[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 3/14
G06F 17/50

[12] 发明专利说明书

[21] ZL 专利号 97118520.4

[43] 授权公告日 2003 年 4 月 16 日

[11] 授权公告号 CN 1105964C

[22] 申请日 1997.9.12 [21] 申请号 97118520.4

[30] 优先权

[32] 1996. 9.13 [33] JP [31] 243137/1996

[71] 专利权人 株式会社山武

地址 日本东京

[72] 发明人 加藤正人

[56] 参考文献

CN1098214A 1995.02.01 G06F15/60

US5301301A 1994.04.05 G06F15/31,G06F15/46

US5490245A 1996.02.06 G06F3/14

WO9607957A1 1996.03.14 G05B19/042

审查员 范玉霞

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所

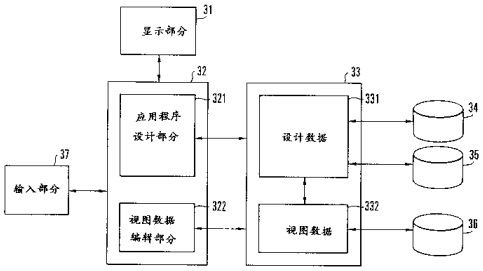
代理人 杨国旭

权利要求书 1 页 说明书 13 页 附图 8 页

[54] 发明名称 用于可视编程中显示功能对象的方法

[57] 摘要

一种用于显示可视编程中功能对象的方法，该显示方法的安排是：功能对象的图标(视图)是通过视图编辑部分生成/编辑的，且这些视图数据存储在内部存储器中，另一方面，包含在设计数据中的功能对象通过以上视图数据在可视编程时显示在计算机屏幕上。 这时，使用以上视图数据能够显示所有相同类型的功能对象，或者使用以上视图数据可以仅显示特定的功能对象。 此外，以上视图数据也用于在计算机屏幕上显示对象条中独特的功能对象。



ISSN1008-4274

1. 一种用于在可视编程中显示功能对象的方法，在一种可视编程环境下，其中对应于特定功能的功能对象标识为计算机屏幕上被显示的应用程序编辑页面中的图标，并且所述图标是通过链连接的以便定义所述功能对象之间的连接关系，该显示方法包括:

视图编辑过程，用于生成/编辑任一特定功能对象的图标;

存储过程，用于存储由所述视图编辑过程生成/编辑的图标；以及

视图显示过程，使用由所述存储过程所存储的所述图标用于在可视编程时在所述应用程序编辑页面上显示所述特定功能对象;

其特征在于，

所述视图显示过程能够移动用于连接链的所述功能对象的输入/输出端口到在所述视图编辑过程中所生成/编辑的所述图标上的任何位置。

2. 根据权利要求1所述方法，其特征在于，

在所述应用程序编辑页面上显示与所述特定功能对象相同类型的功能对象时，所述视图显示过程自动地使用由所述视图编辑过程所生成/编辑的图标。

3. 根据权利要求1所述方法，其特征在于，

与在所述视图编辑过程中所生成/编辑的图标相关的数据，能够独立于作为可视编程的主题的应用程序，而保存在所述应用程序编辑页面上。

4. 根据权利要求1所述方法，其特征在于，

与在所述视图编辑过程中所生成/编辑的图标相关的数据，能够作为可视编程的主题的应用程序的一部分而保存在所述应用程序编辑页面中。

用于可视编程中显示功能对象的方法

技术领域

本发明涉及工程工具等等中的人机界面领域，并特别涉及用于显示可视编程中的功能对象的方法，可视编程在计算机屏幕上图形化地定义作为功能组件操作的功能对象之间的数据流。

背景技术

可视编程是用来图形化地表示代表特定功能及其数据流的功能对象的技术，可视编程在计算机屏幕上使用框图以便从其连接关系对程序进行编码。当用于控制设计的配置工具当作使用这种可视编程技术的一例时，对应于控制系统中的输入/输出(I/O)点基本功能组件(以下称为功能块)，诸如PID组件等等的控制点准备作为功能对象。这些功能对象包括对应于相关功能的设置项，向其给出的设置数值，以及用来实现该功能的过程。这种功能块的的用户，即控制程序的编程员，在计算机屏幕上在应用程序编辑页面中显示适当的功能块，并定义块组件之间的关系，同时用称为链的直线画出框图。

图1是表示上述可视编程系统的硬件配置及可视编程中的计算机屏幕。该可视编程系统是在个人计算机(PS)或者工作站(WS)上构成的，包括作为显示装置的显示器11，由CPU、存储器之类构成的主机12，以及键盘13和鼠标14等输入系统。图1中在显示器11上所显示的有以图标指示各个可用的功能块列表的对象条15，以及用来设计和开发应用程序称为应用编辑页面的编辑窗口16。

在设计和开发应用程序时，用户使用鼠标从对象条15选择任一功能块，并在应用程序编辑页面16上显示该功能块。例如，使用鼠标14在对象条15中的“AI”15a点击、拖动、投放时，具有模拟输入功能的功能块17显示

在应用程序编辑页面16上。以这种方式在应用程序编辑页上显示功能块称为“粘贴”。

此外用户还可以调用被称为容器19的框将其显示(粘贴)到应用程序编辑页面16上。这里容器是指保持一个包含多个功能块并实现一个功能的子系统。这种容器是从一个文件(图1中未示出)读出到应用程序编辑页面16上的。容器19的内容，即子系统的多个功能块成分，还可分开显示在另一应用程序编辑页面(编辑窗口)上。并且它们还能够存储在一个文件中。这样，通过在一个容器中保持包含多个功能块的一个子系统用于处理，可以表示出要设计的对象分层结构。于是，用户可以不过问这一子系统的个别功能块成分，而能够把容器19作为一个功能组件对待。以下，这种容器和功能块将一并称为功能对象。

向应用程序编辑页面16粘贴了功能对象之后，用户使用鼠标14通过称为链18的直线连接功能块17的连接端20或者容器19的端口21，以便图形化地描述两个功能对象之间的数据流。以这种方式所设计和开发的应用程序是作为粘贴到应用程序编辑页面16上的功能对象及其连接关系存储的。此外，通过以下称为实现的过程，应用程序的源代码能够从功能对象及其连接关系生成。这就是说，只要通过在应用程序编辑页面16上在功能块或容器之间图形化地定义连接关系，用户就能够生成用于控制的程序的源代码，而不必关心包含在每一功能对象内的程序。

在可视编程中，粘贴到应用程序编辑页面的功能对象是以图标显示的。图8表示了用于功能对象的图标的一例。图8(a)是用于功能块的图标一例。在矩形图标22中，显示了功能块的内部参数。

这里所显示的内部参数是一特定应用程序中的表示相关功能块属性的逻辑名22a，表示属于相关功能块的功能的功能名22b，以及表示用来标识特定功能块的点名称的物理名(标签名)22c。此外，图标22带有用来连接表示与其它功能块等等的连接关系的链的连接端20。

另一方面，如图8(b)所示，一个保持有多个功能块成分子系统的容器是以一个矩形图标23显示的。图标23带有用来连接链的端口21。在容器内通

过这一端口21维持的功能块组与相关容器之外的功能块或者与其它容器交换数据。以下，粘贴到应用程序编辑页面上的这些功能对象的图标将称为视图。

在传统的可视编程中，诸如功能块或者容器之类的功能对象的一般使用的图形表示(指默认视图)大体限于基本上以矩形表示的简化的图形表示。其结果是，所有显示在显示器上的功能对象最终假定为外形类似，而负责设计和开发应用程序的用户不得不阅读这里所显示的逻辑名和功能名以辨别粘贴到应用程序编辑页面上的各个功能对象。

然而，当在显示器上功能块或者容器复杂时，或者当形成大的程序时，使用这种表示方式，则表示功能等的文字就要变得很小，于是可能导致不适当的功对象被错误地连接。

此外，在使用可视编程系统的对于控制设计配置工具的顾客或市场中，需要各种功能对象的表示形状。特别对于某些顾客或市场，有时要呈现广泛识别的视图。这种情形下，希望能够提供满足顾客需要的视图。然而基于默认视图传统上一般使用的表示方法不能满足这种需求。

发明内容

本发明的一个目的是要提供允许用户加工/编辑任一视图的表示方法，于是显示在计算机屏幕上的功能对象易于正确地可视地辨别。

为了达到上述目的，在一种可视编程环境下，其中对应于特定功能的功能对象指定为计算机屏幕上所显示的应用程序编辑页面中的一个图标，并且上述图标是通过链连接的以定义功能对象之间的连接关系，根据本发明用于在可视编程中功能对象的显示方法，在上述应用程序编辑页面上包括用于生成/编辑任一特定功能对象的图标的视图编辑过程，一个用于存储通过视图编辑过程所生成/编辑的图标的存储过程，以及使用由上述存储过程所存储的上述图标用于在可视编程时显示特定功能对象的显示过程。

这里，视图编辑过程是要通过粘贴诸如线段或折线、字符、位图等等图象表示出用来在由用户所指定的功能对象中生成和编辑一个视图的所

有的过程。于是，可视编程系统可以装有专用的画图工具，但是视图的生成/编辑可能受到使用一般的诸如CAD工具这类画图工具的影响。诸如这样生成/编辑的图象的视图数据存储在计算机的一个存储器中，例如内部存储器或者文件中。并基于这些存储的视图数据，显示过程在应用程序编辑页面上使用事先生成/编辑的视图代替诸如矩形之类默认的视图来显示上述功能对象。这样，功能对象能够在应用程序编辑页面上由任何视图设定，这就便于功能对象的可视辨别。

结果，防止了功能对象的错误连接等等，这种错误在通过传统的默认视图一般使用的图形表示中是易于出现的，其结果能够改进可视编程的生产率及应用程序的可靠性。此外，由于用户能够生成/编辑用于每一功能对象的任何视图，从而本发明能够适合满足可视编程系统或应用程序的顾客或市场需求的表示形状。

本发明的显示方法的特征在于，在以上视图显示过程能够把以上用于连接链的以上功能对象的输入/输出端口移动到以上视图编辑过程中所生成/编辑的以上图标上的任何位置。这样，能够设置输入/输出端口的位置使之与视图一致，并且能够使得通过链与其它功能对象的连接关系在视觉上更为可视化。

根据本发明在可视编程中显示功能对象的一种方法，还能够通过使用在视图编辑过程所生成/编辑的视图而只显示规定的特别功能对象，但是根据本发明用于显示可视编程中功能对象的第五个方法，不仅显示在生成/编辑视图中所规定的功能对象，而且自动地显示使用该视图粘贴到应用程序编辑页面上的同类功能对象。这样，不论已经粘贴到一个应用程序编辑页面上还是将来要粘贴，使用在视图编辑过程中所生成/编辑的视图，所有功能对象连同具有特定数值的内部参数都能够被显示。于是用户无需对多个功能对象进行同一种相同的画图。

这可以通过把表示特定功能对象类型的内部参数的数值与形成视图的图象数据相关联以便组成视图数据的一部分而实现。具体来说，通过使得功能名加入到指示功能对象类型的内部参数的数值之中，认为视图与特定

的功能名相关。例如，当规定具有功能名“模拟输入”的特定功能块生成/编辑一个视图时，通过把指示视图的图象数据与功能名“模拟输入”相关联而形成一致的视图数据，其它已经粘贴到应用程序编辑页面上并具有功能名“模拟输入”的功能块，以及具有功能名“模拟输入”将来要粘贴的功能对象能够借助于这种相关视图显示。

此外，在根据本发明方法的可视编程中用于显示功能对象的一个方法中，特别是其存储过程能存储在视图编辑过程中生成/编辑的视图相关联的信息，此即，与图象数据(以下称为视图数据)相关联并进入内部存储器的信息，但是它还是内部参数等，被认为在诸如硬盘或软盘等外部存储器存储/保留这些视图数据。这里，第六发明的特征在于，与应用程序无关而作为在以上应用程序编辑页面中的可视编程的一个题目，特别是以上存储过程能够保留与在以上视图编辑过程中所生成/编辑的视图相关联的数据。这样，曾经在一定的的应用程序中生成/编辑的视图能够在另一应用程序中被再次使用。于是，用户无需对每一应用程序重新生成一视图。

进而，根据本发明的第四显示方法的特征在于，作为在以上应用程序编辑页面中可视编程的一个题目，以上存储过程能够保留在以上视图编辑过程作为应用程序一部分所生成/编辑的图标相关联的数据。通过这样保留诸如图象数据之类的视图信息作为应用程序的一部分，促进了在视图编辑过程中所生成/编辑的视图与其应用程序的一致性。于是，对于诸如具有与一定的应用程序紧密连接的功能的容器这样的功能对象，认为在特定的应用程序最适合的视图能够与相关应用程序一致。特别地，当开发特定业务类型的应用程序时，使用在相关业务领域被广泛辨认的视图用作具有独特功能的容器将能够满足顾客或市场的需求。

附图说明

图1是一示意图，表示可视编程系统的硬件配置及可视编程中的显示器;

图2是一框图，表示根据本发明的实施例的可视编程系统的配置;

图3是一流程图，表示生成/编辑功能对象的视图的过程；

图4是一示意图，表示第一实施例中视图编辑过程及存储过程；

图5是一示意图，表示第一实施例中功能对象的视图显示过程；

图6是一示意图，表示第二实施例中视图编辑过程及存储过程；

图7是一示意图，表示第二实施例中功能对象的视图显示过程；以及

图8是一图示，表示粘贴到编辑页面上功能对象的传统图标的一例。

具体实施方式

以下参照图示详细说明本发明。

实施例1是具有两种方式的可视编程系统，即进行可视编程的编程方式和进行功能对象的视图生成和编辑的视图编辑方式。

如图1所示，这种可视编程系统的硬件配置包括显示器11，包括CPU、存储器等及诸如键盘13和鼠标14的输入装置的主机12。其中，虽然在图1中没有示出，其中在计算机硬件上存有用于操作根据本发明的可视编程系统的光磁盘之类的存储介质可连接到主机12。

如图2中所示，这一可视编程系统一般包括对应于图1的显示器11的显示部分31，用于控制应用程序的设计和开发及视图的生成和编辑的控制部分32，用于存储应用程序或视图数据的内部存储器33，诸如功能对象数据库34之类的外部存储器，其中存储事先定义的功能块、容器等等，用于存储/保留应用程序的应用程序数据文件35，用于存储/保留存储在内部存储器33中的视图数据配置视图文件36，以及包括鼠标14等的输入部分。

这里，控制部分32包括CPU和主存储器，其功能可分为两部分：应用程序设计部分321及视图编辑部分322。应用程序设计部分321涉及到编程方式并一般控制可视编程，诸如把功能对象粘贴到显示在显示部分31的应用程序编辑页面(以下称为编辑页面)16上，建立这些功能对象之间的链接，并进一步处理称为具体化过程，用于从其输入/输出关系被定义的功能对象生成源代码。另一方面，视图编辑部分322涉及视图编辑方式并一般控制视图编辑过程，诸如借助于画图工具或内部参数规范生成/编辑视图。

当在这种可视编程系统中实现设计/开发应用程序时，诸如内部存储器33或外部存储器等硬件资源及数据之间的关系是如下运用的。即在编程方式中粘贴诸如功能块和容器等功能对象到编辑页面16上时，从功能对象数据库34读出相关的功能对象到内部存储器33，作为应用程序设计数据331。结果，这些功能对象的每一个能够在其应用程序中被赋予唯一的物理名。对于在内部存储器33上形成应用程序设计数据331的功能对象，用户定义连接关系。表示这一连接关系的链也变为应用程序设计数据331的一部分。这样组成的应用程序设计数据331可被保留在应用程序数据文件35中。此外，还能够从应用程序数据文件35向内部存储器33加载曾经存储的应用程序设计数据，并继续进行可视编程。

另一方面，在视图编辑方式中，功能对象视图的生成/编辑是在视图数据编辑部分322下进行的。其中，在稍后所述的视图生成/编辑过程中所生成/编辑的视图数据332存储在内部存储器33中。

图3表示功能对象视图的生成/编辑中的过程。在一个视图生成/编辑之前，以编程方式如同传统可视编程那样用户把功能对象粘贴到编辑页面16上(步骤301)，并然后规定这些粘贴的功能对象之一作为视图生成的对象(步骤302)。这时，待指定的功能对象与应用程序的设计/开发或者为从第一个生成/编辑视图而粘贴到编辑页面的进程无关。在规定了任何功能对象之后，通过从方式选择菜单(未示出)选择“视图编辑”画图工具自动地起动，并且设定视图编辑方式。

用户使用其画图工具生成/编辑以上规定的功能对象的视图(步骤303)。此外，通过起动内部参数设置工具，用户能够设置在新生成/编辑的视图及其显示位置中要显示的内部参数。在完成视图编辑过程时这一视图数据自动地存储/保留在一个文件中(步骤304)。303和304这些步骤对应于视图编辑方式。并在返回编程方式时，以上规定的功能对象借助于在视图编辑方式中所生成/编辑的视图来显示(步骤305)。

下面参照图2和4，将详述以上所述的这种过程。

在粘贴到编辑页面16的功能对象之中，图4示出规定功能块17具有功能

名“模拟输入”的情形。在方式选择菜单中选择视图编辑方式时(图2和4中未示出)，控制部分32的操作经过从应用程序设计部分321到视图编辑部分322，从而进入视图编辑方式。

当操作从程序方式进入视图编辑方式时，视图编辑部分322自动地起动画图工具41。使用这一画图工具41，用户画出诸如矩形、椭圆折线或字符等画面，或者制成通过扫描仪等形成位图数据的图象，于是用户能够生成并编辑画面数据。这样生成和编辑的画面数据作为对于先前所规定的功能对象的新的视图存储在内部存储器33中。

此外，通过起动内部参数设置工具42，用户能够设置在新生成/编辑的视图及其显示位置中要显示的内部参数。即，当规定的功能对象为具有多个内部参数的功能块时，要考虑的是与在可视编程时图8中所显示的默认视图一同显示逻辑名、功能名及物理名。这样，要显示的内部参数是这样安排的，使得能够由用户使用内部参数设置工具来确定。从而，与在可视编程中新的视图一同显示内部参数数值成为可能。例如，图4表示，在内部参数设置工具42中确定“逻辑名”42a为要显示的参数。此外，画图工具41示出内部参数的显示位置41a确定为画面数据的一部分。

与上述内部参数的显示相关的信息与作为视图数据332的一部分的画面数据一同存储在内部存储器32中。图4表示画面数据332a和显示内部参数“逻辑名”332c在内部存储器33上作为视图数据332的一部分。顺便来说，这一内部参数的规定是可选择的，用户可以借助于内部参数设置工具不总是实行这一规定。

进而，如图4中所示，视图数据332包括规定的功能对象17的功能名“模拟输入”作为其一部分(332b)。在可视编程方式中，通过使用规定的功能对象的功能名作为如上述的关键字，借助于由用户生成/编辑的视图，不仅在向视图编辑方式转变中所规定的功能对象(例如图4的功能块17)，而且同类的功能对象也被显示。

如果参照图4从视图数据生成/编辑过程的观点描述，则成为以下的情形。即在规定了功能对象(功能块)17之后当操作进入视图编辑方式以便生

成/编辑一个新的视图时，视图编辑部分322自动地在内部存储器33的功能名332b中存储其功能块17作为关键字的功能名“模拟输入”。这时，如果规定的功能对象为一容器，则视图数据332的“容器”功能名332b存储在内部存储器33的功能名332b。以下将说明使用存储在内部存储器33中的功能名332b的程序方式中功能对象的显示过程。

顺便而言，使用功能名作为关键字只是上述功能对象类型描述的一个例子。于是自不必说，通过选择逻辑名(例如“A101”)作为关键字代替功能名，只要是具有相同属性的功能对象也能够借助于先前生成/编辑的视图来显示。

上述所生成/编辑的视图数据332包括作为规定的功能对象视图的画面数据332a和指示功能对象类型的功能名332b，并包含与选择显示的内部参数相关的信息332c。并且，在视图编辑方式完成时内部存储器33上的视图数据332存放在配置视图文件36中。

当视图的生成/编辑和存储如此完成时，操作自动地从视图编辑方式切换到编程方式，而用户使用刚才生成/编辑的新的视图能够实现可视编程。这时，通过指定相同的功能块再次进入视图编辑方式，能够修改曾经生成的视图。此外，在这一编程方式中，还能够指定不同的功能块并再次进入视图编辑方式。这种情形下，用户按照类似的过程能够重新生成/编辑其它视图数据。

在这一实施例中，只要应用程序存储在应用程序数据文件35中，视图数据就作为独立的文件保存在配置视图文件36中。而且，从这一应用程序之后生成/编辑的新的视图数据保存在对于应用程序共用的一个配置视图文件36中，使得该数据被添加到已经生成/编辑的视图数据上。这样完成的配置视图文件36包含所有在这一应用程序中要使用的视图数据，但是成为独立于应用程序自身的文件。于是，保留在这一配置视图文件36中的视图数据能够对不同的应用程序使用。

以下参照图5将说明实施例1中的显示过程。

由上述视图编辑过程所生成/编辑的视图数据332存储在内部存储器33

上。这些视图数据332包括画面数据332a，作为关键字的功能对象的功能名332b，以及要显示的内部参数332c。除了在视图编辑方式中刚才已经生成/编辑那些数据之外，视图数据332可以是那些从配置视图文件36读入的数据。在编程方式下，当上述视图数据332出现在内部存储器33上时，图2中所示的应用程序设计部分321上在编辑页面16上使用画面数据332a显示对应于来自功能对象的视图数据332的那些数据。

在本实施例中，使用内部参数的数值作为关键字，具有共同性质的多个功能对象能够基于一个视图数据集被显示。这种视图这里称为配置视图。具体而言如图5所示，由于视图数据332包含功能名“模拟输入”332b作为关键字，粘贴到编辑页面16上的功能对象中具有相同功能名的那些视图数据，使用画面数据332a独立于其粘贴时间而被显示(51,52)。此外，当以相同功能名(“模拟输入”)作为关键字的功能对象粘贴到编辑页面16上时，它们使用画面数据332a被显示。

顺便来说，作为关键字，实施例1使用了上述的功能名，但是也可以使用其它内部参数代替该名。就是说，使用逻辑名，例如“A101”，作为视图编辑过程中的关键字，也可以生成配置视图数据。在这种情形下，以图5作为编程方式中的一个例子，使用画面数据332a显示具有逻辑名“A101”功能对象51，而具有不同逻辑名的功能对象52是借助于图8(a)所示的默认视图显示的。

顺便来说，在这种情形下，用作在对象条15中的图标是默认的。

另一方面，在配置视图51和52中，在先前的视图编辑过程中使用内部参数设置工具42所指定的内部参数数值(图4)，即功能对象的逻辑名“A101”和“A102”，显示在指定的位置(51a和52a)。此外，借助于由用户生成的视图所表示的功能对象51和52的输入/输出端口51b和52b，能够在新视图周围的任何位置显示。

这样，借助于由用户所生成/编辑的视图在编辑页面16上显示特定类型的功能对象，也可以使用在生成时适合该应用程序的视图，于是在可视编程中功能对象的可视识别变得容易。

此外，对于对应于对象条15中的功能对象“模拟输入”的视图53，使用视图数据332的画面数据332a。于是，在从对象条15选择功能对象时易于进行可视修改。

由于这种配置视图数据独立于应用程序存储/保留在配置视图文件中，故曾经生成的配置视图也能够用于其它的应用程序。例如，在以上说明中，视图数据332的一个集要存储到内部存储器33中，但是自然，两个或者更多的视图数据集也可以存储。

以下将参照图6和7说明本发明的实施例2。

实施例2的特征在于，除了实施例1中所述的配置视图之外，可以使用称为定制的视图。这里，定制视图的意思是指只对于特定的功能对象本身使用的视图。在专属特定的功能对象本身的一个视图中，这与通过关键字能够用于性质共同的多个功能对象的配置视图不同。这种定制视图特别的意义在于使用具有对于其应用程序特定功能的容器的情形。

定制视图的生成/编辑基本上是根据图3所示的过程进行的。即首先考虑指定编程方式中粘贴到编辑页面16的功能对象(例如图6的容器A19)以便进入视图编辑方式，这对于实施例1的操作是共同的。然而对于实施例2，在进入视图编辑方式时，用户必须选择两个视图编辑方式之一，即定制视图编辑方式或配置视图编辑方式。这里将假定选择定制视图编辑方式进行说明。

这里当选择定制视图编辑方式时，如同实施例那样，画图工具41被自动地起动。类似于实施例1中所述的那些操作，用户可使用画图工具41画出诸如矩形，椭圆和折线或字符，或者可处理由扫描仪等所提取的图象作为位图数据，以便这样生成和编辑画面数据。由于定制视图变为对事先特定的功能对象所特有的一个视图，故在定制视图编辑方式中存储在内部存储器33中的定制视图数据61只满足如图6所示画面数据。

这时，这些定制视图数据61被看作是存储在内部存储器33的应用程序设计数据62中特定功能对象的部分。这是通过具有指针的功能对象指示定制视图数据到其对象数据而实现的。并且，在保存应用程序设计数据62到

应用程序数据文件35中时，这些定制视图数据61作为这一特定应用程序的部分被保存。它们在这一点不同于配置视图文件36中独立于应用程序(图2和4)而存储/保留的配置视图。

同时，还能够选择配置视图编辑方式而不是定制视图编辑方式。即使是这种情形下，画图工具41也是自动地起动，而接下去的配置视图编辑过程与实施例1相同。

以下将参照图7说明根据本实施例的定制视图显示过程。

当定制视图如上被生成/编辑时，或者当包含定制视图数据的应用程序从应用程序文件35加载时，除了应用程序设计数据62之外定制视图数据61存储到内部存储器33。当粘贴到编辑页面16的功能对象具有定制视图数据时，在可视编程系统中的应用程序设计部分321(图2)自动地使用包含在内部存储器33的定制视图数据(画面数据)61，以便借助于定制视图71显示这一对象。这里，通过指定借助于定制视图71所显示的功能对象以便再次转移到定制视图编辑方式，也能够修改或者重新制作现有的定制视图71。

其中，在实施例2中，由于可使用两个类型的视图，即定制视图和配置视图，故对于显示过程提供了以下2个类型的视图显示方式，用户可从视图显示菜单中任选它们之一。

首先，第一视图显示方式是指定制视图显示方式，而如果通过上述视图编辑过程定义了定制视图，则使用该显示方式显示功能对象。否则，如果定义了配置视图，则以配置视图显示功能对象，此外则以默认视图显示。

另一方面，第二视图显示方式是指配置视图显示方式。这里不论是否存在定制视图，定义了配置视图的视图的显示是借助于它们自身的配置视图进行的，或者其它视图的显示是借助于默认视图的。由此，通过选择配置视图显示方式，用户能够禁止按定制视图显示。

如上所述，在定制视图编辑方式中，用户能够生成/编辑功能对象特有的视图，特别是对于具有特定功能的容器所特有的视图，并适合于在设计中的应用程序的应用。此外，在定制视图编辑方式中，输入/输出端口72可移动到视图的任何位置。在定制视图显示方式中，由于借助于它(它们)

特别的视图在视图编辑页面上只有特定的功能对象才显示，故用户能够轻易地从其它功能对象中区分出它(它们)，并在可视编程时同时可视地区分它(它们)的功能。

顺便来说，在实施例1和2中，分别描述了对于功能块使用配置视图及对于容器使用定制视图的例子，但是自不必说，反过来也可以对于功能块使用定制视图及对于容器使用配置视图。

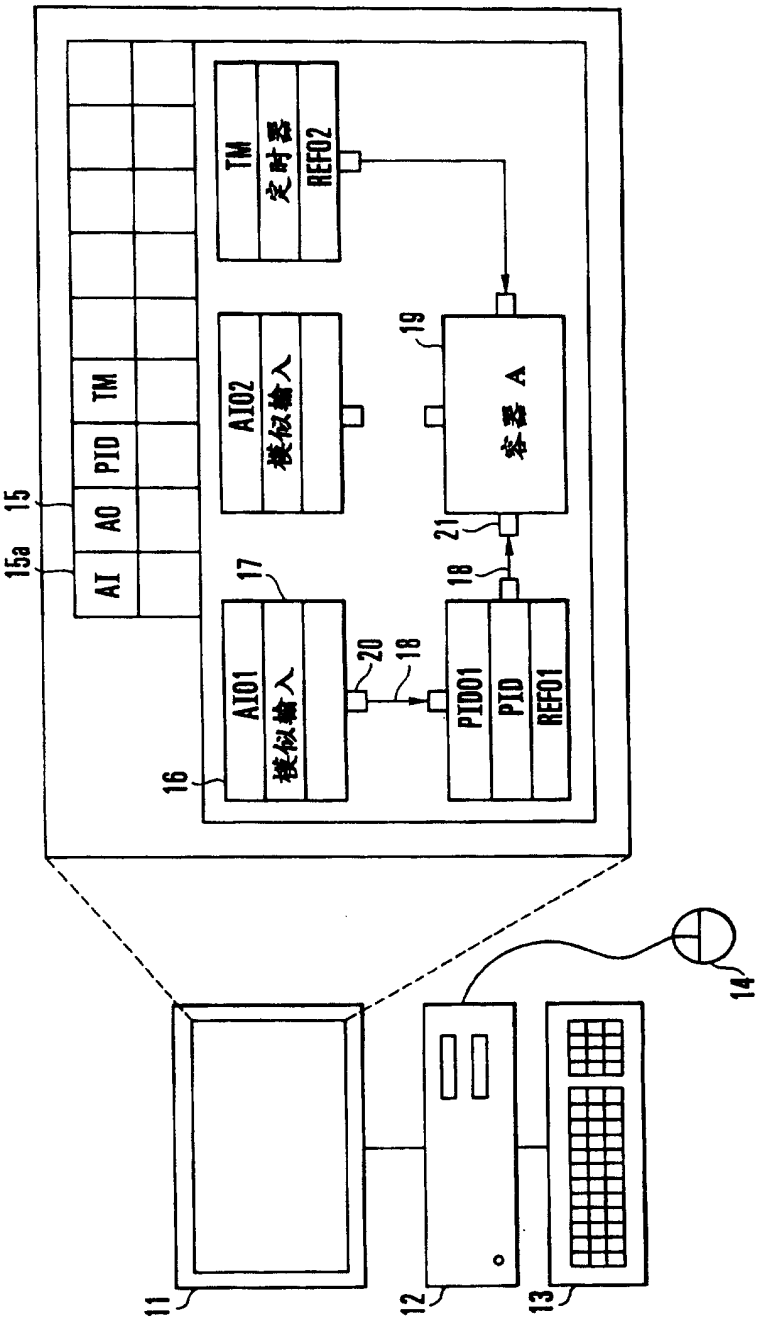

TM
定时器
REFO2
AIO2
模拟输入
容器 A
TM
PID
AO
AI
15
15a
19
21
18
17
AIO1
模拟输入
20
18
PIDO1
PID
REFO1
16
11
12
13
14

图 1

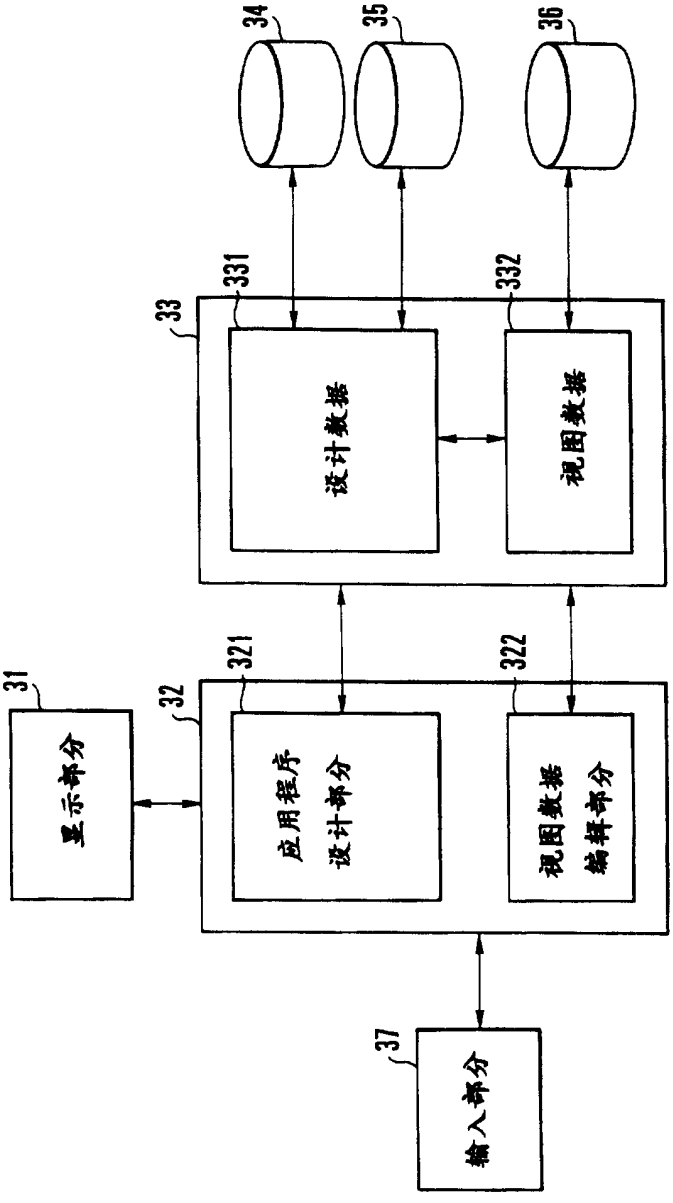

34
35
36
331
332
33
设计数据
视图数据
321
322
31
32
显示部分
应用程序
设计部分
视图数据
编辑部分
37
输入部分

图 2

图 3

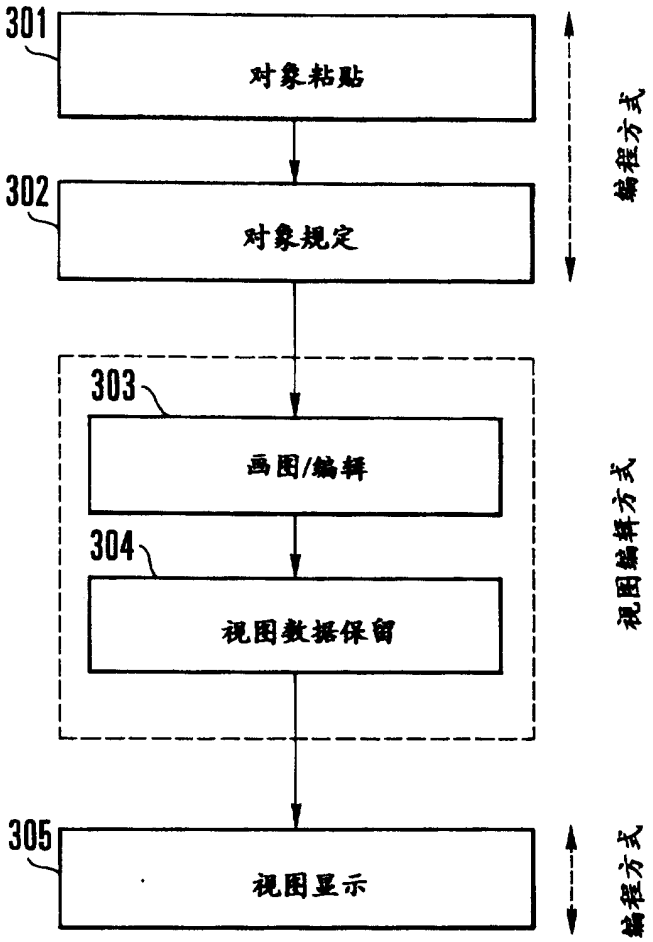

301
对象粘贴
302
对象规定
编程方式
303
画图/编辑
304
视图数据保留
视图编辑方式
305
视图显示
编程方式

图 4

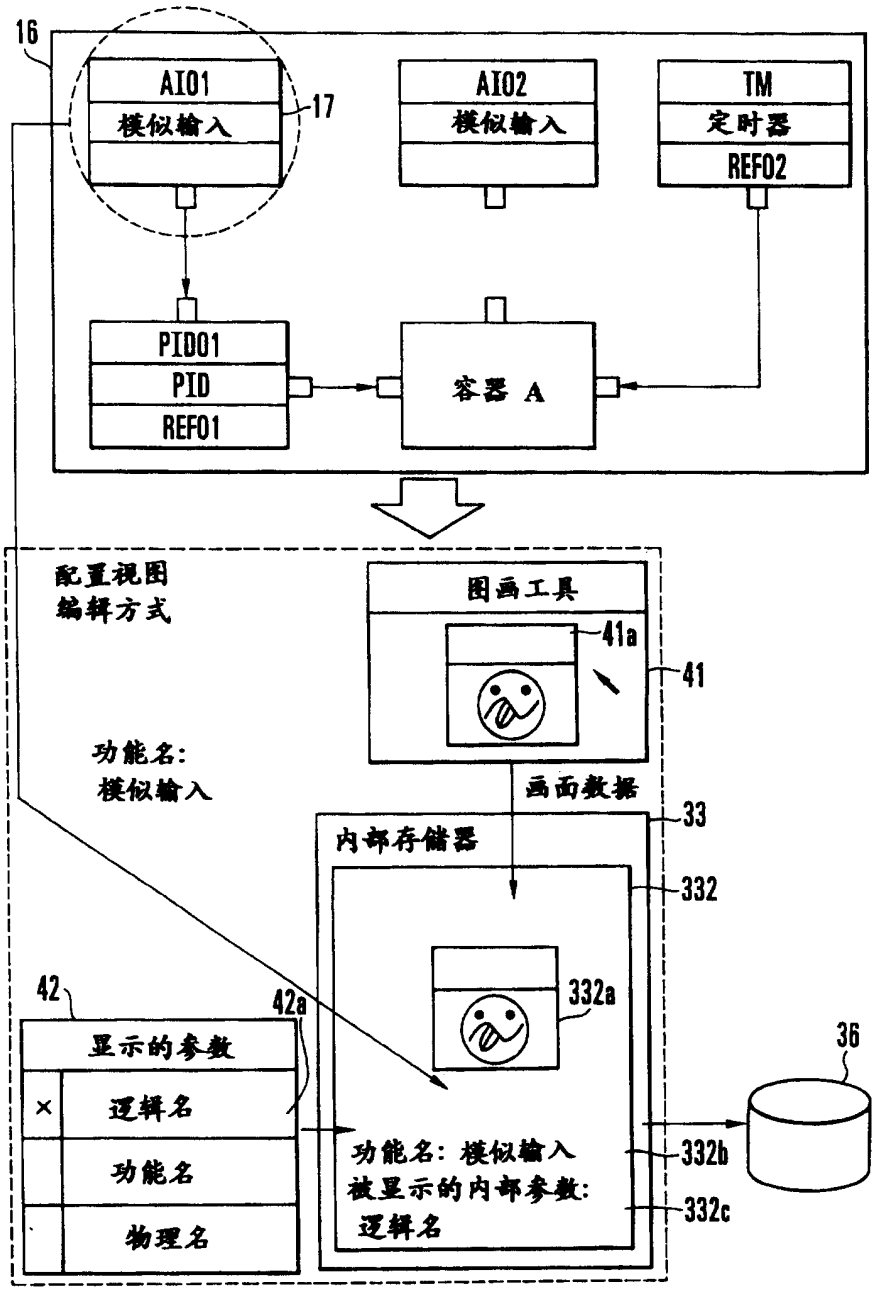

16
AIO1
模似输入
17
AIO2
模似输入
TM
定时器
REFO2
PIDO1
PID
REFO1
容器 A
配置视图
编辑方式
图画工具
41a
41
功能名:
模似输入
画面数据
33
内部存储器
332
332a
42
42a
显示的参数
×
逻辑名
功能名
物理名
36
功能名: 模似输入
被显示的内部参数:
逻辑名
332b
332c

图 5

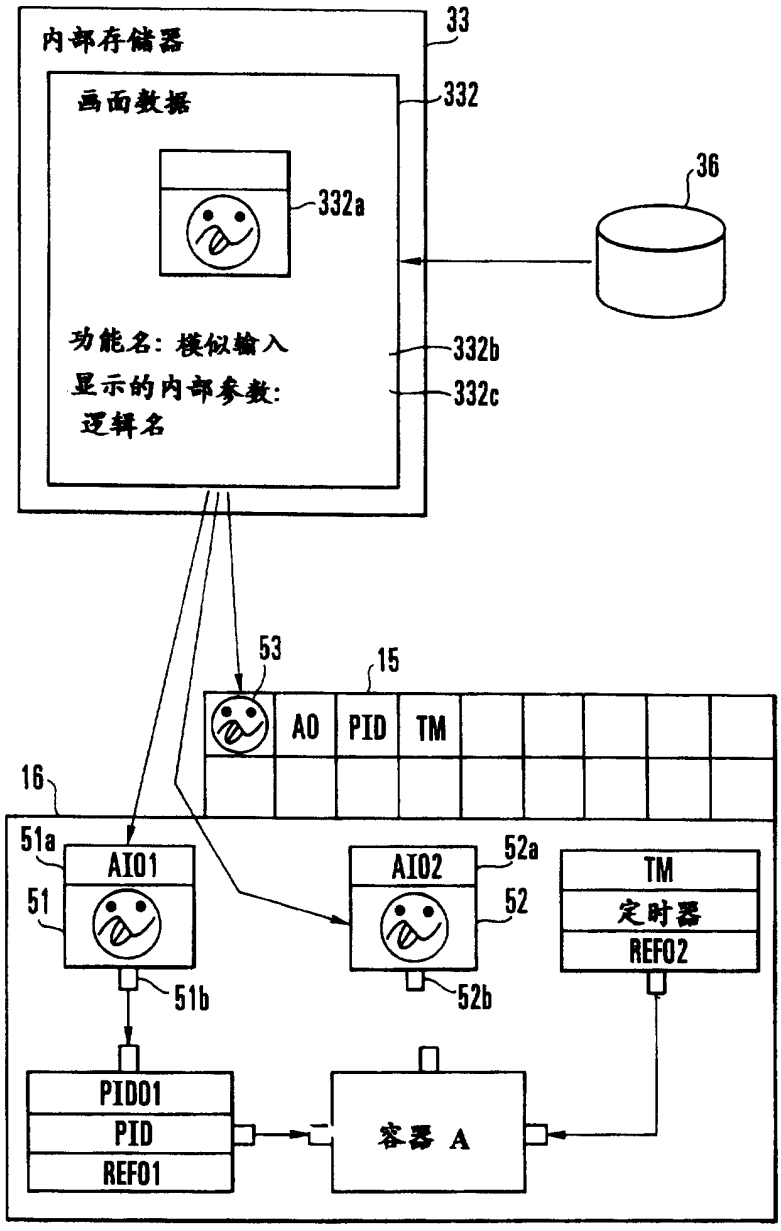

内部存储器
33
画面数据
332
332a
36
功能名: 模似输入
显示的内部参数:
逻辑名
332b
332c
53
15
AO
PID
TM
16
51a
AIO1
51
51b
AIO2
52a
52
52b
TM
定时器
REFO2
PIDO1
PID
REFO1
容器 A

图 6

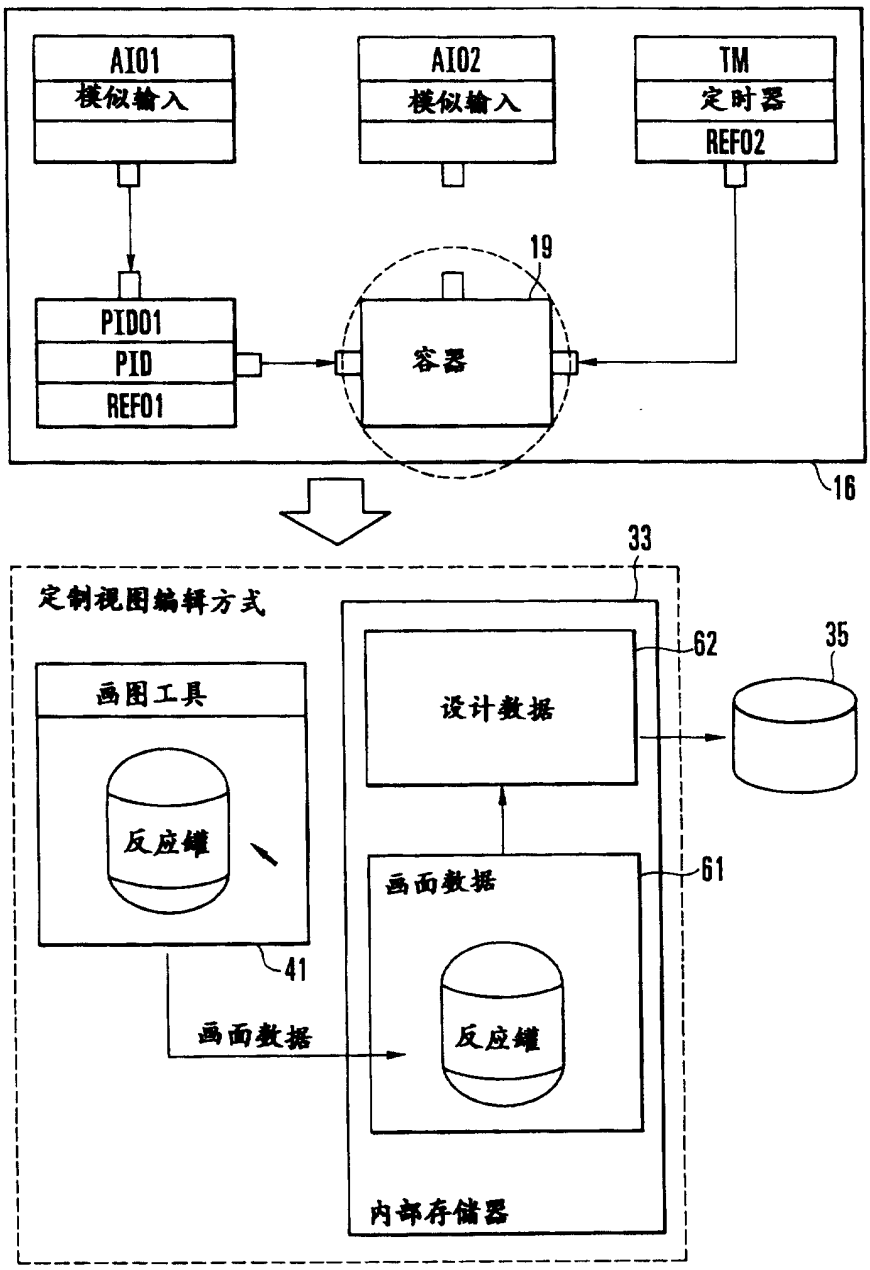
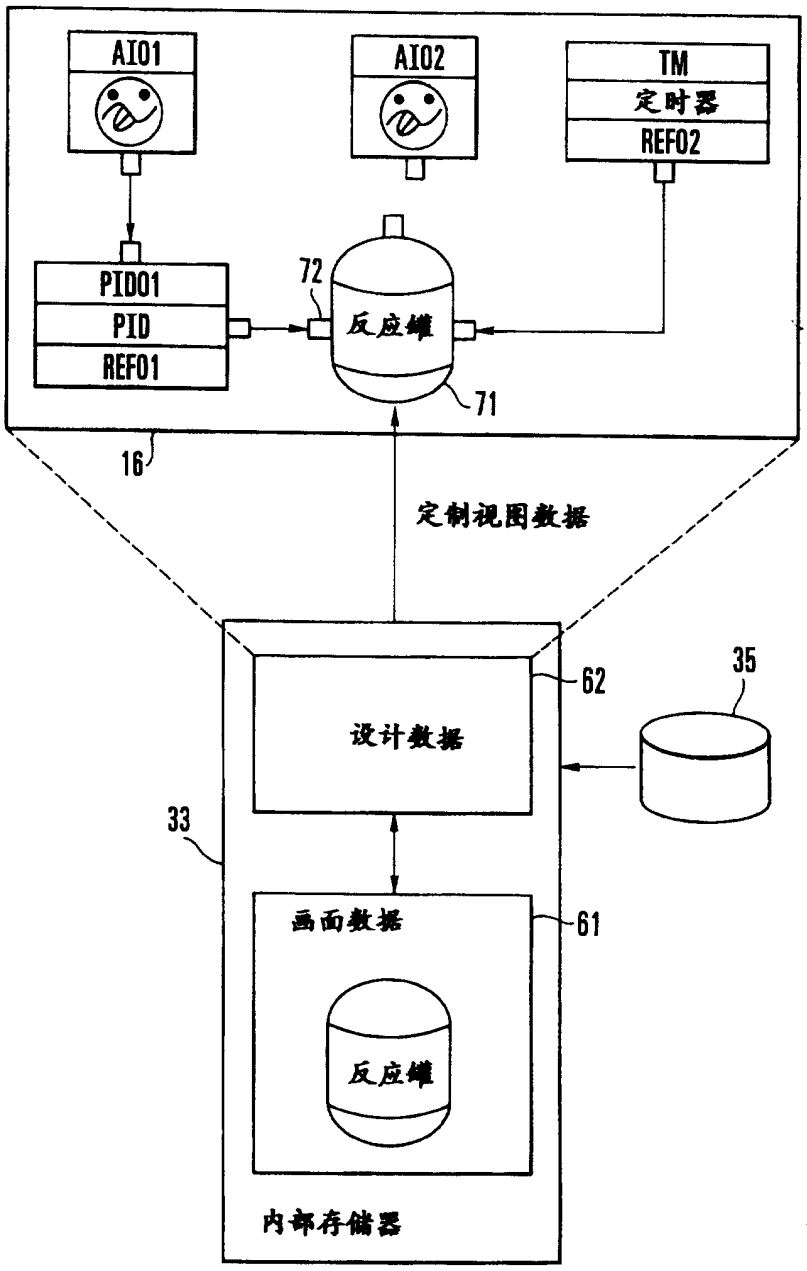

AI01
模似输入
AI02
模似输入
TM
定时器
REF02
19
PID01
PID
REF01
容器
16
33
定制视图编辑方式
画图工具
反应罐
41
画面数据
62
35
设计数据
61
画面数据
反应罐
内部存储器

图 7


AIO1
AIO2
TM
定时器
REFO2
PIDO1
PID
REFO1
72
反应罐
71
16
定制视图数据
62
35
设计数据
33
画面数据
61
反应罐
内部存储器

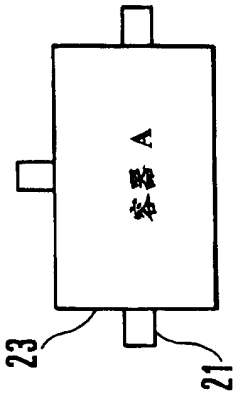


图 8B

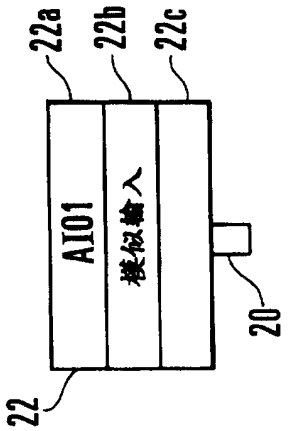


图 8A