independent claims. Exemplary embodiments are shown by the dependent claims.

[0004] According to an exemplary embodiment of the present invention, a data processing device is provided comprising a first generation unit for generating, based on generic data defining a functional interface (or a functionality) of an apparatus in general terms, a formalized description of the functional interface (or the functionality) of the apparatus, and a second generation unit for generating, based on the formalized description, source code for realizing the functional interface (or the functionality) of the apparatus. Optionally, a third generation unit may be provided for generating interface documentation out of the formalized description.

[0005] According to another exemplary embodiment, an apparatus for providing a functional interface (or a functionality) is provided, wherein the apparatus comprises a formalized description of the functional interface (or the functionality), and wherein the formalized description is generated based on generic data defining the functional interface (or the functionality) of the apparatus in general terms.

[0006] According to still another exemplary embodiment, a method of processing data is provided, the method comprising generating, based on generic data defining a functional interface (or a functionality) of an apparatus in general terms, a formalized description of the functional interface (or the functionality) of the apparatus, and generating, based on the formalized description, source code for realizing the functional interface (or the functionality) of the apparatus.

[0007] According to yet another exemplary embodiment, a computer-readable medium is provided, in which a computer program of processing data is stored, which computer program, when being executed by a processor, is adapted to control or carry out the above-mentioned method.

[0008] According to still another exemplary embodiment, a program element of processing data is provided, which program element, when being executed by a processor, is adapted to control or carry out the above-mentioned method.

[0009] Embodiments can be partly or entirely embodied or supported by one or more suitable software programs, which can be stored on or otherwise provided by any kind of data carrier, and which might be executed in or by any suitable data processing unit. The data processing scheme can be realized by a computer program, i.e. by software, or by using one or more special electronic optimization circuits, i.e. in hardware, or in hybrid form, i.e. using software components and hardware components.

[0010] One exemplary aspect of embodiment can be seen in the fact that, for assisting in the development of an apparatus like a laboratory apparatus, it is sufficient that a human being inputs generic data defining the functionality of the apparatus, particularly defining a functional interface of the apparatus, to be designed in general terms, that is to say in a verbal but in some sense formalized manner according to a syntax understandable for a human being or close to a human language. The data processing system according to an embodiment may then automatically generate a (more or further) formalized description of the functionality of this apparatus. This formalized description may be a formulation of the desired functionality of the apparatus and may provide instructions in a symbolic logic to be readable by a machine to be a proper basis for generating source code for realizing the functionality of the apparatus. Such automatically generated source code, after being compiled, may be executed in the apparatus to be developed and may then serve to control or provide the functionality of the apparatus.

[0011] According to one aspect of embodiments, a tool for automated creation of source code and/or corresponding documentation from a formalized lab device interface description may be provided.

[0012] According to an aspect of embodiments the, the generated source code may define the interface of the apparatus to be developed, that is to say all functions of the apparatus which may be accessed via the functional interface.

[0013] In other words, when developing a new apparatus, it is made possible by the data processing scheme according to an embodiment that an engineer just defines an entire set of functions of the apparatus (e.g. commands, events, parameters, etc.) accessible via a functional interface in a language according to a unique syntax relatively close to a human language with which the engineer is familiar, for instance via a GUI (graphical user interface) based template on a computer monitor. The system according to an embodiment is then capable to translate or transfer this code in a more formalized representation. Such a more formalized representation can be, for instance, XML (Extensible Markup Language). This code may be independent from a special kind of language used by the human user.

[0014] Such a formalized description, for instance in the form of an XML file, may then be used according to an embodiment for generating computer-readable program source code which is, after compilation and when being executed by a processor, capable of realizing the entire functional interface of the apparatus to then readily developed. Thus, source code may be generated in accordance with the generic data and may be generated automatically after having specified the desired functionality in general terms. Consequently, a programmer never has to touch the automatically generated source code, which in turn eliminates the problem that implementation and documentation may differ. The described scheme according to an embodiment may thus allow designing any apparatus, for instance a laboratory apparatus, in a significantly simplified manner, faster and cheaper. Generated source code may be implemented in the apparatus without the necessity that a human programmer has to review the code.

[0015] In a scenario in which only one or some particular functions of an already developed apparatus shall be modified for developing a similar but functionally modified apparatus, it is easily possible to just "reload" the interface description in the formalized language to the data processing device according to an embodiment so that it is not necessary to program everything again from the very beginning. It may be sufficient to selectively modify one or more parameters related to a function to be modified. Then, it may be sufficient to just press some kind of "Generate Code" button on a screen or a keypad to instruct the system to generate the updated or modified source code. Moreover, retrofitting of particular functions to an already existing apparatus can be performed easily.

[0016] The term "generic data defining a functionality of an apparatus in general terms" may particularly denote the opportunity to provide the system with data in a language in which the desired functionality may be articulated by a human being. Thus, the information defining the desired functionality can be verbalized in a manner with which a human user is familiar. For instance, such generic data defining the functionality of an apparatus in general terms may be "the device shall include a command which initiates the mixing of a first liquid having a first concentration with a second liquid having a second concentration". This non-formalized description may be input by a user via a given data input structure. For instance, this information may be input in a structured manner via a mask with data input fields in which the different pieces of information are insertable by a human user. This can, for instance, be realized with a graphical user interface on which computer windows are visualized allowing a user to input the generic data. In other words, a system according to an embodiment may provide a user with a communication interface which may allow the user to enter the data without the necessity that the user cares about the question if the input data can be read by a machine.

[0017] The input fields of such a generic data inputting tool may then be coupled, combined or linked according to a complex data connection structure (which is usually invisible for a human user) which may process the data in a manner as to generate a code which is a formalized description.

[0018] The term "formalized description" may particularly denote a description in a language which can be read or interpreted by a machine or an algorithm without the interference of human intelligence. The formalized or logical description may be any program code encoding the functionality of the apparatus in a format that needs not to be understood by a human being.

[0019] According to an exemplary embodiment, the formalized description of the functionality (for instance in the form of an XML-file) can be stored in the produced apparatus, wherein it may be possible to be accessed from an external device. This XML-file provides a formal description of all functions, which may be realized with the apparatus.

[0020] A developer may simply specify which functions an apparatus to be designed shall provide. This group of functions may then be transferred in a formalized description like an XML-file, particularly in an automatic manner. From this formalized description, a header for firmware, that is to say compilable source code, may be generated which may include all information necessary for the interface of the apparatus to be designed. For generating the source code, the system may optionally access a program library containing particularly standard routines, which may be used by the machine to generate source code in a short time.

[0021] Optionally, the formalized description may not only be used as a basis for generating the source code, but may also be used as a basis for generating a documentation or manual for a user, for instance as an HTML file, a Word file or an PDF file.

[0022] Thus, it may be prevented that the documentation of a device's instruction interface and its actual implementation differ. The developer (for good reasons) often changes some instructions or parameters but forgets to consistently update the documentation. This results in incompatibilities with controlling software and confusion for the user. Such incompatibilities may be securely avoided by the described aspect.

[0023] Since it may be made possible according to embodiments, from one and the same formalized description and in an automatic manner, source code on the one hand and a documentation readable by a human user on the other hand, it may be guaranteed that the documentation delivered to a customer coincides with the functionality actually provided by the apparatus. In other words, documentation and source code are conform with high reliability.

[0024] Further, it may be dispensible that a developer has to manually write a documentation. Instead of this, the documentation may be generated automatically, since all information required for this purpose is derivable from the generic data and thus also from the formalized description. Therefore, a perfect conformity between documentation and source code may be achieved.

[0025] According to an exemplary embodiment, it may be possible to generate at least two different documentation files according to at least one predetermined parameter. For instance, a documentation generated for a customer may be different from a documentation generated for the employees of a company developing the apparatus. For example, a documentation accessible for the employees of the company developing the apparatus may contain confidential information which may not be accessible in the documentation for the customers. Thus, any item of the input generic information may be assigned with a flag or the like indicating that a particular piece of information may be confidential. However, distinguishing between confidential and non-confidential data is not the only possible criteria for deciding which information should be included in which type of documentation. Another exemplary criteria for distinguishing which information should be included in which type of documentation is the importance of a particular piece of information. Thus, it may be possible to generate a brief summary documentation of the functionality including only information classified as particularly important, and to generate a detailed documentation of the entire functionality including all information regardless its importance.

[0026] Furthermore, when extending or modifying the functionality of an apparatus to be designed, it may be no more required to review and rewrite the entire documentation again, but it may be sufficient that the generic data or the formalized description is only punctiformly modified. Then, a “Generate” button may be pressed for generating both modified documentation and modified source code.

[0027] Thus, the functionality of the apparatus to be designed may be extendable in a simple manner, and it may be ensured that an automatically generated manual fits to the specification of the device according to the source code.

[0028] Thus, according to one aspect of embodiments, a set of commands for an apparatus to be developed (e.g. functions, parameters, events) may be defined by a developer and described in a colloquial language or common speech. From this generic data formulated in general terms, a formalized description of the desired set of commands may be generated, for instance in XML, using a data connection structure for processing and evaluating the different generic data items. From this formalized description, source code for firmware may be generated. Simultaneously, the formalized description can be used as a basis for generating a user manual in an automated manner. Thus, discrepancies between documentation and control code may be securely avoided. The term “common speech” may particularly denote a language which is in some sense formal (since it may be restricted by particular rules), but which may still be understood by a (skilled) human user. Further, the term “common speech” may particularly denote an unambiguous syntax which is close to human speech.

[0029] An example for a command which may be input as generic data is “COMP [% B], [% C], [% D]”. “COMP” may a keyword for a particular function. The three items “[% B]”, “[% C]”, and “[% D]” may be parameters for specifying the function in more detail. The keyword COMP may define the functionality of a pump for treating four liquid components A, B, C, D which shall be mixed according to preset percentages % B, % C, % D, wherein the percentage of component A results from the frame condition that the sum of A, B, C, and D should be 100%. Based on such generic data like “COMP [% B], [% C], [% D]”, corresponding source code may be generated automatically, which can be executed by the apparatus to be developed. Another example for a piece of functionality to be defined in a common speech is “EV 0560”. This may define that the apparatus to be designed provides as an output that a particular event has happened, for instance “Analysis Completed” when an entire analysis procedure has been carried out successfully.

[0030] For instance, by simply pressing a button, the data structure input by the human user may be connected and processed automatically so that an XML file will be generated. The translation into the XML file may be performed by a particular software element. This software element connects the different items of plaintext. In other words, the data related to the input fields in the user interface may be linked in such a manner that the XML file or any other formalized description is automatically generated.

[0031] The Extensible Markup Language (XML) may be considered as a simplified subset of SGML, capable of describing many different kinds of data. A purpose is to facilitate the sharing of structured text and information. Languages based on XML (for example RDF, RSS, MathML, XSIL and SVG) are themselves described in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.

[0032] Such an XML file may be processed using a Stylesheet. A Stylesheet may be used in order to separate the form of a description from the content of a structured document (for instance XML). Stylesheets may allow to interpret contents in dependence of an output device. For this purpose, the content does not have to be changed. Examples for Stylesheet languages are CSS, XSL and DSSSL. Source code readable by a computer may be generated based on the XML file. Such a source code may be, for instance, C++ code.

[0033] Furthermore, a Stylesheet may be used in combination with the XML file to automatically generate the manual which may be directly readable by a human user. Such a documentation may be a HTML file or a PDF file.

[0034] In this specification, the term “functionality” may particularly denote one or more functional items which an apparatus to be defined may service or provide, or an instruction repertoire. It may also denote the entirety of all functions which the apparatus offers. “Functionality” may include commands which the apparatus understands and which the apparatus may carry out. “Functionality” may include actions which the apparatus may perform by itself, without external control or trigger, for instance the indication of a particular operation state.

[0035] In the following, further exemplary embodiments of the data processing device will be described. However, these embodiments also apply for the apparatus for providing a functionality, for the method of processing data, for the computer-readable medium and for the program element.

[0036] The “generic data” may define the functionality of the apparatus in a language understandable by a human being. In contrast to this, such generic data may be incomprehensible for a machine like a computer. The generic data may be input in plaintext or clear text in a manner which is close to a human articulation. By structuring input fields of the receiving unit in order to enable a human user to input information in human language, but already pre-structured, it is possible to communicate with a human developer and to simultaneously cause the human developer to enter the data according to a given order. Thus, the receiving unit may allow a communication in general terms with the human being, but may set the course for a subsequent translation of the instructions into a machine language, similar like a form or a questionnaire.

[0037] Furthermore, the “formalized description” may define the functionality of the apparatus in a language interpretable, compilable or understandable by a machine. Thus, the data processing device may abstract and/or formalize the generic data to reformulate it in a machine-readable manner.

[0038] The data processing device may comprise a third generation unit for generating, based on the formalized description, a user manual documenting the functionality of the apparatus. Such a manual can be generated automatically from the formalized description including all required information so that it may be dispensible that a detailed user manual has to be written manually by a human in a cumbersome manner. Furthermore, consistence between documentation and actually implemented functionality, reflected by the source code, may be achieved.

[0039] The third generation unit of the data processing device may be adapted to generate the user manual in a manner to be displayable and/or to be printable. Thus, the user manual may be displayed on a display device of a graphical user interface (GUI), for instance in the form of an HTML file comprising a plurality of linked pages. A user may use an input tools like a computer mouse to click through the pages, or may electronically search the documentation file to rapidly find a subject of interest. The documentation may also be provided in a form that it can be printed as a hard copy using a printer device. For this purpose, it may be appropriate to generate the documentation as a PDF file or as a Word file.

[0040] The data processing device may comprise a receiving unit for receiving the generic data defining the functionality of the apparatus. The receiving unit may be a user interface via which a user may input the generic data. Particularly, such a receiving unit may include a Graphical User Interface (GUI) via which a human user may input data. Such a graphical user interface may include a display device (like a cathode ray tube, a liquid crystal display, a plasma display device or the like) for displaying information to a human operator, like masks in the form of Windows in which input fields may be provided. Such a graphical user interface may further comprise an input device allowing a user to input specification data or to provide the system with control commands. Such an input device may include a keypad, a joystick, a trackball, or may even include a microphone of a voice recognition system. The GUI may allow a human user to communicate with the system in a bidirectional manner.

[0041] Furthermore, the receiving unit may be adapted to receive the generic data defining a complete functionality of the apparatus. In other words, an entire, completingly defined functionality may be specified via the receiving unit so that a virtual development of a device is possible. The realization of the software for controlling such an apparatus may be generated in a computer-based manner from the input specification. Thus, an apparatus construction set may be provided according to an embodiment.

[0042] The receiving unit may be adapted to receive an entirety of all commands executable by the apparatus and/or an entirety of all actions performable by the apparatus. Such a command may allow a user of the apparatus to access or call a particular function which is intended to be carried out. An event may denote an action which may be performed automatically by the apparatus in the presence of a particular scenario.

[0043] The first generation unit of the data processing device may be adapted to generate, based on the generic data, an Extensible Markup Language file as the formalized description of the functionality of the apparatus. However, using an XML file is only an example for an appropriate formalized description language, any other suitable formalized description or programming language may be used as well.

[0044] The second generation unit may be adapted to generate source code readable by a parser of the apparatus. A parser may particularly denote a program or a physical entity which is capable to decide whether an input data set is compatible with the grammar of a particular language. Parsing may include a syntactic check of the input data set. Thus, the source code may be read by the parser of the apparatus.

[0045] The second generation unit may be adapted to generate the source code using a Stylesheet.

[0046] Furthermore, the second generation unit may be adapted to generate the source code in any appropriate program language, which may be object-oriented or not. Exemplary possible program languages are C, C++, C#, Pascal, Basic, Fortran, or Java.

[0047] The data processing device may be adapted in such a manner that the apparatus to be designed may be at least one of the group of a measurement device for performing a measurement in a coupled or connected measurement environment, a sensor device, a test device for testing a device under test, a device for chemical, biological and/or pharmaceutical analysis, a fluid separation system adapted for separating components of a fluid, and a liquid chromatography device. However, these fields are only exemplary, and the development of any other apparatus realizing a specified functionality is possible as well.

[0048] Thus, the apparatus can be a measurement device which may perform any kind of measurement.

[0049] The apparatus can also be any kind of sensor sensing any physical, chemical or other parameter like temperature, humidity, pressure.

[0050] Further, the apparatus can be realized as a test device for testing a device under test (DUT). For testing electronic devices, in particular integrated electronic circuits providing digital electronic output signals, a test or stimulus signal may be fed to an input of the DUT, and a response signal of the DUT may be evaluated by an automatic test equipment, for example by comparison with expected data.

[0051] In a realization of the apparatus as a device for chemical, biological and/or pharmaceutical analysis, functions like (protein) purification, electrophoresis investigation of solutions, or chromatography investigations may be realized by the analysis device.

[0052] According to another exemplary embodiment, the apparatus to be designed may be a fluid separation system adapted for separating compounds of a fluid. Such a fluid separation system may comprise a fluid delivering unit adapted for delivering fluid, a separation unit adapted for separating compounds of the fluid and to provide at least one separated component.

[0053] According to another exemplary embodiment, the data processing device may comprise a storage unit for storing the generic data (for instance received by the receiving unit), and may comprise a modification unit allowing to modify the generic data stored in the storage unit to modify the functionality of the apparatus. In other words, a set of generic data which has been input before (for instance in the frame of the development of another apparatus) can be reloaded in the user interface, and a selective modification of individual functions may be carried out. This may allow to generate source code and/or a user manual in a simple manner without the necessity to start developing the apparatus from the very beginning.

[0054] In the following, exemplary embodiments of the apparatus for providing a functionality will be described. However, these embodiments also apply for the data processing device, for the method of processing data, for the computer-readable medium and for the program element.

[0055] Particularly, the apparatus may comprise an interface via which the formalized description of the functionality of the apparatus is providable to an entity connected to the interface. Thus, the formalized description (for instance formulated in accordance with a particular industrial standard) may be stored within the apparatus and may be downloaded via the interface. This allows a user to get a fast overview of the entire functionality of an apparatus.

BRIEF DESCRIPTION OF DRAWINGS

[0056] Objects and many of the attendant advantages of embodiments will be readily appreciated and become better understood by reference to the following more detailed description of embodiments in connection with the accompanied drawings. Features that are substantially or functionally equal or similar will be referred to by the same reference signs.

[0057] **FIG. 1** shows a data processing device according to an exemplary embodiment.

[0058] **FIG. 2** shows a schematic flow diagram illustrating a method of processing data according to an exemplary embodiment.

[0059] **FIG. 3** shows an apparatus for providing a functionality according to an exemplary embodiment.

[0060] **FIG. 4** to **FIG. 11** show screenshots in accordance with a data processing device according to an exemplary embodiment.

[0061] The illustration in the drawing is schematically.

[0062] In the following, referring to **FIG. 1**, a data processing device **100** according to an exemplary embodiment will be described.

[0063] The data processing device **100** comprises a graphical user interface **101** including a monitor **102**, a keypad **103** and a computer mouse **104**. Via the graphical user interface **101**, a human user may input generic data defining a functionality of a biochemical analysis device to be designed. For instance, a sequence of computer windows may be displayed on the monitor **102** which may include input fields allowing a user to input specification information concerning the biochemical analysis device to be developed, that is to say to input information relates to all commands which the biochemical analysis device shall understand, all actions which the biochemical analysis device shall be capable of carrying out, or the like.

[0064] The menu-based structure of the window architecture and the data input fields displayed on the monitor **102** allow a user to specify the functionality of the biochemical analysis device to be designed or realized in general terms and in a language of a human user. Particularly, the human user may be directed through a sequence of interconnected or linked windows to allow to enter the input data in the input data fields in an intuitive manner.

[0065] The data processing device **100** further comprises a central processing unit (CPU) **105** which receives the generic data entered via the graphical user interface **101** and which is capable for generating, based on the input generic data defining the functionality of the biochemical analysis device in general terms, a formalized description of the functionality of the biochemical analysis device in the form of an XML file. In other words, the data structure received by the GUI **101** is analyzed and processed by the CPU **105** to generate the XML file containing the formalized description of the entire functionality.

[0066] The plaintext data entered in the data input fields and the XML file can be stored in a storage unit **106**. The storage unit can be any kind of storage unit, for instance a RAM memory, an EEPROM, a flash memory, an MRAM, an FRAM, an SRAM, or the like.

[0067] The CPU **105** can access the XML file stored in the storage unit **106**. The CPU **105** is capable of generating, based on the XML file stored in the storage unit **106**, source code for realizing the functionality of the biochemical analysis apparatus. For this purpose, the CPU **105** may also access program libraries which may also be stored on the storage device **106** and which may contain a collection of standard program routines.

[0068] The generated source code can be stored on a further storage device **107** which may be a harddisk or a removable storage cartridge, like a USB stick. Thus, on the storage device **107**, the source code for controlling the apparatus to be designed is stored.

[0069] The CPU **105** may further generate, based on the XML file stored in the storage unit **106**, a user manual file documenting the functionality of the apparatus. This user manual file may be stored on the further storage device **107** and/or may be sent to a printer **108** to generate a hardcopy of the user manual. The manual file can be stored for instance a HTML file, allowing a user to read the manual on the display device of a computer.

[0070] **FIG. 2** shows a flow diagram **200** illustrating a method of processing data according to an exemplary embodiment.

[0071] The diagram **200** shown in **FIG. 2** schematically illustrates generic data **210** which may be input by a user in general terms/in a human language, for instance via the GUI **101** of **FIG. 1**.

[0072] The generic data **210** defining a functionality of the biochemical analysis device to be designed are translated automatically in a formalized description, that is to say into an XML file **220** denoted as lnst.xml.

[0073] This XML file **220** includes all information related to the functionality of the apparatus in accordance with the generic data or specification 210, however in a machine-readable format.

[0074] Using a Stylesheet, a file Header **230** is generated based on the XML file **220**. The file Header **230** is a basis for source code **240** in C++ language, denoted as a file xyz.h. This source code may be used in the biochemical analysis device for providing or controlling its functionality.

[0075] Parallel to the generation of the source code **240**, a Stylesheet may be used to generate a user documentation Document from the file Inst.xml. Particularly, documentation files **250** may be generated as a user manual in the form of an HTML file (xx.html), a PDF file (xx.pdf) and/or a Word file (xx.doc).

[0076] **FIG. 3** shows an exemplary embodiment of a system **300** including a computer or workstation **301** and a biochemical analysis apparatus **302** developed in accordance with embodiments of the invention.

[0077] The apparatus **302** comprises an interface **303** via which the computer **301** may communicate with the apparatus **302**. Further, a parser **304** is provided in the apparatus **302** which is capable of interpreting or analyzing source code. The parser **304** may be coupled to a plurality of executing units **305** for particularly executing functions using compiled code. The communication between the computer **301** and the apparatus **302** may be realized via a network **310** which may be a LAN (Local Area Network).

[0078] An XML file (like Inst.xml, see **FIG. 2**) defining the entire functionality of the apparatus **302** can be stored in the apparatus **302**. This XML file can be downloaded to the computer **301** coupled to the apparatus **302** via the interface **303**. Thus, the computer **301** can get all required information related to the functions realizable by the apparatus **302**.

[0079] In the following, referring to **FIG. 4** to **FIG. 11**, screenshots of a Windows-based application according to an exemplary embodiment are shown, that is to say a user interface via which the functionality according to the embodiments of the invention may be carried out.

[0080] In the following, referring to **FIG. 4**, a screenshot **400** of a main window of a system according to an exemplary embodiment will be described.

[0081] The computer window **400** shows a user interface via which a human user may input a specification of a laboratory device to be developed (including all commands, answers and events which the laboratory device to be designed shall offer).

[0082] In a sub-window **401**, a previously generated XML file is displayed which has been loaded into the memory of a computer on which the described application runs. This XML file includes all commands of an apparatus which has already been developed beforehand. Since the code in the first sub-window **401** is an XML code, it is difficult to read for a human user.

[0083] As can be seen in **FIG. 4**, a portion of the XML file denoted as COSY:NEW has been highlighted by a user (for instance using a computer mouse or the like) in the first sub-window **401**. As a consequence of this action, a second sub-window **402** displays specification information concerning the command COSY:NEW. In a third sub-window **403**, corresponding documentation information is included which explains the command COSY:NEW in clear text.

[0084] Each of the commands defined in the XML file shown in the first sub-window **401** can be displayed in clear text in the sub-windows **402**, **403** to enable a human user to understand a particular functionality of interest, like COSY:NEW.

[0085] Although the screenshot **400** relates to an already existent apparatus, it is possible to change any of the data included in the sub-windows **402**, **403** or the structure of the XML file in the window **401** to update or modify an already generated device, for instance to change a particular function of such a device.

[0086] For generating source code and a user manual in accordance with the modified apparatus to be designed, after having input all modifications in the corresponding data fields (see explanation below), a menu list can be activated by clicking on selection field “File”**404** in the menu bar of **FIG. 4**. In the menu list which then becomes visible in the window **400**, it is possible to press a “Generate” button to cause the software to automatically generate the source code and the user manual. When pressing this “Generate” button, firmware, that is to say source code defining the functionality of the modified apparatus (see **FIG. 11**), and a documentation (see **FIG. 9**, **FIG. 10**) may be generated.

[0087] In the following, referring to **FIG. 5** to **FIG. 8**, computer windows **500**, **600**, **700**, **800** will be described allowing a user to input generic data for defining a functionality of an apparatus to be designed.

[0088] Referring to **FIG. 5**, in the computer window **500**, a plurality of input fields **501** are provided allowing a user to define a detailed specification concerning the command COSY:NEW. After having modified the commands as desired by a user, an OK button **502** may be pressed so that the corresponding data are accepted.

[0089] Furthermore, via a sub-window description docu shown in the window **600** of **FIG. 6**, a documentation may be defined or modified via input fields **601**. After having pressed an OK button **602**, these documentation information will be included for subsequent processing.

[0090] Referring to **FIG. 7**, a reply codes sub-window of a window **700** is shown. Via data fields **701**, reply code information may be input and may be confirmed by pressing on an OK button **702**.

[0091] **FIG. 8** shows a further dialog window **800** allowing to modify command parameters in various data input fields **801**. After confirmation by pressing an OK button **802**, these parameters related to particular commands are accepted.

[0092] After having entered all data relevant for the functionality of the apparatus to be designed in the data input fields shown in **FIG. 4** to **FIG. 8**, a user may activate the above-described “Generate” button in the menu **404** shown in **FIG. 4**. Consequently, the XML file shown in the sub-window **401** is updated, or a completely new XML file is generated reflecting the entire defined functionality as a formalized description.

[0093] Such an XML file can be stored and may be, for instance, provided at an interface of the physical apparatus to be designed so that any user can access this XML code describing the whole functionality in formalized description.

[0094] However, by pressing the “Generate” button as described above, the software generates automatically a documentation as an HTML file, as shown in **FIG. 9** and **FIG. 10**. The information needed for generating this documentation originates from the XML file.

[0095] **FIG. 9** shows a communication window **900** documenting the functionality of the apparatus which has been designed. Different communication sub-windows **902** are shown including links to other pages which include further detailed information concerning the various aspects mentioned as headlines in the windows **902**.

[0096] For instance, by pressing a COSY:NEW button **903**, a window **1000** as shown in **FIG. 10** will pop up. In this window **1000**, a HTML user manual related to the command COSY:NEW is shown in a sub-window **1001**. Another sub-window **1002** allows to access information concerning other commands.

[0097] However, by pressing the "Generate" button as described above, not only the documentation shown in **FIG. 9** and **FIG. 10** is generated. Furthermore, C++ source code is automatically generated which may be used to control the apparatus **901**. Such source code is shown in a window **1100** of **FIG. 11**. The C++ code included in the window **1100** may be denoted as firmware code containing all information concerning the apparatus **901** in a machine-readable form. This firmware can be implemented in the apparatus for controlling the same.

[0098] It should be noted that the term "comprising" does not exclude other elements or steps and the "a" or "an" does not exclude a plurality. Also elements described in association with different embodiments may be combined. It should also be noted that reference signs in the claims shall not be construed as limiting the scope of the claims.

1. A data processing device, comprising

a first generation unit for generating, based on generic data defining a functional interface of an apparatus in general terms, a formalized description of the functional interface of the apparatus;

a second generation unit for generating, based on the formalized description, source code for realizing the functional interface of the apparatus.

2. The data processing device of claim 1, comprising at least one of the features:

the generic data defines the functional interface of the apparatus in a language which is understandable for a human being;

the formalized description defines the functional interface of the apparatus in a language which is interpretable, compilable or understandable by a machine.

3. The data processing device of claim 1, comprising at least one of the features:

a third generation unit for generating, based on the formalized description, a user manual documenting the functional interface of the apparatus;

A third generation unit for generating, based on the formalized description, a user manual documenting the functional interface of the apparatus, wherein the third generation unit is adapted to generate the user manual in a manner to be displayable and/or to be printable.

4. The data processing device of claim 1, comprising a receiving unit for receiving the generic data defining the functional interface of the apparatus.

5. The data processing device of claim 4, comprising at least one of the features:

the receiving unit is a user interface adapted to receive the generic data input by a user;

the receiving unit is adapted to receive generic data defining a complete functional interface of the apparatus;

the receiving unit is adapted to receive an entirety of all commands executable by the apparatus and/or an entirety of all actions performable by the apparatus.

6. The data processing device of claim 1, comprising at least one of the features:

the first generation unit is adapted to generate, based on the generic data, the formalized description of the functional interface of the apparatus in Extensible Markup Language;

the second generation unit is adapted to generate the source code in a form which is readable by a parser of the apparatus;

the second generation unit is adapted to generate the source code using a Stylesheet;

the second generation unit is adapted to generate the source code in C, C++, C#, Pascal, Basic, Fortran, or Java;

the data processing device is adapted to design the apparatus as at least one of a measurement device for performing a measurement in a coupled measurement environment, a sensor device, a test device for testing a device under test, a device for chemical, biological and/or pharmaceutical analysis, a fluid separation system adapted for separating compounds of a fluid, and a liquid chromatography device.

7. The data processing device of claim 1, comprising:

a storage unit for storing the generic data; and

a modification unit for modifying the generic data stored in the storage unit to modify the functional interface of the apparatus.

8. An apparatus for providing a functional interface, the apparatus comprising

a formalized description of the functional interface, wherein the formalized description is generated based on generic data defining the functional interface of the apparatus in general terms.

9. The apparatus of claim 8, comprising at least one of the features:

an interface via which the formalized description of the functional interface of the apparatus is providable to an entity coupled to the interface;

the apparatus is adapted as at least one of a measurement device for performing a measurement in a coupled measurement environment, a sensor device, a test device for testing a device under test, a device for chemical, biological and/or pharmaceutical analysis, a fluid separation system adapted for separating compounds of a fluid, and a liquid chromatography device.

10. A method of processing data, the method comprising

generating, based on generic data defining a functional interface of an apparatus in general terms, a formalized description of the functional interface of the apparatus;

generating, based on the formalized description, source code for realizing the functional interface of the apparatus.

11. A computer-readable medium, in which a computer program of processing data is stored, or a program element of processing data, which computer program or program element, when being executed by a processor, is adapted to control or carry out the method of

generating, based on generic data defining a functional interface of an apparatus in general terms, a formalized description of the functional interface of the apparatus;

generating, based on the formalized description, source code for realizing the functional interface of the apparatus.

* * * * *