[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 9/06

[12]发明专利申请公开说明书

[21]申请号 99120898.6

[43]公开日 2000年4月19日

[11]公开号 CN 1250907A

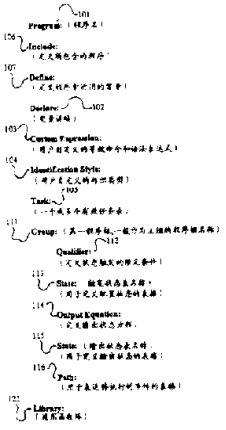
[22]申请日 1999.10.8 [21]申请号 99120898.6
[30]优先权
[32]1998.10.9 [33]US [31]09/169462
[71]申请人 林亚夫
地址 美国加利福尼亚州
[72]发明人 林亚夫
[74]专利代理机构 中国专利代理(香港)有限公司
代理人 王 勇 王忠忠

权利要求书9页 说明书17页 附图页数11页

[54]发明名称 表格格式编程

[57]摘要

适用于利用多种语言并通过因特网和网络环境下载的表格格式编程方法提供了一个用于定义定制表达式的可选表格(103,200),一个用于规定多任务应用的可选表格(105,260),一个用于规定标号的打印字格式的可选表格(104,230)以及一个与路径表(116)交互作用的配置状态表(113)。一个可选限定表(112)定义了一种输入配置状态的限定条件。硬件实施包括一个用于处理表格格式编程过程并释放主处理器用于其它应用的协处理器。在编程过程中,程序员提供了用户定制标号来描述所需的功能和操作。这些标号还可用于其它程序组中开发。



ISSN1008-4274

权　利　要　求　书

1. 一种由计算设备执行的计算机程序，包括具有 x 种配置状态的第一表格，所述配置状态中的至少一种定义了一个或多个限定条件；

规定了 y 个路径的第二表格，当满足所述第一表格中列出的一个限定条件时，执行所述路径中的至少一个；所述程序还包括下述表格中的至少一个：

（1）一个规定了用于表示一种编程语言的相应预定义指令的用户定制表达式的表格；

（2）一个定义了关键字的可取字格式的表格；

（3）一个用 m 种任务状态定义 n 个任务的有效性的表格，所述任务状态中的至少一种规定了 k 个有效的选定任务；

（4）一个用 p 种任务状态定义 q 个任务的表格，所述任务状态中的至少一个规定了将被服务的有效任务的优先权；

（5）一个定义了 x 个限定表达式的表格，每个限定表达式表示一种配置状态的一个限定条件；以及

（6）一个定义了 y 个输出表达式的表格，每个输出表达式表示一种配置状态或所述第二表格中列出的一个路径元素的一个输出条件。

2. 如权利要求 1 所述的计算机程序，其特征在于用存储在用于存储数字数据的存储装置中数据来表示。

3. 如权利要求 2 所述的计算机程序，其特征在于该程序被嵌入用于销售商品的计算设备中。

4. 一种用于对响应一个或多个限定条件执行一个或多个路径的计算设备进行编程的编程方法；所述编程方法最少包括步骤：

（1）规定 x 种配置状态；

（2）规定一个或多个限定条件到至少一种配置状态，

（3）规定 y 个由所述计算设备执行的路径；

（4）分配 z 个标号来表示步骤（3）中的一个或多个路径元素，其中 z 是一个大于等于 1 的整数；

（5）对于步骤（2）中的每个限定条件，当满足一个特定限定条件时，就规定步骤（3）中的一个将被计算设备执行的路径；

（6）将配置状态中的至少一种规定为有效配置状态；

（7）为步骤（4）中的每个分配标号，定义一个用所述标号表示的可执行程序。

5. 如权利要求 4 所述的编程方法，其特征在于步骤（4）中的标号最初不能被所述计算设备执行并且另一步骤（7）被配置成能够被所述计算设备执行所述标号。

6. 如权利要求 5 所述的编程方法，其特征在于还包括识别不能被所述计算设备执行的所述标号以用于构成步骤（7）的程序的步骤。

7. 如权利要求 4 所述的编程方法，其特征在于用包括表格格式编程语言在内的任何可用编程语言来构成步骤（7）的程序。

8. 如权利要求 7 所述的编程方法，还包括定义用于在步骤（1）到（6）所表示的程序和标号所表示的可执行程序之间传递参数的装置的步骤。

9. 如权利要求 4 所述的编程方法，还包括规定指向于所述 x 种配置状态中的至少一种的一个或多个输出条件的步骤。

10. 如权利要求 4 所述的编程方法，其特征在于在一个路径或一种配置状态中规定了至少一个输出条件。

11. 如权利要求 4 所述的编程方法，还包括识别所述计算设备管理其状态和路径所需的资源的步骤。

12. 如权利要求 4 所述的编程方法，还包括指示所述计算设备用于构成步骤（7）的程序的资源的步骤。

13. 如权利要求 4 所述的编程方法，其特征在于所述计算设备由两个或多个处理器构成；通过第一处理器提供管理特定状态和路径的资源并利用第二处理器资源运行步骤（7）的程序的至少一部分。

14. 如权利要求 4 所述的编程方法，其特征在于所述计算设备被定义为第一计算设备；所述编程方法还包括通过通信连接从与所述第一计算设备距离遥远的第二计算设备接收表示上述步骤的数字数据的步骤。

15. 如权利要求 4 所述的编程方法，其特征在于至少部分所述步骤按照表格格式进行组织。

16. 如权利要求 15 所述的编程方法，还包括将步骤（1）和（2）的配置状态分成一个或多个表格的步骤。

17. 如权利要求 15 所述的编程方法，其特征在于配置状态或路径

不必按照顺序彼此依次列出。

18. 如权利要求 4 所述的编程方法，还包括将配置状态和路径说明转换成存在所述计算设备中的数字数据以便用于执行的步骤。

19. 如权利要求 4 所述的编程方法，还包括用于转换利用不同格式的第二编程语言的所述步骤的至少部分说明的步骤。

20. 如权利要求 4 所述的编程方法，还包括用于识别由所述步骤构成的程序位置的步骤。

21. 如权利要求 7 所述的编程方法，其特征在于步骤（7）的程序是一个位于由权利要求 1 中的步骤构成的程序之外的程序。

22. 如权利要求 21 所述的编程方法，还包括用于识别所述外部程序位置的步骤。

23. 如权利要求 4 所述的编程方法，还包括用于形成一个独立的用于规定由步骤（1）到（7）构成的程序是有效还是无效的表格的步骤。

24. 如权利要求 4 所述的编程方法，还包括用于分配一个表示一种配置状态的标号的步骤。

25. 如权利要求 4 所述的编程方法，还包括用于分配一个表示一个路径的标号的步骤。

26. 一种用于对执行多个程序的计算设备进行编程的编程方法，包括步骤：

（1）规定 n 个可由所述计算设备执行的程序；

（2）定义 m 种任务状态，每种任务状态规定 k 个有效的选定程序，其中 k 是一个大于等于 0 的整数；以及

（3）规定步骤（2）中的一种任务状态为有效任务状态。

27. 如权利要求 26 所述的编程方法，其特征在于至少一个程序是一个表格格式程序，包括：

x 种程序配置状态，所述配置状态中的至少一种定义了一个或多个限定条件；

y 个由所述计算设备执行的路径；以及

当满足一个特定的限定条件时，所述计算设备执行所述路径中的一个。

28. 如权利要求 26 所述的编程方法，其特征在于步骤（1）的程序是用不同的语言编写而成。

29. 如权利要求 26 所述的编程方法，还包括用于提供一个关键字以表示构成步骤（1）到（3）的一个表格格式程序组的步骤。

30. 如权利要求 26 所述的编程方法，其特征在于所述任务状态不必按照顺序彼此依次列出。

31. 如权利要求 26 所述的编程方法，其特征在于步骤（1）到（3）定义了第一任务表，所述编程方法还包括定义一个不同的第二任务表的步骤。

32. 一种用于对一个远程计算设备编程的方法，包括步骤：

（1）规定 x 种配置状态，所述配置状态中的至少一种定义一个或多个限定条件；

（2）规定 y 个由所述计算设备执行的路径；

（3）对步骤（1）的每个限定条件，当满足一个特定的限定条件时，规定一个由所述计算设备执行的路径；

（4）规定一种限定配置状态变成有效配置状态；

（5）将表示上述步骤的数字数据存入一个本地计算设备；以及

（6）通过通信连接将步骤（5）的数字数据下载到所述远程计算设备中。

33. 如权利要求 32 所述的方法，其特征在于所述通信连接是一个网络。

34. 如权利要求 33 所述的方法，其特征在于所述网络包括因特网、内联网、外联网或 LAN。

35. 如权利要求 32 所述的方法，还包括将一个路径元素指向于以不同格式的第二编程语言编写而成的程序的步骤。

36. 如权利要求 32 所述的方法，还包括评估所述远程计算设备结构的步骤和将上述步骤配置成同所述远程计算设备结构运行的步骤。

37. 如权利要求 32 所述的方法，还包括将所述步骤中规定的至少部分数据形成一个表格格式的步骤。

38. 如权利要求 32 所述的方法，还包括将表示所述配置状态和路径的数字数据存入所述远程计算设备中以便执行的步骤。

39. 如权利要求 32 所述的方法，还包括将配置状态和路径的至少部分说明转换成不同格式的第二编程语言的步骤。

40. 如权利要求 32 所述的方法，其特征在于配置状态或路径不必

按照顺序依次列出。

41. 一种由第一处理器和第二处理器构成的多处理器计算设备，其特征在于所述第一处理器用于执行一个具有 m 种配置状态和 n 个路径的表格格式程序的至少一部分。

42. 如权利要求 41 所述的多处理器计算设备，其特征在于第二处理器执行用不是表格格式的第二语言编写而成的程序。

43. 如权利要求 42 所述的多处理器计算设备，其特征在于第二处理器用于执行一个由所述第一处理器执行的表格格式程序所指导的程序。

44. 如权利要求 41 所述的多处理器计算设备，其特征在于所述第一和第二处理器位于一个集成电路上。

45. 如权利要求 41 所述的多处理器计算设备，其特征在于处理器之一可执行的至少一个指令不同于另一个处理器可执行的指令集。

46. 如权利要求 41 所述的多处理器计算设备，还包括用于在所述第一和第二处理器之间传递参数的装置。

47. 一种用于构成一个适于对具有 m 种配置状态和 n 个路径的表格格式程序进行编译的编译器的方法，包括步骤：

（1）识别表示配置状态的区域；

（2）识别表示路径的区域；

（3）识别一种配置状态的至少一个限定条件并将其与具体的路径相连接；以及

（4）将配置状态 A 与一个将引用所述配置状态 A 作为其元素的路径相连接。

48. 如权利要求 47 所述的方法，还包括用于将步骤（1）到（4）的功能综合为一个现有语言的编译器的过程。

49. 如权利要求 47 所述的方法，在步骤（1）中还包括一个识别表示配置状态开始部分的关键字的过程。

50. 如权利要求 47 所述的方法，在步骤（2）中还包括一个识别表示路径开始部分的关键字的过程。

51. 如权利要求 47 所述的方法，还包括使一个用户定制表达式与表格格式编程语言的一个具体指令相等的步骤。

52. 如权利要求 51 所述的方法，还包括识别一个使用户定制表达

式与表格格式编程语言的预定义指令集相等的表格的步骤。

53. 如权利要求 47 所述的方法，还包括根据所述表格格式程序的指令集识别不可执行的标号的步骤。

54. 如权利要求 53 所述的方法，还包括将所述不可执行的标号与一个外部程序相连接的步骤。

55. 如权利要求 54 所述的方法，其特征在于所述外部程序是用包括表格格式编程语言在内的任何可用编程语言编写而成的程序组成。

56. 如权利要求 47 所述的方法，还包括区分两个或多个程序组的步骤，其中每个程序组由至少一个配置状态表和一个路径表组成。

57. 如权利要求 56 所述的方法，还包括对所述多个程序组之间的交互作用进行编译的步骤。

58. 如权利要求 57 所述的方法，还包括定义一个用于激活一个程序组的指令的步骤。

59. 如权利要求 58 所述的方法，其特征在于所述指令用一个或多个任务表表示。

60. 一种对计算装置编程的方法，包括步骤:

（1） 选择一种具有预先定义的指令集的编程语言;

（2）定义用于表示选定语言的一个具体指令的替换表达式;

（3）用所述替换表达式编写程序; 以及

（4）将所述程序配置成所述计算装置可执行的程序。

61. 如权利要求 60 所述的方法，其特征在于选定的编程语言是表格格式编程语言，包括:

x 种配置状态，所述配置状态中的至少一种定义了一个或多个限定条件;

y 个能被所述计算设备执行的路径; 以及

当满足一个特定限定条件时，所述计算设备就执行一个路径。

62. 如权利要求 60 所述的方法，其特征在于步骤（4）是一个由编辑器、编译器、解释程序或一个翻译程序执行的翻译过程。

63. 如权利要求 62 所述的方法，还包括显示用经过翻译的选定语言的预定的指令集组成的程序的步骤。

64. 如权利要求 62 所述的方法，还包括显示用用户定制表达式组成的程序的步骤。

65. 如权利要求 60 所示的方法，还包括提供一个位于程序内部的表格以便将所述替换表达式与相应的特定指令进行链接的步骤。

66. 一种用于对响应一个或多个虚拟限定执行一个或多个路径的计算设备进行编程的编程方法，所述编程方法包括步骤:

（1）规定 x 种配置状态，所述配置状态中的至少一种定义了一个或多个限定条件;

（2）定义步骤（1）中的限定条件中的至少一个以表示一个虚拟限定;

（3）规定 y 个由所述计算设备执行的路径;

（4）对于步骤（1）中的每个限定条件，当满足所述限定中的一个具体限定条件时，再规定一个由所述计算设备执行的路径;

（5）规定限定配置状态中的一个变成有效配置状态。

67. 如权利要求 66 所述的编程方法，其特征在于所述计算设备被定义为第一计算设备；所述编程方法还包括通过通信连接从与所述第一计算设备距离遥远的第二计算设备接收表示上述步骤的数字数据的步骤。

68. 如权利要求 66 所述的编程方法，还包括规定一种或多种配置状态以构成一种输出配置的步骤。

69. 如权利要求 68 所述的编程方法，其特征在于所述输出配置定义了所述计算设备的一个或多个输出终端的输出条件。

70. 如权利要求 68 所述的编程方法，其特征在于所述输出配置定义了由所述计算设备产生的虚拟计算输出。

71. 如权利要求 66 所述的编程方法，还包括将配置状态和路径说明转换成所述计算设备可执行的数字数据的步骤。

72. 如权利要求 66 所述的编程方法，还包括用于转换利用不同格式的第二编程语言的配置状态和路径的至少部分说明的步骤。

73. 如权利要求 66 所述的编程方法，其特征在于 x 种配置状态和 y 个路径不必按照顺序彼此依次列出。

74. 如权利要求 66 所述的编程方法，还包括提供一个用于识别由所述步骤组成的程序位置的关键字的步骤。

75. 如权利要求 66 所述的编程方法，还包括将步骤（1）中的配置状态分成一个或多个表格的步骤，每个表格中有一种配置状态被规定

为有效。

76. 如权利要求 66 所述的编程方法，还包括构成一个独立的用于确定由步骤（1）到（7）组成的程序是有效还是无效的表格的步骤。

77. 如权利要求 66 所述的编程方法，还包括下列步骤:

（6）提供一个预先定义的指令集以便对路径和配置状态编程;

（7）定义用于表示步骤（6）中的指令集的一个特定指令的替换表达式;

（8）用替换表达式编写程序；以及

（9）将所述程序配置成所述计算装置可执行的程序。

78. 如权利要求 77 所述的编程方法，其特征在于步骤（9）是一个由编辑器、替换表达式和相应的特定指令执行的翻译过程。

79. 如权利要求 66 所述的方法，还包括规定一种配置状态为有效的步骤。

80. 如权利要求 66 所述的方法，其特征在于将一个路径定义为在初始化时将被执行的缺省路径。

81. 一种用于在一个计算设备的程序列表中识别预定类型的字（wordings）的编程方法，包括从多种预定字格式（wording style）选择中选择一种字格式的步骤。

82. 如权利要求 81 所述的编程方法，还包括步骤:

（1）定义 x 种配置状态，所述配置状态中的至少一种定义了一个或多个限定条件;

（2）定义由所述计算设备执行的 y 个路径；以及

（3）当满足一个具体限定条件时，向所述计算设备分配一个路径以执行。

83. 如权利要求 81 所述的编程方法，其特征在于通过一个具有关键字的表格来定义所述字格式。

84. 如权利要求 81 所述的编程方法，其特征在于所述预定字型被配置成高亮度的用户定制标号。

85. 如权利要求 81 所述的编程方法，还包括形成两组或多组用户定制字格式并且其中一组被定义为有效的步骤。

86. 一种用于对响应一个或多个限定条件执行一个或多个路径的计算设备进行编程的编程方法；所述编程方法包括步骤:

（1）规定 x 种配置状态，其中 x 是一个大于等于 1 的整数；

（2）规定一个或多个到于所述 x 种配置状态中的至少一种的限定标号；

（3）使步骤（2）中的限定标号指向一个用于描述限定条件或所述标注所表示的条件的独立的表达式；以及

（4）规定 y 个可执行路径，其中响应步骤（3）中规定限定条件来执行至少一个路径。

87. 一种计算装置，包括：

用于执行一个程序的计算装置；

用于访问于一个远程计算设备相连接的通信连接的装置；以及

用于存储所述计算装置或所述远程计算设备所执行的数字数据的存储装置；其中所述数字数据包括具有 x 种配置状态和 y 个路径的表格格式程序表示；所述配置状态中的至少一种定义了一个或多个限定条件；而在满足一个提单限定条件时，所述计算装置执行所述路径中的一个。

说　明　书

表格格式编程

本申请是1995年10月2日提交的申请号为08/538,426且专利号为US 5,867,818的美国专利的部分继续申请。在此引用此申请作为参考。

本发明涉及一种通过填写不同种类的表格来实现编程功能和便于控制程序流的对计算设备进行编程的方法。

本发明通常涉及一种编程工具，该编程工具被设计成程序员与计算机之间的接口。这种编程方法工具广泛使用表格来表达程序员的逻辑思维过程并使编程过程更易于为他人所理解。因此，这些改进提高了编程效率，减少了出现程序故障和结构错误的机会。另外，一个程序员学习这种编程方法所需的培训成本是最低的。将来任何程序员都能够很容易地读懂和维护依据本发明而编写的程序。

传统的编程语言定义一组编程指令和编程规则。诸如BASIC、C和JAVA之类的通用编程语言是按照从上到下逐行顺序列表的形式编写程序。在很多应用程序中，需要编写几百页代码来描述要求的作业功能。要求一个程序员从很长的程序列表中指出一个程序的逻辑流程是非常困难的。因此，程序的长短为将来的维护工作带来了困难。除此以外，多页充满了代码行序列的程序很难被以前没有涉及此作业的其他程序员所理解。虽然已经利用了大量的压缩指令或编程符号，但以不同编程语言编写而成的程序的长短仍超出了一个专业程序员能够容易理解的范围。在很多情况下，一个程序的原作者会发现经过了一个较长的时期之后，他也很难理解他或她本人所编写的程序。压缩指令集的要求与可读性是彼此矛盾的。因此非常需要一种编程方法，该方法能够提供压缩的程序长度并易于被其他程序员及非专业人员所理解。

每次发明一种新的编程语言时，都定义编码符号和指令集来描述一个程序功能。还定义规则来限制如何使用编程指令以便计算设备能够翻译和执行利用该方法编写而成的程序。在很多情况下，为了确证所有这些指令集并向程序员讲述编程规则，用以描述一种编程语言的指令手册超过了两英寸厚。利用几个月来培训一个程序员学会所有的

指令集并理解一种新编程语言的所有编程规则是非同一般的。通常是通过数年的编程经验而学到更多的编程技巧。

虽然很多编程语言允许一个程序按照该作业的结构要求来提供条件跳转、调用功能或分支到程序的其它分段，但是一个专业的程序员在花费了大量时间阅读该程序之后仍很难解释这些交互式分支操作在程序结构中的逻辑流程。

市场上提供了各种可用的编程语言，每种专用于一种特定应用或按照特殊的编程环境而设计。许多其他的编程语言被设计成程序指定硬件配置或系统。在很多情况下，不同的编程语言使用不同的符号或表达式来表示一个描述相似功能的指令。一个程序员在学习了多种分别具有不同格式和指令表达式的编程语言之后，对于应当使用什么符号或表达式经常会感到混淆不清。因此非常需要一种能够使程序员克服这些困难的新的编程方法。

由于大多数编程语言被设计成在其指定的应用环境中能够最佳地解释编程功能，所以很多涉及多种功能状态的编程作业最好被分解成多个模块，每个模块是按照其指定的环境以不同语言编写而成的。因此，非常需要一种适用于管理以不同语言编写的程序段的集成的专用编程方法。随着因特网的普及，非常需要一种通用的编程方法，该方法适用于协调不同格式的编程模块以形成一个组合程序，并适用于将程序下载到与主计算机距离遥远的单个本地计算机上。对于这种类型的应用，程序必须是压缩的并且能够根据不同本地计算机的种类进行自重构。

专利号为US 5,867,818的美国专利最先引入了一种用以对一个具有多个输入和输出端子的硬件控制器芯片进行编程的原始表格格式方法。本申请指向在发明这种原始结构之后进行的进一步研究中所实现的各种改进。

本发明面向学习一个通用模块结构的方法以便利用表格格式编程的概念来对一个计算设备进行编程。本发明的一个目的是创建用以指导计算设备操作的编程方法的一个通用模块并组建该模块以使普通人员无须接受专业培训即可使用该模块。本发明的另一个目的是创建一个编程模块使得以这种格式编写的任何程序易于被不具有编程语言方面的经验的其他人员所理解或者可包含在已被编写的程序中。本发明

的另一个目的是创建一个功能通过编码过程而清楚地标识编程作业的结构的编程模块，从而能够花费最少的力量来维护该程序。其结果是，能够相当多地减少学习一种编程语言的培训成本、编写程序的时间成本、调试程序的时间成本以及修改程序的维护成本。通过一种用户友好的编程结构，希望在编程过程中出现较少的编程错误，从而减少调试所需的时间成本。

本发明的另一个目的是创建一种结构编程格式，该格式能够清楚地表达该作业的骨架结构并使该单个具有特定功能的程序模块以选定的最适于该功能的任何语言编写。这种方法进一步提高了编程效率并减少了将来维护工作的工作量。

本发明的编程方法包括以程序数据填写两个以上表格的步骤。各种类型的表格被设计成用以执行不同的支持功能。一个表格被定义为一个数据矩阵。一个表格可以是一维或多维的。通常用 m 行和 n 列表示一个表格矩阵，其中 m 和 n 是大于等于 1 的整数。可用一个标签、一个表达式或一组表达式来表示表格的每个元素。

在本发明的第一实施例中，编程方法引入了任务表的概念，任务表包括一个用于控制多个任务的操作的数据表格。一个程序中可能存在一个以上的任务表并且每个表格由一种或多种任务状态构成。在编程过程中无论何时指定一种任务状态，都可以定义所指定的任务的状况和/或操作。而且任务优先权信息可包括在操作任务表或由该计算设备的另一个单独表格以便在以计算设备的共享资源运行多个任务的同时分配处理优先权。

在本发明的基本方面，形成一个表格来定义包括输入和/或输出信息在内的一种或多种配置状态。当满足了一种配置状态的输入限定条件时，将以另一个被称为事件表或路径表的表格中所表示的一系列特定操作作为响应。将至少一种配置状态指定为有效状态。用于微控制器的表格格式编程方法的最早版本在美国专利 US 5,867,818 中有所描述。本申请是面向多年的研究对基本概念的各种改进以扩大该技术的服务范围。

在本发明的另一个实施例中，形成另一个表格来定义触发一种限定符配置状态所需的限定条件。在本发明中，首次引入了虚拟限定符的概念。一个虚拟限定符被定义为表示来自一个物理硬件终端的任何

限定条件。在考虑到由一个软件指令所造成的结果或由整个软件程序造成的结果时，虚拟限定符是非常重要的。虚拟限定符还涉及计算设备内部的硬件，如内部寄存器的溢出。其可通过一个内部软件或硬件中断的结果，一个标记的产生、极性信号、表示存在来自另一个系统的数据或存在一个标记的信号而被触发。虚拟限定符的另一个通用例子是一个鼠标驱动程序的输出，它可以指示一个指针随鼠标的移动而移动的方向。

在本发明的另一个实施例中，形成一个表格来定义一个输出状态的输出配置。该输出状态可以是一个将通过一个物理硬件终端而被发送的信号或一种虚拟计算输出状态的配置。虚拟计算输出状态被定义成任何一种不描述硬件终端的操作的输出状态。虚拟计算输出状态的一个例子是触发一个软件程序开始运行，或设置用于控制一个软件的特殊条件参数，或触发软件以操作某些特定功能。

当获得合格的触发时，执行一个相应的操作。路径表或事件表是用于将一个或多个相应操作进行分组的表格。一个路径可表示在接收到合格条件时所要执行的一个操作或一系列操作。一个路径可被另一个路径初始化。这个相应的操作或操作序列被称为一个路径方程(path equation)。状态表与路径表的交互式组合与状态图所示的事件结构相似。除此以外，表格格式编程还提供了一种能够更好地反映出人类思维的程序表示方式，从而使其对用户更为友好。为了更好地表达程序流的描述内容，可根据程序员的意愿为路径表中的每个路径分配一个有意义的标号。一个有意义的命名标号可被分配给一个配置状态或一个路径。所有这些标号将帮助程序员或其他人员理解所编写的程序的操作和逻辑流程。这是本发明的一个重要贡献，有助于解释编译程序和减少将来的维护成本。

所发明的编程方法中所包含的另一个表格将一个输出状态下的输出条件指向于一个特定的操作序列。可用另一个表格来表示这个序列以便简化编程过程。

本发明中所引入的另一种类型的表格使程序员能够将表格格式编程语言的表达式和/或语法变成所需的其它形式的表达式或符号。所包括的其它表格定义了一组或多组程序库或外部程序以支持编程工作。

在表格格式编程方法的支持之下，传统的编程过程被大大改变。

当开始结构编程作业时，程序员用状态和路径表对程序流的骨架结构进行交互式的描述。程序员可自由地分配有意义的标号或名称以便表示每种配置状态和路径。对每种状态配置和路径方程的有意义命名将有助于描述程序的流程。当包括在一个路径中的一个操作难以用可得到的指令集表达或最好用另一种语言的程序模块来描述该操作时，程序员自由地分配一个有意义的标号来表达这个所需的操作。在编程过程的结尾，程序员应当为所分配的每个未定义标号提供可执行的程序模块。以这种方式，通过交互式地组织多组表格而形成了程序模块。一个具有至少一个状态表和一个路径表的程序模块被称为一个程序组。可向每个程序组分配另一个有意义的标号以便描述程序模块的功能。然后在需要跳转或功能调用时，这个标号可用于另一个程序模块或程序组以指向此程序组。为方便起见，将在程序开始时所执行的缺省的第一个程序组称为“main”程序组。在每个程序组中，应当具有一个“start”路径，用作在初始化程序或缺省启动程序时的启动路径。为了改善编程效率，可用不同于主程序组中所用的表格格式编程方法的语言来构成支持程序模块。在同一程序中可使用多种不同的语言。这就非常希望按在表格格式编程方法中具有能够在不同语言的程序模块之间传递参数和变量为特征设计。在一个最佳实施例中，一个表格格式程序被设计成与不同编程语言的接口并被用作在以不同语言编写的程序模块之间进行通信的桥梁。可以说表格格式编程语言大大改变了编写程序的概念和习惯。在一种富有逻辑和较好结构的方式下编写一个程序。通过自由地分配有意义的标号，程序流程被尽量平滑地组合而成以便描述该编程作业。然后每个标号与一个用任何语言编写的外部程序模块相链接。有时，可排列成多级程序组或模块。由于每个表格格式程序组的程序长度通常非常短且该程序具有合理的结构，从而能够容易地实现表格格式编程的上述优越性。

对于表格格式编程来说，自由分配或使标号与指令关键字等效对于支持多语言编程平台是非常重要的。这是由于多种语言可对同一类型的应用程序使用不同的语法。标号的自由分配和等效特征使用户能够根据他/她的意愿而统一标号和语法。

表格格式编程的特性使其成为一种用以增强另一种语言的编程结构的极佳支持工具。例如，一种高级语言的编译程序可被修改成包括

仅足以提供以该语言编写而成的一个程序的骨架结构的表格格式编程函数。

由于表格格式编程方法是唯一和独立于其协同的第三种语言程序的，所以通过只提供一个主要用于处理所有表格格式编程任务的简单表格格式协处理器就能够简化程序管理作业。这种多处理器结构将减轻主处理器的工作量并使其集中处理常规作业和提高尤其在多任务运行环境中的整体系统性能。表格格式协处理器可以是一个位于主处理器外部或与主处理器共存于同一集成电路芯片上的处理器。当表格格式编程应用于微处理器或微控制器时，经过编译或编码的表格格式程序变成了存储在诸如 ROM、EPROM 或闪速存储器之类的存储器中的用于微处理器或微控制器运行的数字数据。然后一个由处理器和存储表格格式程序的编译形式的存储器构成的印制电路部件被用于制造可出售的商业成品。

根据表格格式编程的结构组织，它特别适用于事件驱动的应用程序，例如在 windows 环境下的编程，网站，交互式游戏或控制编程。用于事件驱动编程中的表格格式的一个重要特征是用高度压缩的格式来表示一个程序。大多数用于小作业的表格格式程序的长度都少于一页。证明了表格格式编程中的一页可表示八页汇编语言程序的实验性结论是根据可用指令集的复杂性而作出的。与很多高级语言相比，还可在相当大的程度上节省代码的长度。这种特性使其在用作通过一个有限带宽的通信信道进行通信的媒体时是相当经济的；在例如与网络或因特网应用中的多个本地计算机进行通信的主系统之间。应当注意一个网络，通信连接或通信信道是指将两个计算设备连接在一起的任何装置，包括串行端口、并行端口、USB 端口、因特网、内联网、外联网、LAN 和任何使两个计算设备接口的通信装置。两端的设备可通过有线、无线或混合连接模式进行连接。在因特网编程中，在本地计算机执行一个下载程序之前，还需要一个估计用户计算机的系统配置，然后调整程序设置的步骤。计算机中很多与人接口的设备配置，例如监视器、图形卡、声音发生设备、指针控制、游戏控制器控制等都属于将被定义的配置设置。由于因特网通信线通常是一个系统的瓶颈，所以建议在本地计算机上执行表格格式编程的编译作业，以便利用表格格式编程的压缩代码的优点。

图 11 是示出了一个本地计算机 801 如何与一个远程计算机 803 相连接的方框图。在很多应用中，一个表格格式程序被存储在本地计算机中。根据请求，通过一个通信连接 802 将该程序下载到远程计算机中。该表格格式程序可存储在远程计算机的存储装置 804 中或立刻由远程计算机进行编译。编译文件是一个用以执行远程计算机的某些预定工作的可执行文件。这个可执行文件也可存储在远程计算机的存储装置 804 中。

当一个编译器被设计成用于处理表格格式编程时，着重建议使用关键字来识别每个表格以及相应程序组的状态和位置。以任务，程序组，限定符，状态，路径和程序库为示例关键字用于识别所示实施例中的功能表格。应当注意表格格式编程方法的编译，翻译，解释或转换包括将表格格式程序变换成其它程序格式，例如机器语言或任一种更高级语言的进程。该变换进程可由用另一种编程语言编写或被一个可执行表格格式程序支持的编译程序执行。

任务表的概念有效地提高了在多任务运行环境下进行表格格式编程的便利性。本发明的特征如后面的权利要求所述。对计算设备的限制是指具有计算能力的任何设备，包括计算机、微控制器、微处理器、由微控制器或微处理器构成的印制电路部件。除计算机以外，用以支持本发明技术的其它支持硬件包括调试硬件、诸如电缆、通信口、网络集线器之类的通信连接以及特定的网络。表格格式程序可被显示在显示终端和印刷品上，并可被编码为数字数据。表示该表格格式程序的编码数字数据被存储在诸如 RAM、ROM、磁盘驱动器和 CD ROM 之类的任何存储设备中。实施例中所用的技术术语、关键字和标号只是作为例子，它可具有各种变形和修改，并且能够容易地预见到表格格式的重新排列可达到相同的效果，所有这些特征都包含在后述的权利要求范围内。结合附图的详细说明更有利于本发明的理解。

为了充分利用本发明技术的有益之处，用户需要为计算机预处理操作进行精心的准备。典型的计算机预处理操作包括分析程序作业说明，将方案简化为表格格式的相关状态图。在提高程序结构清晰度的同时引入表格格式程序组需要用户在开始应用编程技术之前清楚地识别和定义方案的具体功能模块。由于表格格式编程方法的优点，另一个接口说明过程需要确证表格格式程序组或模块的接口关系以便为一

组程序员分配编程作业。在网络通信或下载应用中，计算机预处理操作涉及通过一种通信连接或网络向一个远程计算机发送表格格式程序。

通过表格格式编译器执行处理中的计算机操作，该编译器将表格格式程序翻译成本地计算机、目标微控制器或远程计算机可执行的代码。后一计算机操作通常是目标计算机或微控制器可执行的代码。这个可执行代码还可被计算机或微控制器运行以便根据原始的编程说明来执行功能。编译后的可执行代码通常存储在诸如 RAM、ROM、任何可编程非易失性存储器之类的存储装置或任何其它商业可用的存储设备中。在微控制器作为消费品的情况下，用于存储编译后的可执行文件的存储装置通常位于出售品中而不是位于编译计算机中。在这种情况下，编译计算机只是作为一个开发系统或远程计算设备的程序供应方。

图 1 示出了表格格式程序的一个实施例的结构。

图 2A 是用于定义等效于用户定义表达式的表格格式表达式的表格。

图 2B 示出了如何将一个名称分配给图 2A 所示的表格。

图 3 是用于定义将被识别的字的不同打印格式的表格。

图 4 是示出了一个任务表的不同任务状态的实施例。

图 5A 是示出了一个程序中的多个任务表的实施例。

图 5B 是图 5A 的数字化表示。

图 6A 是示出了一个表格格式组的一个实施例的程序。

图 6B 是图 6A 的实施例的另一种开发形态。

图 7 是示出了所包括的文件列表的表格。

图 8 是图 7 所示的表格的另一种形态。

图 9 示出了用于表格格式编程中的一些建议长度和描述性的指令命令。

图 10 示出了现有技术中用于声音发生微控制器中的基本表格格式程序。

图 11 是示出了一个本地计算机通过一种通信连接将一个表格格式程序下载到一个远程计算机中的方框图。

在下面整个详细描述中，所有附图中的相同附图标记表示相同的

元素。

首先参照图 10 中所示的由美国专利 US 5,867,818 公开的基本表格格式编程。该表格格式程序由用以对一个声音发生可编程控制器编程的两部分构成：一个状态表和一个路径表。在状态表的第二行中，定义了控制器的相应输入触发引脚的顺序。状态 0 到状态 4 中的每一个定义了控制器的一种可能触发状态。例如，R: Path1 元素指向于状态 0 的 TG1，表示如果检测到一个上升沿（用“R”表示），则执行命名 Path1 的路径。在 Path1 中，有效状态改变到状态 1，之后产生一个名为“Sound1”的声音。在发出该声音之后，控制返回 Path1 并开始另一个循环的声音发生序列，直到在 State1 状态中由 TG1 接收到一个下降沿触发（用 F: Path11 表示）为止。这个程序例示出了 TG1 到 TG4 的“level hold”功能。即，当压下 TG1 到 TG4 中的一个时，将产生一个声音。声音将是循环发生的，直到放开触发按钮为止。

下面参照图 1，图 1 示出了一个改进后的表格格式程序实施例中所选定的一个表格的结构。关键字 101 表示程序的开始。程序员在关键字下面的位置上指定了程序名。关键字 106 表示一个列出了所包括的程序模块的表格的开始部分。关键字 107 表示一个定义了该程序中所用常量的表格的开始部分。关键字 102 对程序中所用的变量进行说明。关键字 103 是一个列出了用户定义的等效命令和语法表达式的表格。为了使程序关键字是可区分的，程序员可利用关键字 104 在表格的开始部分定义关键字的打印格式。上述所有特征是为表格格式程序做准备工作。任务表 105 提供了一种或多种任务状态，用以定义哪个任务是有效的、暂停的或终止的。程序组 111 由一个可选的限定表 112，至少一个状态表 113 和一个路径表 116 组成。实际的程序作业由状态表和路径表的内容交互构成。可附加输出状态表 115 和输出方程表以便进一步定义程序的输出状态。最后，附加一个程序库表 121 以便提供通用的命令串和子程序。应当注意上述所有表格无须按顺序排列并且很多表格的建立只是为了提供可选的特征。另外，一个程序可包含多个相同类型的表格，如任务表和状态表的情况。可在关键字冒号之后随意指定表格的名称以便更好地体现该程序的含义。表格格式编程中所用的关键字可多于一个字并且可以用指令、具体的变量、常量和系统硬件来表示。应当注意所提供的关键字只是举例说明，还可

使用其它的关键字名称。除此以外，对表格的范围进行合理的改进是可能的，应将其视为本申请的保护范围之内。

现在参照图 2A，表格 100 代表图 1 中的表格 103 的具体例子。关键字 201“定制表达式（Custom Expression）”表示这个功能表的开始部分，该表列出了等效于正规表格格式表达式的用户定义表达式。例如，假设逻辑 AND 功能的正规表格格式表达式是 203 所指的“AND”；一个习惯用 C 语言编程的程序员可随意地用 202 所示的 C 语言指令集中的“&&”来代替表格格式命令“AND”。定制表达式适用于语言或系统设置的任何字或符号，例如指令命令、符号和系统关键字。建议每次在定义一个替换表达式时提供一个注释 204. 这个功能的优点在于为程序员提供个人化的支持，以便总可以使用惯用的符号或表达式。但是，当打印所编写的程序时，编译器或编辑器最好以正规表达式打印出程序清单以便于其他人员阅读。对程序编辑器亦是如此。只要键入用户定义的表达式，将可以显示出正规的表达式。还建议在预定的正规或用户定义的表达式之间提供一种转换功能，使用户可以选择是以正式的形式还是以定制的形式显示或打印程序。利用这种用户定制表达式的特征，在保持程序员使用缩写的表达式或较短的符号来表示指令命令和语法的同时，还可利用有意义的长表达式名称使程序清单对其他人员而言更具有可读性。利用公知的查表方法，在表格格式程序中建立正式的语法组和相应的用户定义的等效语法和标号组。一个典型的应用实例如图 9 所示，图中列出了一些较长的描述性指令集。移位操作符“BIT SHIFT LEFT”和“BIT SHIFT RIGHT”清楚地描述了所要执行的操作。但是，这些指令对于有经验的程序员来说太长和不受欢迎。程序员将“BIT SHIFT LEFT”指令等效于 C 语言中的简洁而描述性较差的“<<”指令。长的指令名使使程序易于理解但对有经验的程序员来说是无用的。“定制表达式”表在保持长指令表达式的优点的同时有效地解决了这个问题。

由于 QWERT 键盘上可用的符号是非常有限的，因此从键盘上很难发现足够的有意义符号来组成一种新的象表格格式编程方法那样与其它某些通用记号和符号的传统用法不相矛盾的编程方法。通过使用户能够按照自己的意愿重构指令和符号，定制表达式表可作为一种解决此问题的方式。

元素 210 示出了了如何将一个名称“MySign”分配给“客户表达式”表。当为了使一个以上的用户操作或阅读程序而提供一组以上的“客户表达式”表时，特别需要一个名称。例如，除了包括在程序中的表“MySign”以外，另一个名为“JohnsSign”的定制表可被加到同一个程序中。如果 John 想要阅读该程序，只需将“JohnsSign”表达式表设为缺省显示表，John 将可以以他所喜欢的格式来显示该程序。此表的新特性使每个用户都包括他们自己的定制表达式设置，以便于按照他们喜欢的格式来转换或编辑程序。

尽管图 2A 所示的表格提供了一种重构关键字和表达式的方法，但需要一种为一个小作业提供另一种语言的替换表达式或命令的简单方法。这是可以通过在用该语言规定的表达式前指定一个表示该语言的符号来实现的。图 2B 示出了如何用二进制数 00001111 屏蔽（mask）寄存器 A 的内容以便于得到寄存器 A 最后四位的内容，然后进一步显示这个数字。表达式 216 以元素“（C: &&）”来表示“&&”指令是一个“C”语言指令。然后利用一个预定义的“Display”命令来显示寄存器 A 的屏蔽值。可替换地，利用一个微处理器的汇编语言中的“&”指令可得到执行同一功能的表达式。标记“A: （）”是一个表示括号中的操作是以汇编语言编写而成的表达式。尽管这两种方法都很方便，但是仍没有图 2A 中所示的能够使一个全程序的个人化表示式是被转换或个人化的定制表达式表功能强大。

由于表格格式编程中涉及用户指定的大量的分立标号，并且这些标号分散在程序中并与关键字和指令命令相混合，因而对于一个阅读该程序的用户来说，难以从其它关键字和指令命令中识别出这些标号。因此最好提供用以识别这些标号的装置以便使该程序对用户更为友好。图 3 示出了一个控制如何表现用户分配标号的程序表。表元素 234 提供了可选择的情况。可用于选择的典型情况包括首字母、所有大写字母和所有小写字母。表元素 239 表示可选择的字母类型。典型的可选择类型包括黑体、斜体和下划线。应当注意元素 234 和 239 都为包括黑白打印机在内的各种显示设备提供了卓越的识别功能。关键字 231 表示用于定义程序部分的识别类型的开始部分。建议在关键字 231 后面设置一个由用户分配的名称以便表示下面的设置最适用于某个特定的人员。在程序中可包括多个依据个人喜好而分配的识别类型

表，并且可为每个表分配一个用元素 232 表示的名称。选择其中一个识别类型表可将打印格式设置成适用于正在阅读该程序的特定用户。

为了表示在多任务运行环境下具有较好结构的程序流，在图 4 所示的实施例中引入了一个任务控制表（此后称为任务表）。元素 261 是用于识别一个任务表的关键字。元素 262 是一个对任务表命名的标号。提供这个名称是为了在需要两个或多个任务表时与其它任务表相区别。元素 263、264 表示可在任务表的控制下运行的不同任务或程序。任务表的每一行表示一个任务状态。在每种任务状态下，指定一种任务条件以表示每种任务的运行条件。下面列出了用以描述一种任务条件的表达式的几个例子:

Start: 表示不论该任务或程序是正在运行、中止或已经结束，都从头开始重新运行该程序;

Continue: 表示如果一个任务已经开始运行，则继续运行之;

Pause: 表示任务或程序被置于中止状态;

Run: 表示如果一个任务还没有开始运行，则开始运行之；如果一个任务或程序正在运行，则继续运行之；如果一个任务被中止，则恢复该任务的运行;

X: 表示结束该任务的运行。

在行 265，任务的名称为“Task Status 1”，它指示“Main”程序开始运行而程序 2 到 n 都处于结束状态。在任务状态 268，指示所有的程序运行。应当注意对每个任务表来说，在任何时刻只有一个任务状态被指定为有效。在具有有限资源的系统中，向正在运行的有效任务指定优先权是非常重要的。任务状态 273 和 274 为任务们分配优先权。应当注意可以建立一个独立的表格来描述分配给任务的优先权。由于有效任务表和优先权任务表中每一元素的开头元素是相同的，所以可以将两种类型的任务表组合成一个任务表，如图 4 所示。在这种情况下，就需要两种有效任务状态，一种用于任务有效状态，而另一种用于任务优先权分配。

图 5A 示出了一个介绍任务表概念下的实际应用例子。图 5B 是图 5A 的标号表示。这个例子由三个任务表组成。第一任务表名为“Input”，如元素 301 所示。它由三个分别名为“Keyboard”、“Mouse”和“Gameport”的程序组成。“Keyboard”是一个用于扫描键盘上的

按键的程序。“Mouse”是一个对鼠标的移动进行译码的程序，而“Gameport”则输入来自博弈端口的触发信号。在名为“All”的任务状态305中，运行三个程序以使计算设备对三个输入设备都有响应。在名为“Normal”的任务状态306中，则只认可键盘和鼠标。不使用博弈端口是为了提高计算设备的服务效率。在处于游戏模式时，名为“Game”的任务状态309变成唯一的有效程序或任务。不使用键盘和鼠标是为了使计算设备将所有资源集中在游戏上。第二任务表是一个名为“Ports”用于控制计算设备的串行和并行端口的表格。名为“Device”的第三任务表如元素341所示。它控制驱动器程序操作“CDRom”342、“HardDriveC”343和软盘驱动器。当任务状态“ReadCD”345被激活时，CD Rom和Hard Drive C驱动器程序被激活而软盘驱动器程序被终止。在需要硬盘驱动器全速运行的模式中，任务状态“HDFullSpeed”346变成有效任务状态而硬盘驱动器变成唯一正在运行的设备。根据这个应用实例，建议只对具有相似内容或相互关联的任务进行分组以形成一个共用的任务表。应当注意在任何时候，每个任务表中只有一个任务状态被指定为有效。

参照图6A，示出了表格格式程序的一个主程序组。该程序名为“WebSale”，提供了一个通过因特网提供的销售程序的骨架结构。这个例子示出了在表格格式编程环境下进行多种语言编程的概念。为了便于描述该实施例而在程序中插入了行号。应当注意状态和路径方程不必是按照顺序排列的。现在参照第1行。关键字“Group”表示一个表格格式组或程序模块的开始部分。该组的名称为“Main”。将Main作为一个关键字用以表示这个组是在开始运行该程序时所要执行的第一程序组。第2行以一个关键字“qualifier”开始，它定义了在配置状态中所列出的限定符的限定条件。在第3到6行，将术语“Icon”用作一个功能性命令，用于构造一个图标并在单击该图标时触发该配置状态。在典型的表格格式编程中，按顺序是数形成大量图标。在限定表中定义和命名每个图标。例如，当Icon(1)指向于名称“Catalog”时，字“Catalog”被指定给第一图标并显示于其上。实际上，一旦将一个名称指定给一个图标，其编号就是不重要的，除非在一个程序中提到了术语“icon(n)”并且“n”是一个计算结果。第7行定义了一个名为“FirstPage”的输入状态配置表。在这个表中指定了五种限

定符，即“Catalog”、“Purchase”、“Service”、“Home”和“Quit”。每种限定符系指一个图标的触发，如第2行的限定表中所定义的。第一输入限定符状态名为“Ready”，如第8行所示。在这种状态下，当接收到一个代表图标“Catalog”的限定触发时，执行名为“P_catalog”的路径，对其它限定符来说亦是如此。第9行示出了另一个名为“Hold1”的输入配置状态。状态方程中的“x”表示相应的限定条件是无关条件，在该条件下当出现限定触发时，该触发被阻塞或不需要响应。

下面参照提供了另一个名为“Response”的配置表的图6A中的第11行。这是一个具有五个元素的输出状态配置表。前四个元素具有一个关键字“Group: ”，表示构成了一个程序组的程序。冒号后面是程序组的名称。第一组名为“Info”，用于提供产品信息。第二组“Order”是一个指导用户通过诸如登记信用卡号、产品号、定单数量、总量、选择之类的购买过程，对数据加密并将定单发送到供应商进行译码的程序。第三组“Service”提供了通用的交互式客户服务条件。第四组“Register”登记客户信息。输出配置的最后一个元素是一个与计算设备的扬声器相连接的硬件端口P3.1。当为这个端口分配一个代码P+时，一个正向脉冲串被发送到扬声器并听到一个通知音。端口P3.1是一个硬件终端，因此将其分类为一个面向硬件的输出。前四组都是面向软件的输出条件，因而被分类为虚拟计算输出。任何与一个硬件输出无关的输出条件被定义为一个虚拟输出。虚拟计算输出的含义包括任何用以产生数据的与终端无关的操作，信号或信息的显示或产生，初始化程序，重起动程序，启动软件计时器，或计数器或操作一个内部电路，如寄存器等。第12到17行是在状态表“Response”下配置的输出状态。当一个“Run”命令出现在一个输出配置状态中时，运行相应的组程序。当接收到一个“Continue”指令时，继续运行正在运行的程序或者如果该程序还没有开始运行或处于暂停状态，则该程序保持空闲状态。“x”标记表示不需要输出操作。利用这些描述，第12到17行的输出状态的操作是一目了然的。应当注意在一个程序中可能存在一个以上的输入或输出状态表。多个状态表简化了标格的结构并使编程作业更为容易。但是，应当注意在任何时候，每种输入状态表中只有一种配置状态被指定为有效。作为一种编程技巧，相互

联系的输入限定符和输出条件可被组合成一个状态表。还应当注意，如果需要，可将输入状态和输出状态组合成一种混合状态。

第 18 行启动操作“Path（s）”。当路径名被一种配置状态的任何限定符元素所引用时，每个路径定义一个或多个要执行的操作。第 19 行是一个名为“Start”的路径，在执行 Group 时，该路径是一个缺省起始路径。当首先起动该程序时，从所引用的所需操作中开始编程过程。开始操作“CheckSystem”检查输入显示驱动器、物理端口之类的本地计算系统的配置以得到运行程序可用的扬声器和系统资源。要估计的本地计算设备可用资源包括计算机时间、寄存器个数、可用存储器的存储量、存储器配置、所占用的计时器和计数器、可用的中断通道以及任何专用硬件电路配置。“CheckSystem”操作中应包括一个根据系统参数重构下载程序的过程。下一个步骤是显示第一页。在程序中仅将此操作定义为“DisplayFirstPage”。然后产生一“滴滴”声。“Hold2”表示所有的输出配置置于保持状态，如“Response”状态表所示。“Ready”指令始于输入状态表“FirstPage”的“Ready”状态。在“Ready”状态中，无论何时接收到图标“Catalog”、“Purchase”或“Service”的一个限定触发，都执行相应路径 20 到 22 中的一个。在上述每个路径中，显示指示该操作的窗口并启动促销程序。“BuySolicit”是一个用于请求销售公司产品的交互式程序。在路径“P_purchase”中，操作“Hold1”和“Hold3”限制除图标“Home”和“Quit”之外的来自本地用户终端的可允许响应。“GreyButton”是一个用以改变图标颜色的操作，它不适用于例如由状态命令“Hold1”所指定的图标“Catalog”、“Purchase”和“Service”。当执行第 23 行的路径“Bye”时，结束程序“Terminate”和该程序。一个意味着“程序组结束”的关键字“EOG”位于程序组的结尾，用于通知编译器该程序组到此结束。

应当看到所讨论的编程方法根据组成状态和路径而交互式地描述了程序操作。使用了由程序员分配的有意义的术语，如“Beep”、“CheckSystem”和“Terminate”。这个过程就象编写一篇用以精确描述程序所需操作的论文一样自然。

在此程序的编译过程中，很多程序员分配的术语仍然是未标识的，例如“CheckSystem”、“DisplayFirstPage”和“BuySolicit”。

这些程序员分配的标号都是计算设备不可执行的，除非它们与一个可执行程序链接。图 6B 中所示的下一个步骤就是需要进一步定义未标识标号的描述。使这些标号可执行的典型方法是将它们与一个外部可执行程序相链接或以一个来自程序库的程序来定义该标号。这就非常需要编译器提供一种能够识别所有程序员分配标号的功能。该标识最好在通过一个黑白打印机打印该程序时是可识别的。典型的最佳识别方法包括改变字母字格式和字型，如黑体、斜体或下划线。然后分析每种未标识标号的要求并选择一种最适宜的编程语言来编写一个程序以便提供所需的操作。

可以用任何语言，甚至另一个表格格式程序来编写支持程序。这些支持程序被“包括”在用于编译器的程序中以使所有程序集中在一起。图 6B 的第 18 行表示“CheckSystem”最好是一个用 Java 语言编写的前缀为“EJ”的程序，其中“E”表示它是一个被包括在内的外部程序. 操作“DisplayFirstPage”最好用 Visual Basic 语言编写。在第 19 行中，操作“BuySolicit”，一个征求购买的窗口最好来自一个局部或总程序库。第 23 行表示局部程序库的开始部分。第 24 行是一个精心设计的路径方程，它描述了征求客户的操作。这个操作包括执行一个用“C”编写的外部程序“CheckRecord”和一个用 Visual C++ 编写的程序“SolicitWindow”，以便通过人机对话的形式征求购买产品的客户。第 20 行中的元素“GreyButton”是指一个位于局部程序库中的路径，它由一个用于识别哪个图标具有“x”标记的“C”程序和另一个用 Visual Basic 编写的程序用于将这些图标变成灰暗颜色以表示这些图标不能被触发。

当建立了一个具有大量支持程序的程序库时，一个熟悉表格格式编程的程序员可通过选择和引用各种通用支持程序来开始编程作业。图 7 示出了构成一个包括窗口和收发器的程序时的引用的表示例。在这个引用表中的支持程序是必须遵循的，并因此需要编译器排除不用于所要构成的程序中的任何引用程序。

利用表格格式编程方法来管理用其它语言编写而成的支持程序需要更高的技术要求，例如在程序之间传送参数和使变量相等的方法。如果不同类型的程序是由不同的编译器翻译的，则要特别考虑对这些程序的正常管理。所属技术领域的技术人员能够理解所公开的表格格

式编程方法的优点并建立一个编译系统来完成所需的操作。例如，在引用一个特殊外部程序时，可分配预先定义的寄存器或存储器块来处理参数的传递。

从上述实施例中总结出表格格式编程的主要优点如下：

1. 通过编写描述性标号可比较容易地编写程序。

2. 程序的结构较好以致出现故障的机会很小。

3. 对于硬件终端来说，简化了与虚拟软件输出的组合和对高级程序流的影响。

4. 表格格式程序提供了清楚简洁的表达方式并易于为第三者读懂。它所提供的友好而清楚的表达对于进一步减少程序调试时间和缩减维护成本是非常重要的。

5. 表格格式编程方法使利用多种语言来构造一个程序变得更加简单。根据应用环境和每种语言的特点来选择语言。

6. 用表格格式编写的简明程序提供了较高的数据压缩率并使通过具有有限带宽和数据处理效率的通信通道而从一个远程的主机终端到一个本地计算设备的传输更加理想。

上述本发明的最佳实施例只是举出的例子，可以预想到各种修改、规定的改变、表格的重排、指令和关键字的分配都能达到相同的效果，从而这些修改和改进都应包括在后述的权利要求范围内。

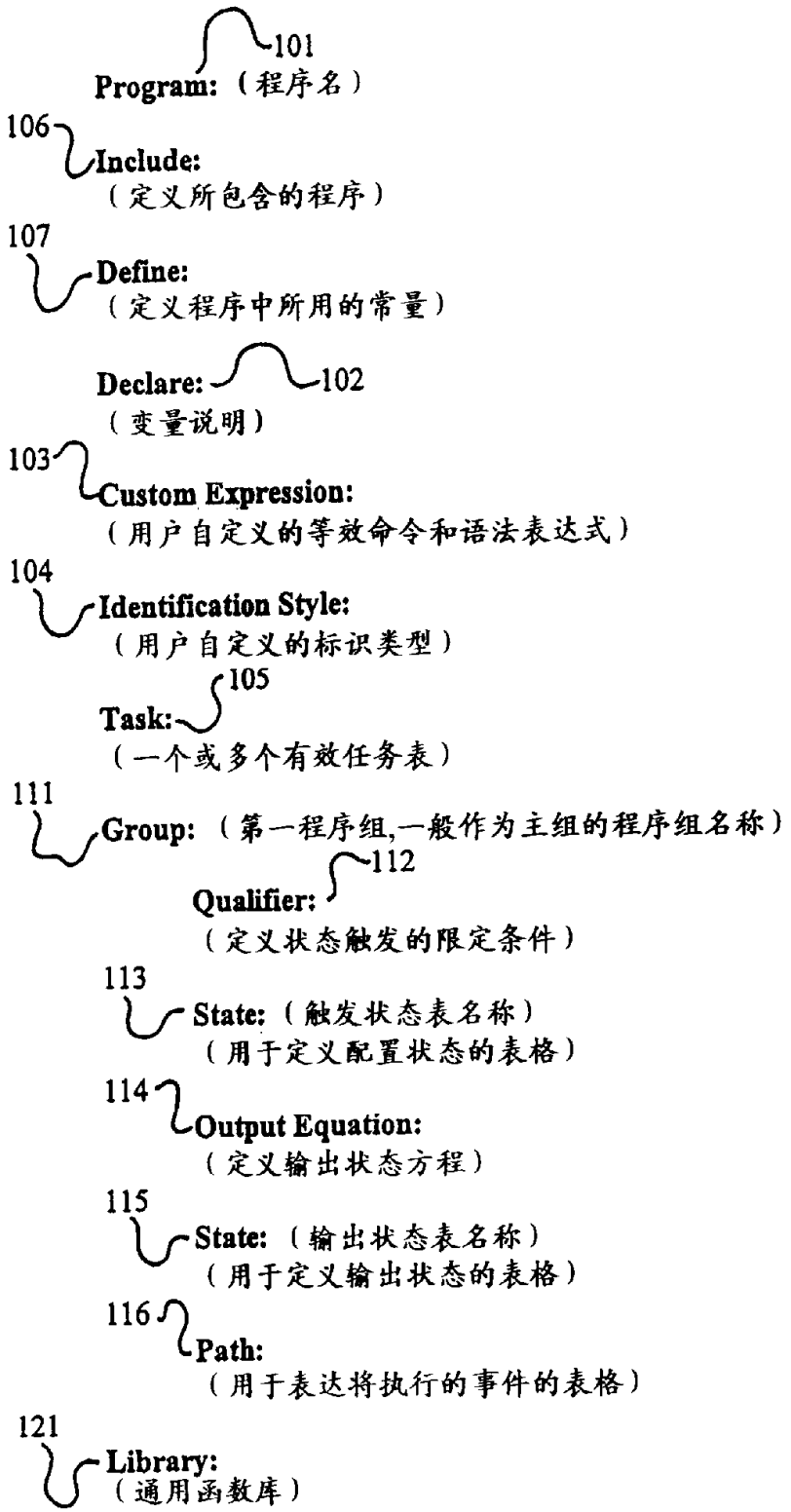
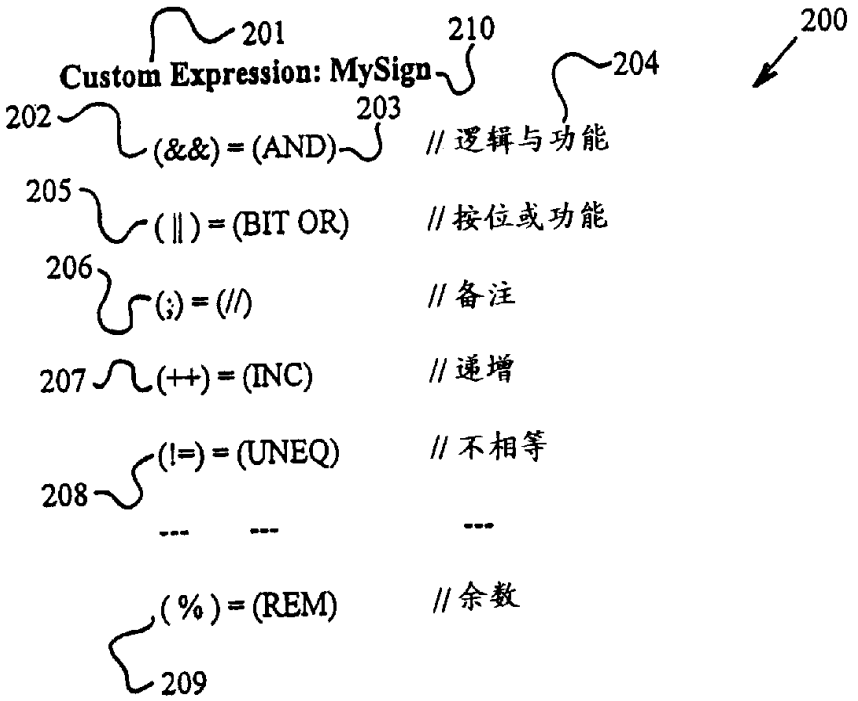

101
Program:（程序名）
106
Include:
（定义所包含的程序）
107
Define:
（定义程序中所用的常量）
Declare:
102
（变量说明）
103
Custom Expression:
（用户自定义的等效命令和语法表达式）
104
Identification Style:
（用户自定义的标识类型）
105
Task:
（一个或多个有效任务表）
111
Group:（第一程序组,一般作为主组的程序组名称）
112
Qualifier:
（定义状态触发的限定条件）
113
State:（触发状态表名称）
（用于定义配置状态的表格）
114
Output Equation:
（定义输出状态方程）
115
State:（输出状态表名称）
（用于定义输出状态的表格）
116
Path:
（用于表达将执行的事件的表格）
121
Library:
（通用函数库）

图 1


201
210
200
Custom Expression: MySign
204
202
203
(&&) = (AND)
// 逻辑与功能
205
(||) = (BIT OR)
// 按位或功能
206
(;) = (//)
// 备注
207
(++) = (INC)
// 递增
(!=) = (UNEQ)
// 不相等
208

(%) = (REM)
// 余数
209

图 2A

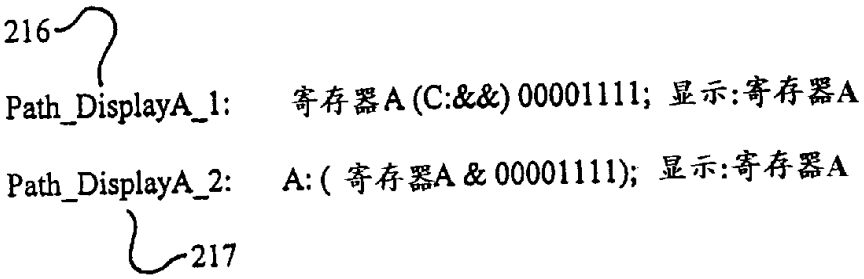

216
Path_DisplayA_1:
寄存器A (C:&&) 00001111; 显示:寄存器A
Path_DisplayA_2:
A: (寄存器A & 00001111); 显示:寄存器A
217

图 2B

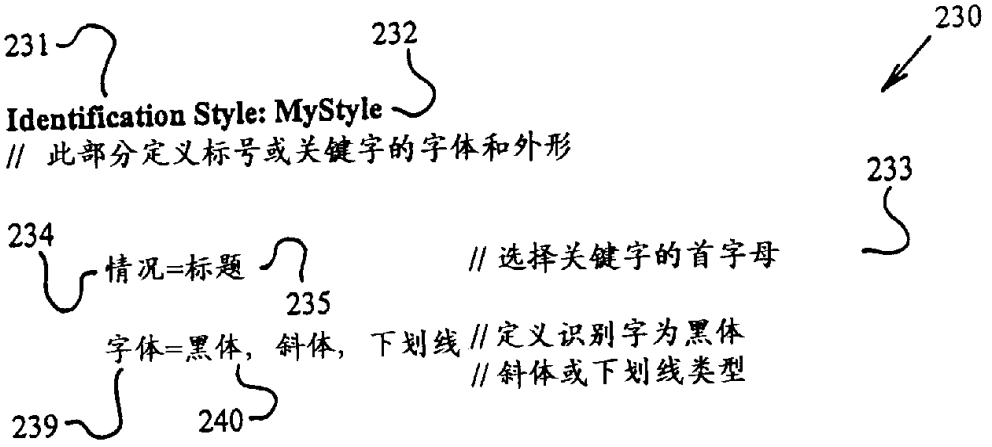
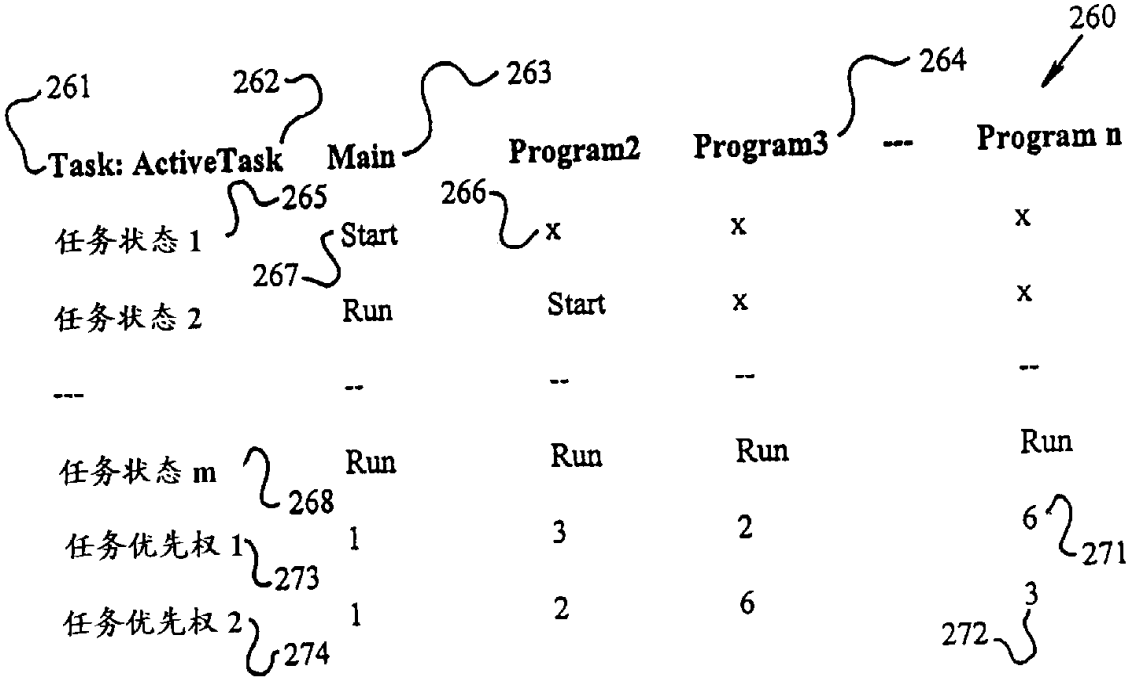

231
232
230
Identification Style: MyStyle
// 此部分定义标号或关键字的字体和外形
233
234
情况=标题
235
// 选择关键字的首字母
字体=黑体，斜体，下划线 // 定义识别字为黑体
// 斜体或下划线类型
239
240

图 3


260
261
262
263
264
Task: ActiveTask
Main
Program2
Program3

Program n
265
266
任务状态 1
Start
x
x
x
267
任务状态 2
Run
Start
x
x

--
--
--
--
任务状态 m
Run
Run
Run
Run
268
任务优先权 1
1
3
2
6
271
273
任务优先权 2
1
2
6
3
272
274

图 4

Task: Input	Keyboard	Mouse	GamePort
All:	Run	Run	Run
Normal:	Run	Run	x
Keyboard:	Run	x	x
Mouse:	x	Run	x
Game:	x	x	Start

Task:Ports	COM1	COM2	ParallelPort
All:	Run	Run	Run
SrMouse:	Start	x	x
ExModem:	x	Continue	x
Printer:	x	x	Continue

Task:Device	CDRom	HardDriveC	Floppy
ReadCD:	Start	Run	x
HDFullSpeed:	x	Run	x
RWFloppy:	x	Run	Run
ALL:	Run	Run	Run

// 可能的任务状态：开始，继续，运行，暂停，x

图 5A

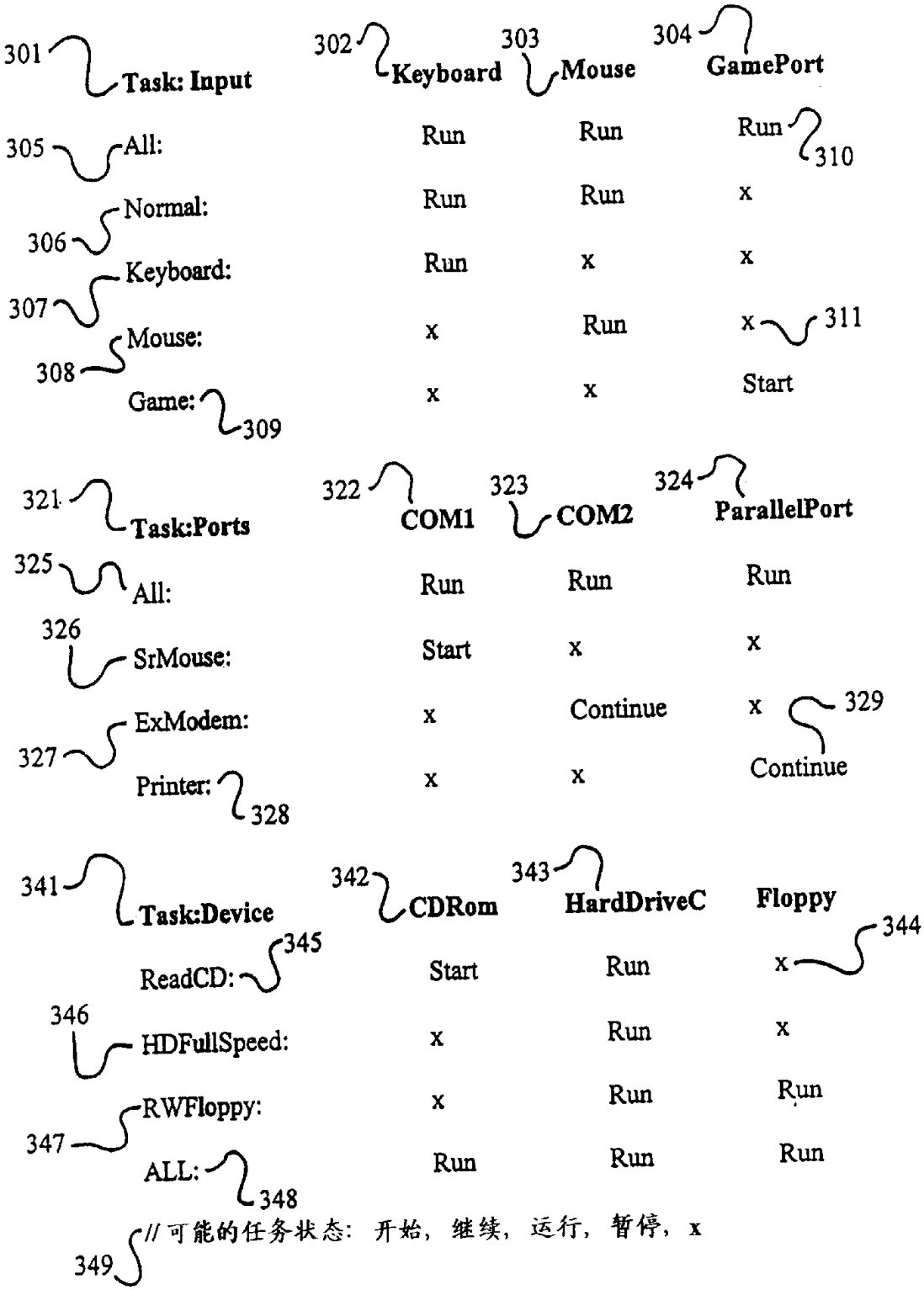

301 Task: Input
302 Keyboard
303 Mouse
304 GamePort
305 All: Run Run Run 310
306 Normal: Run Run x
307 Keyboard: Run x x
308 Mouse: x Run x 311
309 Game: x x Start
321 Task:Ports
322 COM1
323 COM2
324 ParallelPort
325 All: Run Run Run
326 SrMouse: Start x x
327 ExModem: x Continue x 329
Printer: x x Continue 328
341 Task:Device
342 CDRom
343 HardDriveC
Floppy
ReadCD: 345 Start Run x 344
346 HDFullSpeed: x Run x
RWFloppy: x Run Run
347 ALL: Run Run Run 348
349 // 可能的任务状态：开始，继续，运行，暂停，x

图 5B

```
1  Group: Main                    //WebSale 主程序组

2  Qualifier:
3  Catalog = Icon(1)
4  Purchase = Icon(2)
5  Service = Icon(3)
6  Home = Icon(4)
7  Quit = Icon(5)
```

8	State: FirstPage	Catalog	Purchase	Service	Home	Quit
9	Ready:	P_catalog	P_purchase	P_service	Start	Bye
10	Hold1:	x	x	x	Start	Bye
11	State: Response	Group:Info	Group:Order	Group:Service	Group:Register	P3.1
12	WindowCatalog:	Run	x	x	x	x
13	WindowPurchase:	x	Run	x	Run	x
14	WindowService:	x	x	Run	Run	x
15	Hold2:	x	x	x	x	x
16	Hold3:	x	Run	x	Run	x
17	Beep:	Continue	Continue	Continue	Continue	P+

```
18  Path:
19  Start:          CheckSystem; DisplayFirstPage; Beep; Hold 2; Ready; END
20  P_catalog:      WindowCatalog; BuySolicit; END
21  P_purchase:     WindowPurchase; Hold1; Hold3; GrayButton; END
22  P_service:      WindowService; BuySolicit; END
23  Bye:            Terminate; END

24  EOG             //表示程序组结束的关键字

//注意：行号是为了便于描述，在实际中不需要
//编程：状态和路径方程不必按照顺序排列
```

图 6A

1	**Group:** Main	//WebSale 主程序组				
2	**Qualifier:**					
3	Catalog = Icon(1)					
4	Purchase = Icon(2)					
5	Service = Icon(3)					
6	Home = Icon(5)					
7	Quit = Icon(4)					
8	**State:** FirstPage	Catalog	Purchase	Service	Home	Quit
9	Ready:	P_catalog	P_purchase	P_service	Start	Bye
10	Hold1:	x	x	x	Start	Bye
11	**State:** Response	Group:Info	Group:Order	Group:Service	Group:Register	P3.1
12	WindowCatalog:	Run	x	x	x	x
13	WindowPurchase:	x	Run	x	Run	x
14	WindowService:	x	x	Run	Run	x
15	Hold2:	x	x	x	x	x
16	Hold3:	x	Run	x	Run	x
17	Beep:	Continue	Continue	Continue	Continue	P+
18	**Path:**					
19	Start:	EJ_CheckSystem; EVB_DisplayFirstPage; Beep; Hold 2; Ready; END				
20	P_catalog:	WindowCatalog; Lbry:BuySolicit; END				
21	P_purchase:	WindowPurchase; Hold1; Hold3; Lbry:GrayButton; END				
22	P_service:	WindowService; Lbry:BuySolicit; END				
23	Bye:	EJ_Terminate; END				
24	**Library: Local**					
25	BuySolicit:	EC_CheckRecord; EVC_SolicitWindow; END				
26	GrayButton:	EC_CheckX; EVB_X_Is_Gray; END				
27	EOG	//表示程序组结束的关键字				

//注意：行号是为了便于描述，在实际中不需要
//编程：状态和路径方程不必按照顺序排列

图 6B

```
Include:
ReadIR                  //从远程控制读取信号
ReadPanel               //从控制面板读取信号
ReceiveFromLine         //从电缆线输入数据
SendToLine              //向电缆线传输数据
CompressBuffer          //在向电缆线发送之前压缩数据
DecompressBuffer        //对数据文件进行解压缩
CompareTime             //比较寄存器时间和实时时钟
GenIcon                 //在电视屏幕上生成图标
GenWindow               //创建一个新窗口
InputCursor             //输入并译码光标的移动
OutputCursor            //更新光标在电视屏幕上的位置
DisplayData             //在电视屏幕上显示数据
DisplayPicture          //在电视屏幕上显示图形文件
.....                   //.....
.....                   //.....
ErrorMessage            //对错误代码进行译码并传输错误消息
Help                    //显示帮助屏幕
```

图 7

```
Include:
EA_ReadIR                 //从远程控制读取信号
EE_ReadPanel              //从控制面板读取信号
EA_ReceiveFromLine        //从电缆线输入数据
EA_SendToLine             //向电缆线传输数据
EC_CompressBuffer         //在向电缆线发送之前压缩数据
EC_DecompressBuffer       //对数据文件进行解压缩
EA_CompareTime            //比较寄存器时间和实时时钟
EJ_GenIcon                //在电视屏幕上生成图标
EJ_GenWindow              //创建一个新窗口
EVC_InputCursor           //输入并译码光标的移动
EVB_OutputCursor          //更新光标在电视屏幕上的位置
EVB_DisplayData           //在电视屏幕上显示数据
EVB_DisplayPicture        //在电视屏幕上显示图形文件
.....                     //.....
.....                     //.....
EA_ErrorMessage           //对错误代码进行译码并传输错误消息 sage
EVB_Help                  //显示帮助屏幕
```

图 8

功能	表格格式指令
指令分隔符	
标识指令的结束	;
指令的继续	
标识后续行	&(在行首)
Unary Operators	
方向	VALUE (地址)
地址	ADDRESS (值)
页	-
逻辑非	NOT (表达式)
1的补码	COMPLEMENT(表达式)
递增	INCREMENT
递减	DECREMENT
大小	SIZEOF(表达式)
移位操作符	
左移	BIT SHIFT LEFT (操作数,n)
右移位	BIT SHIFT RIGHT(操作数,n)
关系操作符	
小于	<
大于	>
小于等于	<= or =<
大于等于	>= or =>
逻辑操作符	
按位与	BIT AND
逻辑与	AND
异或	XOR
按位或	BIT OR
逻辑或	OR
条件分支	表达式？:[正确语句] / [错误语句]
注解	
	//(至文件末端)

图 9

输入状态:

触发端子:	TG1	TG2	TG3	TG4	TG5	TG6
State0:	R:Path1	R:Path2	R:Path3	R:Path4	X	X
State1:	F:Path11	R:Path2	R:Path3	R:Path4	X	X
State2:	R:Path1	F:Path11	R:Path3	R:Path4	X	X
State3:	R:Path1	R:Path2	F:Path11	R:Path4	X	X
State4:	R:Path1	R:Path2	R:Path3	F:Path11	X	X

;

Path:			
Path 1:	State1	Sound1	Path1
Path 2:	State2	Sound2	Path2
Path 3:	State3	Sound3	Path3
Path 4:	State4	Sound4	Path4
Path11:	State0	结束	

图 10 现有技术

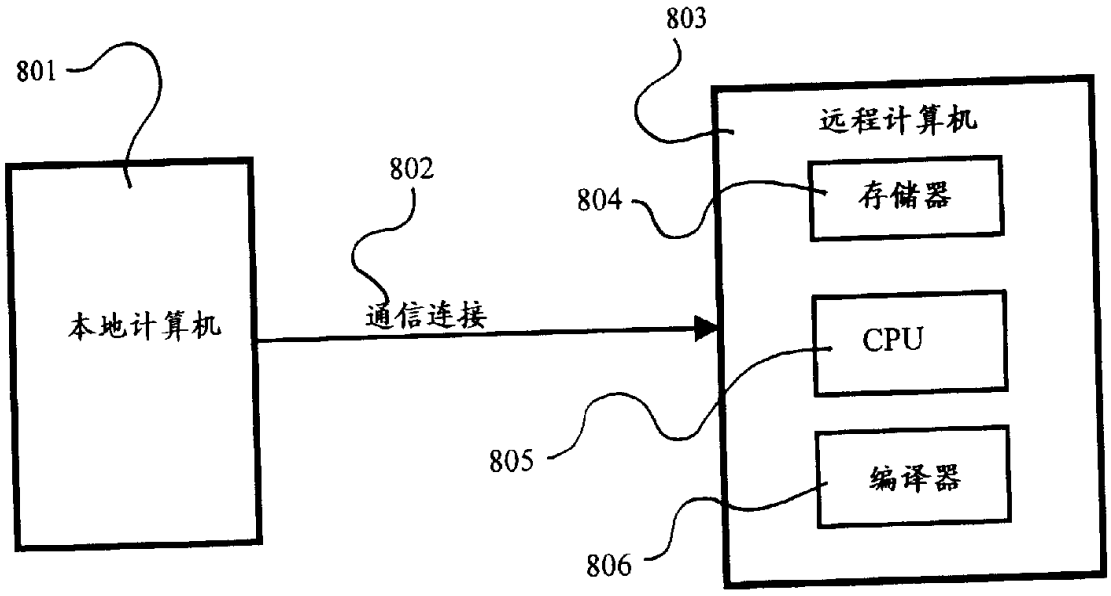


图 11