(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112835564 A
(43)申请公布日 2021.05.25

(21)申请号 202110111200.4

(22)申请日 2021.01.27

(71)申请人 北京海博思创科技股份有限公司
地址 100094 北京市海淀区丰豪东路9号院2号楼12层3单元1201

(72)发明人 姜科 杨洸 林强

(74)专利代理机构 北京同立钧成知识产权代理有限公司 11205
代理人 朱颖 黄健

(51)Int.Cl.
G06F 8/30(2018.01)

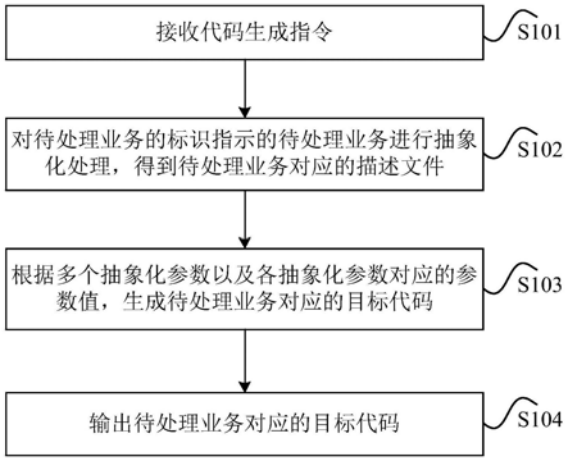
权利要求书2页 说明书17页 附图3页

(54)发明名称

代码生成方法和装置

(57)摘要

本申请实施例提供了一种代码生成方法和装置，在生成代码时，先接收代码生成指令；其中，代码生成指令中包括待处理业务的标识；对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；并根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码；并输出待处理业务对应的目标代码。可以看出，在生成代码时，无需技术人员人工进行手动编写，实现了自动化生成代码，从而有效地提高了代码生成的效率。



CN 112835564 A

1.一种代码生成方法,其特征在于,包括:

接收代码生成指令;其中,所述代码生成指令中包括待处理业务的标识;

对所述待处理业务的标识指示的待处理业务进行抽象化处理,得到所述待处理业务对应的描述文件;其中,所述描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值;

根据所述多个抽象化参数以及所述各抽象化参数对应的参数值,生成所述待处理业务对应的目标代码;

输出所述待处理业务对应的目标代码。

2.根据权利要求1所述的方法,其特征在于,所述根据所述多个抽象化参数以及所述各抽象化参数对应的参数值,生成所述待处理业务对应的目标代码,包括:

根据所述多个抽象化参数,生成所述待处理业务对应的代码生成规则;

根据所述代码生成规则生成所述代码生成规则对应的代码生成工具;

根据所述描述文件和所述代码生成工具生成所述待处理业务对应的目标代码。

3.根据权利要求2所述的方法,其特征在于,所述根据所述描述文件和所述代码生成工具生成所述待处理业务对应的目标代码,包括:

将所述描述文件导入至所述代码生成工具中,生成所述待处理业务对应的目标代码。

4.根据权利要求2所述的方法,其特征在于,所述根据所述代码生成规则生成所述代码生成规则对应的代码生成工具,包括:

根据所述代码生成规则,确定所述待处理业务对应的目标代码的架构;

根据所述待处理业务对应的目标代码的架构,生成所述代码生成规则对应的代码生成工具。

5.根据权利要求2所述的方法,其特征在于,所述描述文件为表格,所述根据所述多个抽象化参数,生成所述待处理业务对应的代码生成规则,包括:

将所述多个抽象化参数中,处于所述表格的表头位置的抽象化参数确定为多个目标抽象化参数;

根据所述多个目标抽象化参数,生成所述待处理业务对应的代码生成规则。

6.根据权利要求5所述的方法,其特征在于,所述方法还包括:

确定所述待处理业务的类型,所述待处理业务对应的目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的实现方式;

根据所述待处理业务的类型,所述目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的运行规则,对所述代码生成规则进行更新,得到新的代码生成规则。

7.根据权利要求1-6任一项所述的方法,其特征在于,所述对所述待处理业务的标识指示的待处理业务进行抽象化处理,包括:

对所述待处理业务进行解析,得到所述待处理业务对应的状态信息、条件信息以及执行动作信息;

基于所述状态信息、所述条件信息以及所述执行动作信息,对所述待处理业务进行抽象化处理。

8.一种代码生成装置,其特征在于,包括:

接收单元,用于接收代码生成指令;其中,所述代码生成指令中包括待处理业务的标

识；

处理单元，用于对所述待处理业务的标识指示的待处理业务进行抽象化处理，得到所述待处理业务对应的描述文件；其中，所述描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；

生成单元，用于根据所述多个抽象化参数以及所述各抽象化参数对应的参数值，生成所述待处理业务对应的目标代码；

输出单元，用于输出所述待处理业务对应的目标代码。

9.一种代码生成装置，其特征在于，包括存储器和处理器；其中，

所述存储器，用于存储计算机程序；

所述处理器，用于读取所述存储器存储的计算机程序，并根据所述存储器中的计算机程序执行上述权利要求1-7任一项所述的一种代码生成方法。

10.一种计算机可读存储介质，其特征在于，所述计算机可读存储介质中存储有计算机执行指令，当处理器执行所述计算机执行指令时，实现上述权利要求1-7任一项所述的一种代码生成方法。

11.一种计算机程序产品，包括计算机程序，其特征在于，该计算机程序被处理器执行时，实现上述权利要求1-7任一项所述的一种代码生成方法。

代码生成方法和装置

技术领域

[0001] 本申请涉及软件工程技术领域，尤其涉及一种代码生成方法和装置。

背景技术

[0002] 软件的开发和运行离不开代码的编写。在软件的开发过程中，通常需要技术人员人为的编写不同的代码，以实现软件的各种功能。

[0003] 但是，在技术人员对代码进行人工编写时，需要耗费大量的时间，使得代码生成的效率较低。

发明内容

[0004] 本申请实施例提供了一种代码生成方法和装置，在生成代码时，无需技术人员人工进行手动编写，实现了自动化生成代码，从而有效地提高了代码生成的效率。

[0005] 第一方面，本申请实施例提供了一种代码生成方法，所述代码生成方法包括：

[0006] 接收代码生成指令；其中，所述代码生成指令中包括待处理业务的标识。

[0007] 对所述待处理业务的标识指示的待处理业务进行抽象化处理，得到所述待处理业务对应的描述文件；其中，所述描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值。

[0008] 根据所述多个抽象化参数以及所述各抽象化参数对应的参数值，生成所述待处理业务对应的目标代码。

[0009] 输出所述待处理业务对应的目标代码。

[0010] 在一种可能的实现方式中，所述根据所述多个抽象化参数以及所述各抽象化参数对应的参数值，生成所述待处理业务对应的代码，包括：

[0011] 根据所述多个抽象化参数，生成所述待处理业务对应的代码生成规则。

[0012] 根据所述代码生成规则生成所述代码生成规则对应的代码生成工具；

[0013] 根据所述描述文件和所述代码生成工具生成所述待处理业务对应的目标代码。

[0014] 在一种可能的实现方式中，所述根据所述描述文件和所述代码生成工具生成所述待处理业务对应的目标代码，包括：

[0015] 将所述描述文件导入至所述代码生成工具中，生成所述待处理业务对应的目标代码。

[0016] 在一种可能的实现方式中，所述根据所述代码生成规则生成所述代码生成规则对应的代码生成工具，包括：

[0017] 根据所述代码生成规则，确定所述待处理业务对应的目标代码的架构。

[0018] 根据所述待处理业务对应的目标代码的架构，生成所述代码生成规则对应的代码生成工具。

[0019] 在一种可能的实现方式中，所述描述文件为表格，所述根据所述多个抽象化参数，生成所述待处理业务对应的代码生成规则，包括：

[0020] 将所述多个抽象化参数中，处于所述表格的表头位置的抽象化参数确定为多个目标抽象化参数。

[0021] 根据所述多个目标抽象化参数，生成所述待处理业务对应的代码生成规则。

[0022] 在一种可能的实现方式中，所述方法还包括：

[0023] 确定所述待处理业务的类型，所述待处理业务对应的目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的实现方式。

[0024] 根据所述待处理业务的类型，所述目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的运行规则，对所述代码生成规则进行更新，得到新的代码生成规则。

[0025] 在一种可能的实现方式中，所述对所述待处理业务的标识指示的待处理业务进行抽象化处理，包括：

[0026] 对所述待处理业务进行解析，得到所述待处理业务对应的状态信息、条件信息以及执行动作信息。

[0027] 基于所述状态信息、所述条件信息以及所述执行动作信息，对所述待处理业务进行抽象化处理。

[0028] 第二方面，本申请实施例提供了一种代码生成装置，所述代码生成装置包括：

[0029] 接收单元，用于接收代码生成指令；其中，所述代码生成指令中包括待处理业务的标识。

[0030] 处理单元，用于对所述待处理业务的标识指示的待处理业务进行抽象化处理，得到所述待处理业务对应的描述文件；其中，所述描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值。

[0031] 生成单元，用于根据所述多个抽象化参数以及所述各抽象化参数对应的参数值，生成所述待处理业务对应的目标代码。

[0032] 输出单元，用于输出所述待处理业务对应的目标代码。

[0033] 在一种可能的实现方式中，所述生成单元，具体用于根据所述多个抽象化参数，生成所述待处理业务对应的代码生成规则；根据所述代码生成规则生成所述代码生成规则对应的代码生成工具；根据所述描述文件和所述代码生成工具生成所述待处理业务对应的目标代码。

[0034] 在一种可能的实现方式中，所述生成单元，具体用于将所述描述文件导入至所述代码生成工具中，生成所述待处理业务对应的目标代码。

[0035] 在一种可能的实现方式中，所述生成单元，具体用于根据所述代码生成规则，确定所述待处理业务对应的目标代码的架构；并根据所述待处理业务对应的目标代码的架构，生成所述代码生成规则对应的代码生成工具。

[0036] 在一种可能的实现方式中，所述生成单元，具体用于将所述多个抽象化参数中，处于所述表格的表头位置的抽象化参数确定为多个目标抽象化参数；根据所述多个目标抽象化参数，生成所述待处理业务对应的代码生成规则。

[0037] 在一种可能的实现方式中，所述生成单元，具体用于确定所述待处理业务的类型，所述待处理业务对应的目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的实现方式；根据所述待处理业务的类型，所述目标代码的函数类型、所述目标代码的文件类型以及所述目标代码的运行规则，对所述代码生成规则进行更新，得到新的代码生成规

则。

[0038] 在一种可能的实现方式中，所述处理单元，具体用于对所述待处理业务进行解析，得到所述待处理业务对应的状态信息、条件信息以及执行动作信息；基于所述状态信息、所述条件信息以及所述执行动作信息，对所述待处理业务进行抽象化处理。

[0039] 第三方面，本申请实施例还提供了一种代码生成装置，该代码生成装置可以包括存储器和处理器；其中，

[0040] 所述存储器，用于存储计算机程序。

[0041] 所述处理器，用于读取所述存储器存储的计算机程序，并根据所述存储器中的计算机程序执行上述第一方面任一种可能的实现方式中所述的代码生成方法。

[0042] 第四方面，本申请实施例还提供了一种计算机可读存储介质，所述计算机可读存储介质中存储有计算机执行指令，当处理器执行所述计算机执行指令时，实现上述第一方面任一种可能的实现方式中所述的代码生成方法。

[0043] 第五方面，本申请实施例还提供了一种计算机程序产品，包括计算机程序，该计算机程序被处理器执行时，实现上述第一方面任一种可能的实现方式中所述的代码生成方法。

[0044] 由此可见，本申请实施例提供的代码生成方法和装置，在生成代码时，先接收代码生成指令；其中，代码生成指令中包括待处理业务的标识；对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；并根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码；并输出待处理业务对应的目标代码。可以看出，在生成代码时，无需技术人员人工进行手动编写，实现了自动化生成代码，从而有效地提高了代码生成的效率。

附图说明

[0045] 此处的附图被并入说明书中并构成本说明书的一部分，示出了符合本公开的实施例，并与说明书一起用于解释本公开的原理。

[0046] 图1为本申请实施例提供的一种代码生成方法的流程示意图；

[0047] 图2为本申请实施例提供的一种生成待处理业务对应的目标代码的流程示意图；

[0048] 图3为本申请实施例提供的另一种生成代码方法的流程示意图；

[0049] 图4为本申请实施例提供的一种代码运行的流程示意图；

[0050] 图5为本申请实施例提供的一种代码生成装置的结构示意图；

[0051] 图6为本申请实施例提供的另一种代码生成装置的结构示意图。

[0052] 通过上述附图，已示出本公开明确的实施例，后文中将有更详细的描述。这些附图和文字描述并不是为了通过任何方式限制本公开构思的范围，而是通过参考特定实施例为本领域技术人员说明本公开的概念。

具体实施方式

[0053] 这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施例

中所描述的实施方式并不代表与本公开相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本公开的一些方面相一致的装置和方法的例子。

[0054] 在本申请的实施例中，“至少一个”是指一个或者多个，“多个”是指两个或两个以上。“和/或”，描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况，其中A，B可以是单数或者复数。在本申请的文字描述中，字符“/”一般表示前后关联对象是一种“或”的关系。

[0055] 本申请实施例提供的技术方案可以应用于代码生成场景中。软件功能的实现离不开代码，例如，由于生产厂商以及车型的不同，电动汽车对于接触器的控制方法不同，可以通过电池管理系统(Battery Management System，简称BMS)进行控制，也可以通过整车的其它电控器件进行控制。对于通过BMS软件控制的电动汽车，由于整车需要采用的不同技术方案，因此，软件开发人员往往需要编写不同的代码，实现BMS软件的不同的整体调度和控制时序，从而达到适配整车不同需求的目的。

[0056] 现有技术中，技术人员根据接收到的指令，确定待开发的软件所需要实现的功能，并根据待开发的软件所需要实现的功能人工编写相应的代码，使得软件能够实现各种不同的功能。但是，人工编写代码需要耗费大量的时间，而且在代码编写的过程中，可能出现编写错误的情况，降低了生成代码的效率。

[0057] 为了解决生成代码的效率低的问题，可以考虑自动化生成代码。本申请实施例提供了一种代码生成方法，先接收代码生成指令；其中，代码生成指令中包括待处理业务的标识；对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；并根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码；输出待处理业务对应的目标代码。

[0058] 由此可见，本申请实施例中，在生成代码时，先接收代码生成指令；其中，代码生成指令中包括待处理业务的标识；对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；并根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码。可以看出，在生成代码时，无需技术人员人工进行手动编写，实现了自动化生成代码，从而有效地提高了代码生成的效率。

[0059] 下面，将通过具体的实施例对本申请提供的代码生成方法进行详细地说明。可以理解的是，下面这几个具体的实施例可以相互结合，对于相同或相似的概念或过程可能在某些实施例不再赘述。

[0060] 图1为本申请实施例提供的一种代码生成方法的流程示意图。该代码生成方法可以由软件和/或硬件装置执行，例如，该硬件装置可以为代码生成装置，该代码生成装置可以为终端或者终端中的处理芯片。示例的，请参见图1所示，该代码生成方法可以包括：

[0061] S101、接收代码生成指令。

[0062] 其中，代码生成指令中包括待处理业务的标识。

[0063] 示例的，代码生成指令有多种形式，可以为文字形式，也可以为特征的符号，其主要用于标识待处理业务。本申请实施例对于代码生成指令的形式，以及待处理业务标识的形式不做任何限制。

[0064] S102、对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件。

[0065] 示例的，在对待处理业务的标识指示的待处理业务进行抽象化处理时，可以先对待处理业务进行解析，得到待处理业务对应的状态信息、条件信息以及执行作用信息；并基于状态信息、条件信息以及执行动作信息，对待处理业务进行抽象化处理。

[0066] 其中，状态信息可以包括多个不同状态，条件信息可以包括多个不同的条件，执行动作信息可以包括多个不同的执行动作，且待处理业务对应的状态信息可以包括多个子状态。例如，状态信息可以包括3个子状态，分别为初始化子状态、运行子状态以及退出子状态。条件信息可以为各个不同的状态之间的跳转信息，执行动作信息可以为每个子状态需要执行的动作。可以理解的是，状态信息、条件信息以及执行作用信息均可用特殊的字符进行表示，本申请实施例对于具体的表示方式不做任何限制。

[0067] 在本申请实施例，通过将待处理解析为状态信息、条件信息以及执行动作，并进行抽象化处理，可以实现将待处理业务转化为可以被计算机识别的语言，这样可以有效地提升代码生成的成功率。

[0068] 在对待处理业务的标识指示的待处理业务进行抽象化处理时，可以得到待处理业务对应的描述文件，该描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值。其中，抽象化参数可以为对待处理业务进行解析得到的状态信息、条件信息、执行动作信息，以及其对应的具体信息和数量，抽象化参数对应的参数则可以为状态信息、条件信息、执行动作信息分别对应的具体数值。本申请实施例对于抽象化参数以及各抽象化参数对应的参数值的具体内容不做任何限定。其中，描述文件的形式可以为多种类型，例如，描述文件可以为EXCEL表格，本申请实施例对于描述文件的具体类型不做任何限定。

[0069] 在对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件后，就可以根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码，即执行下述S103：

[0070] S103、根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码。

[0071] 在根据多个抽象化参数以及各参数对应的参数值，生成待处理业务对应的目标代码时，可以直接获取抽象化参数以及各抽象化参数对应的参数值，也可以通过获取描述文件，读取描述中的抽象化参数以及各抽象化参数对应的参数值，具体的，本申请实施例不做任何限制。

[0072] 在分别获取到多个抽象化参数以及各抽象化参数对应的参数值后，就可以根据该根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码，并输出该目标代码。

[0073] S104、输出待处理业务对应的目标代码。

[0074] 示例的，在输出待处理业务对应的目标代码时，若设置目标代码的文件存储类型，则将待处理业务对应的目标代码存储在对应的文件存储类型的文件内，输出含有目标代码的文件；若未设置目标代码的文件存储类型，则将待处理业务对应的目标代码存储在默认文件存储类型的文件内，输出默认文件存储类型的文件，其中，默认文件存储类型为用户上一次设置的文件存储类型，具体可以根据实际需要进行设置。

[0075] 由此可见，本申请实施例提供的代码生成方法，在生成代码时，先接收代码生成指令；其中，代码生成指令中包括待处理业务的标识；对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值；并根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码。可以看出，在生成代码时，无需技术人员人工进行手动编写，实现了自动化生成代码，从而有效地提高了代码生成的效率。

[0076] 基于上述图1所示的实施例，为了便于理解在本申请实施例中，如何根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码，下面，将通过图2所示的实施例，详细描述在本申请实施例中，如何根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码。

[0077] 图2为本申请实施例提供的一种生成待处理业务对应的目标代码的流程示意图。该生成待处理业务对应的目标代码的方法同样可以由软件和/或硬件装置执行。示例的，请参见图2所示，该生成待处理业务对应的目标代码的方法可以包括：

[0078] S201、根据多个抽象化参数，生成待处理业务对应的代码生成规则。

[0079] 示例的，在根据多个抽象化参数，生成待处理业务对应的代码生成规则时，可以先对多个抽象化参数进行处理，确定代码生成规则所需要的抽象化参数，即目标抽象化参数。示例的，若描述文件为表格，则将多个抽象化参数中，处于表格的表头位置的抽象化参数确定为对个目标抽象化参数；并根据多个目标抽象化参数，生成待处理业务对应的代码生成规则。示例的，若描述文件为文本，则可以通过读取文本的初始位置即可获取多个目标抽象化参数，具体可以根据实际需要进行设置。

[0080] 根据上述实施例所述，当抽象化参数为待处理业务对应的代码中所包含的状态信息、条件信息以及执行动作信息的具体信息和数量时，对应的，代码生成规则为状态信息、条件信息以及执行动作信息的数量信息。例如，若抽象化参数为2个状态信息、4个条件信息以及5个执行动作信息，则对应的代码生成规则为生成的代码中需要包括2个状态、4个跳转条件以及5个执行动作。

[0081] 在本申请实施例中，通过确定多个目标抽象化参数，并生成对应的代码生成规则，能够避免除目标抽象化参数外的其它抽象化参数对目标代码造成影响，该生成规则可以用于辅助生成对应的目标代码。

[0082] 此外，为了进一步完善该生成规则，还可以通过确定待处理业务的类型，待处理业务对应的目标代码的函数类型、目标代码的文件类型以及目标代码的实现方式；并根据待处理业务的类型，目标代码的函数类型、目标代码的文件类型以及目标代码的运行规则，对代码生成规则进行更新，得到新的代码生成规则，这样可以进一步提高生成规则的完整性，从而更好地辅助生成对应的目标代码。

[0083] 其中，待处理业务的类型、待处理业务对应的目标代码的函数类型、目标代码的文件类型以及目标代码的实现方式可以根据具体的业务进行确定。例如，待处理业务对应的目标代码的函数类型可以为void xxx_step(void)的形式，目标代码的文件类型可以为xxx.c和xxx.h的形式，目标代码的实现方式可以为switch……case……的形式，本申请实施例对此不做任何限定。

[0084] 可以看出，在本申请实施例中，通过生成新的代码生成规则，其目的在于：可以使

得生成目标代码的风格统一，避免了多个技术人员对于代码进行分工编写而造成的代码风格不统一以及出现逻辑错误的问题，提高了代码生成的效率。

[0085] S202、根据代码生成规则生成代码生成规则对应的代码生成工具。

[0086] 示例的，在根据代码生成规则生成代码生成规则对应的代码生成工具时，可以根据代码生成规则，确定待处理业务对应的目标代码的架构；根据待处理业务对应的目标代码的架构，生成代码生成规则对应的代码生成工具。其中，待处理业务对应的目标代码的架构可以包括生成的目标代码的重复部分，也可以包括其他部分，本申请实施例对此不做任何限定。

[0087] 进一步地，在根据待处理业务对应的目标代码的架构，生成代码生成规则对应的代码生成工具时，可以根据目标代码的架构编写对应的代码生成工具，也可以通过其他方式生成对应的代码生成工具，本申请实施例对此不做任何限定。

[0088] 在本申请实施例中，通过代码生成规则确定待处理业务对应的目标代码的架构，并生成对应的代码生成工具，使得能够通过代码生成工具生成对应的目标代码，提高了生成代码的准确度和效率。

[0089] S203、根据描述文件和代码生成工具生成待处理业务对应的目标代码。

[0090] 示例的，在根据描述文件和代码生成工具生成待处理业务对应的目标代码时，可以将描述文件导入至代码生成工具中，生成待处理业务对应的目标代码。

[0091] 其中，代码生成工具可以先根据描述文件中的多个抽象画参数对应的参数值，确定目标代码需要的多个目标抽象化参数值，并将多个目标抽象化参数值与待处理业务对应的目标代码的架构进行拼凑，或者将多个目标抽象化参数值补充至目标代码的架构中，从而生成待处理业务对应的目标代码。具体的生成方式可根据实际情况进行设置。通过将描述文件导入至代码生成工具中，使得代码工具根据描述文件中的抽象化参数对应的参数值，生成待处理业务对应的目标代码，避免了人工编写代码出现错误的问题，提高了代码生成的效率。

[0092] 综上所述，本申请实施例提供的代码生成方法，在根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码时，先根据多个抽象化参数，生成待处理业务对应的代码生成规则；并根据代码生成规则生成对应的代码生成工具；根据描述文件和代码生成工具生成待处理业务对应的目标代码，这样能够避免人工手动编写代码造成的耗时长以及错误率高的问题，提高了代码生成的效率。

[0093] 为了便于理解本申请实施例提供的代码生成方法，下面，将以该方法在实际中的具体应用为例，对本申请实施例提供的技术方案进行详细的描述。可参见图3所示，图3为本申请实施例提供的另一种生成代码方法的流程示意图。

[0094] 在代码的生成中可以采用调度器进行，能够保证代码的生成过程各个软件和\或硬件装置的分工明确，从而提升代码生成的效率。图3中的调度定义，即为在接收到代码生成指令时，可以确定待处理业务的类型以及待处理业务的标识，通过对待处理业务进行解析，确定待处理业务对应的状态信息、条件信息以及执行动作信息。假设待处理业务对应的状态通过S表示，条件通过C表示，执行动作通过A表示。

[0095] 示例的，状态S可以包括初始化状态S0和多个运行状态S1～Sn，条件C可以为每个状态S之间的状态跳转条件，Cx_y表示状态Sx向状态Sy的跳转条件，例如，C1_2表示状态S1

向状态S2的跳转条件。其中，每一个状态S均包括3个子状态，分别为初始化子状态en、运行子状态du以及退出子状态ex。执行动作A为每个子状态执行的动作，其中，执行初始化子状态en的动作可以为Aen_1～Aen_n，表示依次执行初始化子状态1至子状态n的动作；执行运行子状态du的动作可以为Adu_1～Adu_n，表示依次执行运行子状态1至子状态n的动作；执行退出子状态ex的动作可以为Aex_1～Aex_n，表示依次执行退出子状态1至子状态n的动作。

[0096] 可以理解的是，上述状态跳转条件为实际代码中对应的条件，执行动作为基础软件层所提供的函数接口。其中，状态、条件和执行动作之间的关系可以参见图4所示，图4为本申请实施例提供的一种代码运行的流程示意图。

[0097] 根据图4所示，在代码运行过程中，先依次执行初始化状态S0中的初始化子状态en的动作，即依次执行初始化子状态1至初始化子状态n的动作。在完成子状态的初始化后，依次执行初始化状态S0中的运行子状态du的动作，即依次执行运行子状态1至运行子状态n的动作，且该依次运行n个子状态的动作是循环进行的，直至接收到跳转条件C＝C0_x。在接收到跳转条件后，停止依次执行初始化状态S0中的运行子状态的动作，变为依次执行初始化状态S0中的退出子状态ex的动作，即依次执行退出子状态1至退出子状态n的动作。其中，跳转条件可以为从初始化状态S0跳转至状态S1，也可以为从初始化状态S0跳转至状态S1，本申请实施例对于具体的跳转条件不做任何限定。

[0098] 在执行完初始化状态S0中的多个子状态的退出动作后，根据具体的跳转条件进行状态的跳转。若跳转条件为C＝C0_1，则由初始化状态S0跳转至状态S1，按照状态S1中的代码运行方式进行运行，并根据接收到的跳转条件C＝C1_2，由状态S1跳转至状态S2，按照状态S2中的代码运行方式进行运行。若跳转条件为C＝C0_2，则由初始化状态S0跳转至状态S2，按照状态S2中的代码运行方式进行运行，并根据接收到的跳转条件C＝C2_1，由状态S2跳转至状态S1，按照状态S1中的代码运行方式进行运行。根据图3所示，状态S1和状态S2中的代码运行方式均与上述初始化状态S0内的代码运行方式是相同的，本申请实施例对此不再赘述。

[0099] 其中，图4所示的跳转条件可以时同时跳转，也可以是按照一定的时间顺序进行跳转，本申请实施例对此不做任何限定。

[0100] 根据图3所示，在进行调度定义后，可以建立调度描述文件，即根据上述对待处理业务的解析结果，得到待处理业务对应的描述文件。其中，调度描述文件可参见下述表1所示。表1为调度器状态表，其包括生成代码的状态的数量、初始化状态动作及其对应的数量、执行状态动作及其对应的数量。

[0101] 表1

[0102]

状态标识	状态编号	初始化动作标识	初始化动作 1	初始化动作 2	执行动作标识	执行动作 1	执行动作 2
state1	0	en1	S0_Aen_1()	S0_Aen_2()	du1	S0_Adu_1()	S0_Adu_2()
state2	1	en2	S1_Aen_1()	S1_Aen_2()	du2	S1_Adu_1()	S1_Adu_2()
state3	2	en3	S2_Aen_1()	S2_Aen_2()	du3	S2_Adu_1()	S2_Adu_2()

[0103] 表1中包括多个抽象化参数以及各抽象化参数对应的参数值，其中，表1中的表头部分为多个抽象化参数，即状态标识、状态编号、初始化动作标识、初始化动作1、初始化动作2、执行动作标识、执行动作1以及执行动作2。抽象化参数对应的各个参数值均为每个抽象化参数所在列的数值，例如，状态编号对应的参数值为0、1和2。

[0104] 此外，调度描述文件还包括调度器状态迁移表，可参见下述表2所示。

[0105] 表2

[0106]

状态跳转标识	当前状态	目标状态	跳转条件	退出状态动作标识	退出状态动作 1	退出状态动作 2
transition1	0	1	C==C0_1	ex1	C0_1_A1( )	C0_1_A2( )
transition2	0	2	C==C0_2	ex2	C0_2_A1( )	C0_2_A2( )
transition3	1	2	C==C1_2	ex3	C1_2_A1( )	C1_2_A2( )
transition4	1	0	C==C1_0	ex4	C1_0_A1( )	C1_0_A2( )
transition5	2	0	C==C2_0	ex5	C2_0_A1( )	C2_0_A2( )
transition6	2	1	C==C2_1	ex6	C2_1_A1( )	C2_1_A2( )

[0107] 表2中同样包括多个抽象化参数以及各抽象化参数对应的参数值，其中，抽象化参数为表2的表头部分，即抽象化参数为状态跳转标识、当前状态、目标状态、跳转条件、退出状态动作标识、退出状态动作1以及退出状态动作2。参数值为各抽象化所在列的数值，例如，抽象化参数跳转条件对应的参数值为C＝＝C0_1、C＝＝C0_2、C＝＝C1_2、C＝＝C1_0、C＝＝C2_0以及C＝＝C2_1。

[0108] 根据表1和表2确定目标抽象化参数，其中，目标抽象化参数为状态编号、初始化动作1、初始化动作2、执行动作1、执行动作2、当前状态、目标状态、跳转条件、退出状态动作1以及退出状态动作2，根据目标抽象化参数生成对应的代码生成规则。

[0109] 示例的，为例进一步完善生成的代码生成规则，除表1和表2中所示的抽象化参数以及各抽象化参数对应的参数值之外，还需要对调度器的参数进行设置，可通过下述表3的中的参数对调度器的参数进行设置，表3为调度器设计表。

[0110] 表3

[0111]

调度器名称	状态数量	状态变量名称
APPSc	3	APPScStep

[0112] 根据表3中的参数可知，本申请实施例的代码生成方法可以采用状态机来实现，且状态机的数量与表3中的状态数量一致，状态机在运行过程中的状态跳转、状态跳转条件以及各个状态的执行动作与表1和表2中的抽象化参数对应的参数值一致。其中，根据表3中的状态变量名称可知，本申请实施例中状态机为switch……case……的形式。

[0113] 可以理解的是，可以根据表3所示的调度器的参数对生成的代码生成规则进行更新完善，得到新的代码生成规则。示例的，在对代码生成规则进行更新时，还可以设置生成代码的其他参数，例如，生成的目标代码包含一个.c和一个.h的文件，生成的文件名分别为xxx.c和xxx.h，其中，xxx为调度名称；且xxx.h中需添加type.h和util.h头文件引用，使得嵌入式芯片相对应的数据类型和系统定义一致；目标代码可以为一个独立的函数，函数名可以为void xxx_step(void)的形式，同时需要在xxx.h中对该函数进行标记。本申请实施例仅以上述参数为例对生成的代码生成规则进行更新，但并不代表本申请实施例仅局限于此。

[0114] 根据图3所示，在生成新的代码生成规则后，根据新的代码生成规则，编写相对应的代码生成工具，并通过将表1和表2所示的调度描述文件导入代码生成工具内，代码生成工具通过读取目标抽象化参数对应的参数值，可以自动生成代码，即生成待处理业务对应的目标代码。

[0115] 根据上述方法生成的目标代码为：

[0116]
```
#include "APPSc.h"

//-------------------------------------------------------//

//Function Name: APPSc

//-------------------User code-------------------------------//

#define       en          0

#define       du          1

#define       ex          2


extern U8 APPScStep=0;

extern U8 APPScStep_0_cnt=0;

extern U8 APPScStep_1_cnt=0;

extern U8 APPScStep_2_cnt=0;
```

[0117]

```
void APPSc_Init(void)
{
    APPScStep=0;
    APPScStep_0_cnt=0;
    APPScStep_1_cnt=0;
    APPScStep_2_cnt=0;
}

void APPSc_step(void)
{
    switch(APPScStep)
    {
        case 0:
            switch(APPScStep_0_cnt)
            {
                case en:
                    S0_Aen_1( );
                    S0_Aen_2( );
                    APPScStep_0_cnt=du;
                    break;
                case du:
                    S0_Adu_2( );
                    APPScStep_0_cnt=ex;
                    break;
                case ex:
                    if (C==C0_1)
                    {
                        C0_1_A1( );
                        C0_1_A2( );
                        APPScStep=1;
```

```
APPScStep_0_cnt=en;
                    }
                    else if (C==C0_2)
                    {
                        C0_2_A1( );
                        C0_2_A2( );
                        APPScStep=2;
                        APPScStep_0_cnt=en;
                    }
                    else
                    {
                        APPScStep_0_cnt=du;
                    }
                    break;
                default:
                    APPScStep_0_cnt = en;
                    break;
            }
            break;
        case 1:
            switch(APPScStep_1_cnt)
            {
                case en:
                    S1_Aen_1( );
                    S1_Aen_2( );
                    APPScStep_1_cnt=du;
                    break;
                case du:
                    S1_Adu_2( );
                    APPScStep_1_cnt=ex;
```

[0118]

```
                    break;
                case ex:
                    if (C==C1_2)
                    {
                        C1_2_A1( );
                        C1_2_A2( );
                        APPScStep=2;
                        APPScStep_1_cnt=en;
                    }
                    else if (C==C1_0)
                    {
                        C1_0_A1( );
                        C1_0_A2( );
                        APPScStep=0;
        APPScStep_1_cnt=en;
```

[0119]

```
                    }
                    else
                    {
                        APPScStep_1_cnt=du;
                    }
                    break;
                default:
                    APPScStep_1_cnt = en;
                    break;
            }
            break;
        case 2:
            switch(APPScStep_2_cnt)
            {
                case en:
```

```
            S2_Aen_1( );
            S2_Aen_2( );
            APPScStep_2_cnt=du;
            break;
        case du:
            S2_Adu_2( );
            APPScStep_2_cnt=ex;
            break;
        case ex:
            if (C==C2_0)
            {
                C2_0_A1( );
                C2_0_A2( );
                APPScStep=0;
                APPScStep_2_cnt=en;
            }
            else if (C==C2_1)
            {
                C2_1_A1( );
                C2_1_A2( );
                APPScStep=1;
                APPScStep_2_cnt=en;
            }
            else
            {
                APPScStep_2_cnt=du;
            }
            break;
        default:
            APPScStep_2_cnt = en;
```

[0120]

```
                    break;
                }
                break;
            default:
[0121]              APPScStep=0;
                    break;
        }
    }
    //-------------------APPSc end----------------//
```

[0122] 图5为本申请实施例提供的一种代码生成装置50的结构示意图，示例的，请参见图5所示，该代码生成装置50可以包括：

[0123] 接收单元501，用于接收代码生成指令；其中，代码生成指令中包括待处理业务的标识。

[0124] 处理单元502，用于对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件；其中，描述文件中包括多个抽象化参数以及各抽象化参数对应的参数值。

[0125] 生成单元503，用于根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码。

[0126] 输出单元504，用于输出待处理业务对应的目标代码。

[0127] 可选的，生成单元503，具体用于根据多个抽象化参数，生成待处理业务对应的代码生成规则；根据代码生成规则生成代码生成规则对应的代码生成工具；根据描述文件和代码生成工具生成待处理业务对应的目标代码。

[0128] 可选的，生成单元503，具体用于将描述文件导入至代码生成工具中，生成待处理业务对应的目标代码。

[0129] 可选的，生成单元503，具体用于根据代码生成规则，确定待处理业务对应的目标代码的架构；并根据待处理业务对应的目标代码的架构，生成代码生成规则对应的代码生成工具。

[0130] 可选的，生成单元503，具体用于将多个抽象化参数中，处于表格的表头位置的抽象化参数确定为多个目标抽象化参数；根据多个目标抽象化参数，生成待处理业务对应的代码生成规则。

[0131] 可选的，生成单元503，具体用于确定待处理业务的类型，待处理业务对应的目标代码的函数类型、目标代码的文件类型以及目标代码的实现方式；根据待处理业务的类型，目标代码的函数类型、目标代码的文件类型以及目标代码的运行规则，对代码生成规则进行更新，得到新的代码生成规则。

[0132] 可选的，处理单元502，具体用于对待处理业务进行解析，得到待处理业务对应的状态信息、条件信息以及执行动作信息；基于状态信息、条件信息以及执行动作信息，对待处理业务进行抽象化处理。

[0133] 本申请实施例提供的代码生成装置50，可以执行上述任一实施例中的代码生成方法的技术方案，其实现原理以及有益效果与代码生成方法的实现原理及有益效果类似，可参见代码生成方法的实现原理及有益效果，此处不再进行赘述。

[0134] 图6为本申请实施例提供的另一种代码生成装置60的结构示意图，示例的，请参见图6所示，该代码生成装置60可以包括处理器601和存储器602；

[0135] 其中，

[0136] 所述存储器602，用于存储计算机程序。

[0137] 所述处理器601，用于读取所述存储器602存储的计算机程序，并根据所述存储器602中的计算机程序执行上述任一实施例中的代码生成方法的技术方案。

[0138] 可选地，存储器602既可以是独立的，也可以跟处理器601集成在一起。当存储器602是独立于处理器601之外的器件时，代码生成装置60还可以包括：总线，用于连接存储器602和处理器601。

[0139] 可选地，本实施例还包括：通信接口，该通信接口可以通过总线与处理器601连接。处理器601可以控制通信接口来实现上述代码生成装置60的接收和发送的功能。

[0140] 本申请实施例所示的代码生成装置60，可以执行上述任一实施例中的代码生成方法的技术方案，其实现原理以及有益效果与代码生成方法的实现原理及有益效果类似，可参见代码生成方法的实现原理及有益效果，此处不再进行赘述。

[0141] 本申请实施例还提供一种计算机可读存储介质，该计算机可读存储介质中存储有计算机执行指令，当处理器执行所述计算机执行指令时，实现上述任一实施例中的代码生成方法的技术方案，其实现原理以及有益效果与代码生成方法的实现原理及有益效果类似，可参见代码生成方法的实现原理及有益效果，此处不再进行赘述。

[0142] 本申请实施例还提供了一种计算机程序产品，包括计算机程序，该计算机程序被处理器执行时，实现上述任一实施例中的代码生成方法的技术方案，其实现原理以及有益效果与代码生成方法的实现原理及有益效果类似，可参见代码生成方法的实现原理及有益效果，此处不再进行赘述。

[0143] 在本申请所提供的几个实施例中，应该理解到，所揭露的装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所展示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或单元的间接耦合或通信连接，可以是电性，机械或其它的形式。

[0144] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元展示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用硬件加软件功能单元的形式实现。

[0145] 上述以软件功能模块的形式实现的集成的模块，可以存储在一个计算机可读取存储介质中。上述软件功能模块存储在一个存储介质中，包括若干指令用以使得一台计算机

设备(可以是个人计算机,服务器,或者网络设备等)或处理器(英文:processor)执行本申请各个实施例方法的部分步骤。

[0146] 应理解的是,上述处理器可以是中央处理单元(英文:Central Processing Unit,简称:CPU),还可以是其他通用处理器、数字信号处理器(英文:Digital Signal Processor,简称:DSP)、专用集成电路(英文:Application Specific Integrated Circuit,简称:ASIC)等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。结合发明所公开的方法的步骤可以直接体现为硬件处理器执行完成,或者用处理器中的硬件及软件模块组合执行完成。

[0147] 存储器可能包含高速RAM存储器,也可能还包括非易失性存储NVM,例如至少一个磁盘存储器,还可以为U盘、移动硬盘、只读存储器、磁盘或光盘等。

[0148] 总线可以是工业标准体系结构(Industry Standard Architecture,ISA)总线、外部设备互连(Peripheral Component,PCI)总线或扩展工业标准体系结构(Extended Industry Standard Architecture,EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示,本申请附图中的总线并不限定仅有一根总线或一种类型的总线。

[0149] 上述计算机可读存储介质可以是由任何类型的易失性或非易失性存储设备或者它们的组合实现,如静态随机存取存储器(SRAM),电可擦除可编程只读存储器(EEPROM),可擦除可编程只读存储器(EPROM),可编程只读存储器(PROM),只读存储器(ROM),磁存储器,快闪存储器,磁盘或光盘。存储介质可以是通用或专用计算机能够存取的任何可用介质。

[0150] 最后应说明的是:以上各实施例仅用以说明本申请的技术方案,而非对其限制;尽管参照前述各实施例对本申请进行了详细的说明,本领域的普通技术人员应当理解:其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分或者全部技术特征进行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本申请各实施例技术方案的范围。

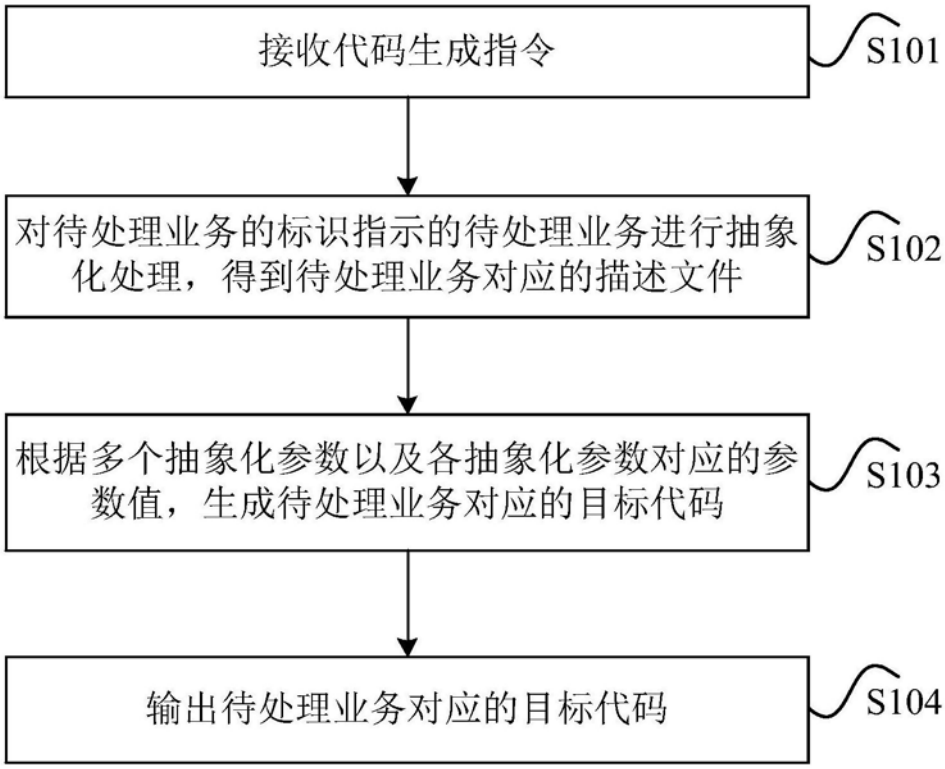

接收代码生成指令
S101
对待处理业务的标识指示的待处理业务进行抽象化处理，得到待处理业务对应的描述文件
S102
根据多个抽象化参数以及各抽象化参数对应的参数值，生成待处理业务对应的目标代码
S103
输出待处理业务对应的目标代码
S104

图1

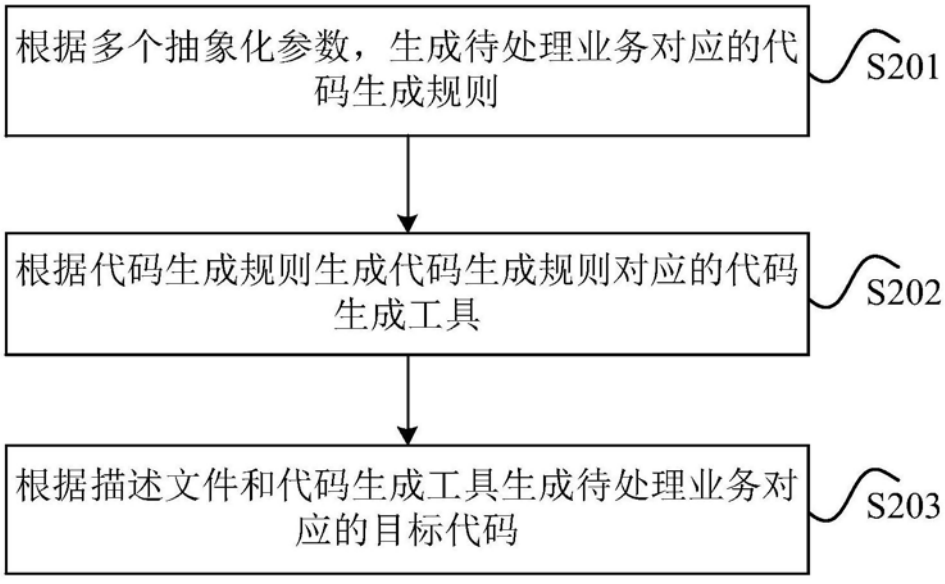

根据多个抽象化参数，生成待处理业务对应的代码生成规则
S201
根据代码生成规则生成代码生成规则对应的代码生成工具
S202
根据描述文件和代码生成工具生成待处理业务对应的目标代码
S203

图2

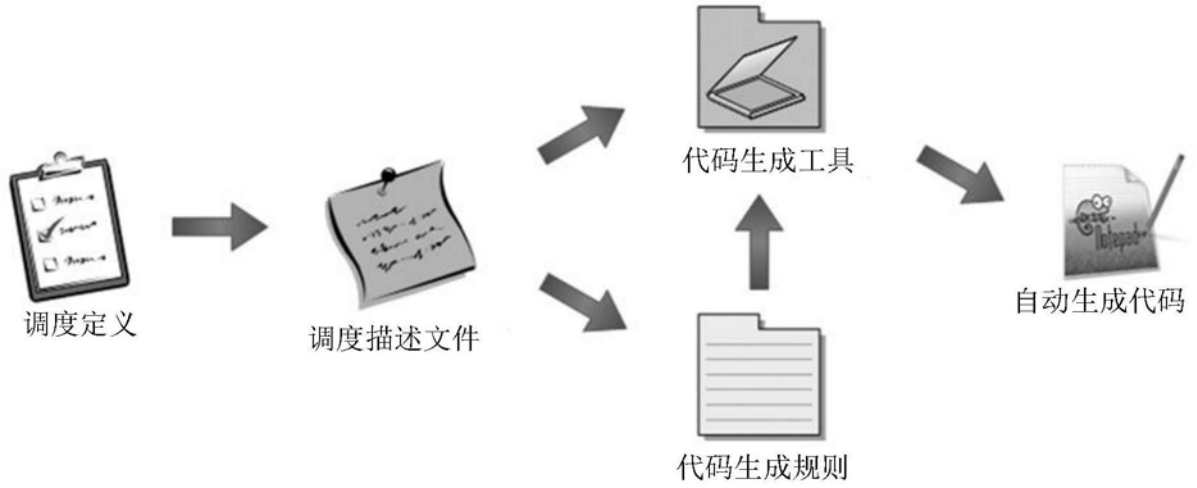

调度定义
调度描述文件
代码生成工具
代码生成规则
自动生成代码

图3

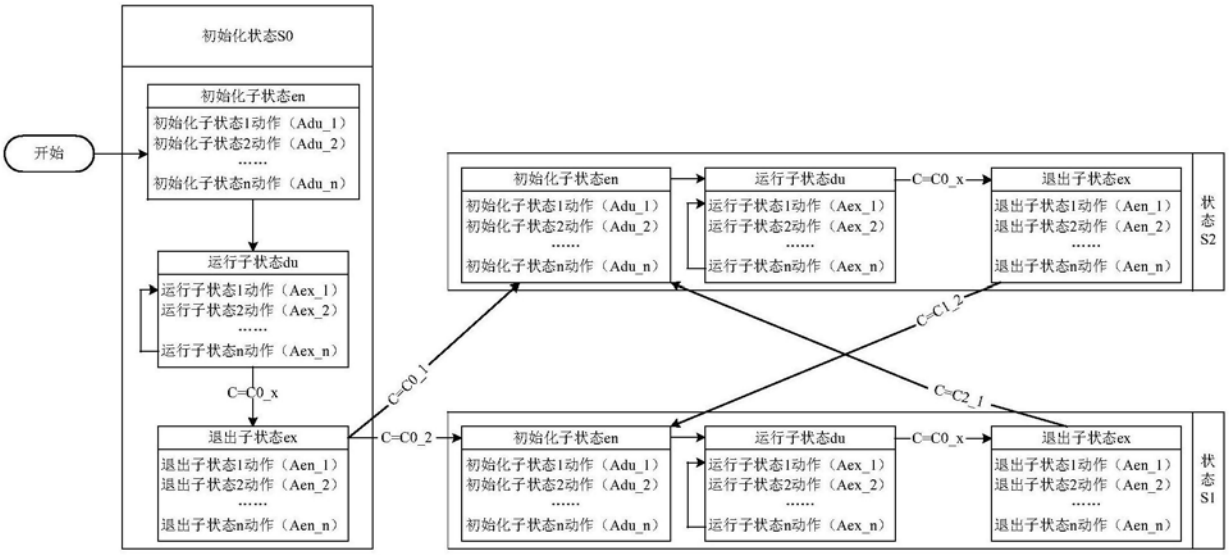

初始化状态S0
初始化子状态en
初始化子状态1动作（Adu_1）
初始化子状态2动作（Adu_2）
……
初始化子状态n动作（Adu_n）
开始
运行子状态du
运行子状态1动作（Aex_1）
运行子状态2动作（Aex_2）
……
运行子状态n动作（Aex_n）
C=C0_x
退出子状态ex
退出子状态1动作（Aen_1）
退出子状态2动作（Aen_2）
……
退出子状态n动作（Aen_n）
C=C0_1
C=C0_2
C=C1_2
C=C2_1
状态S2
状态S1

图4

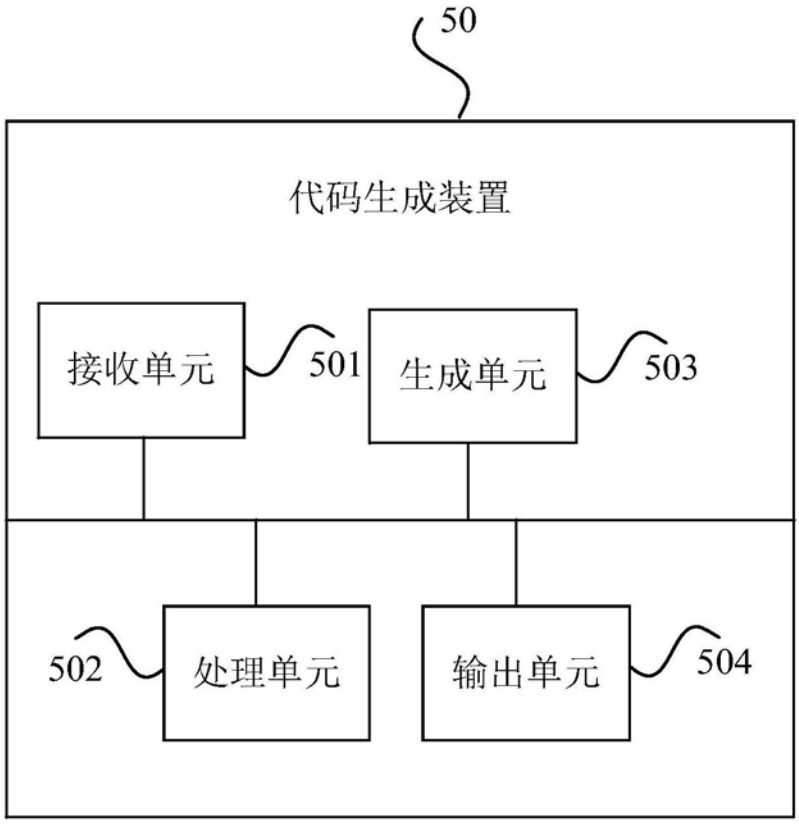

50
代码生成装置
接收单元
501
生成单元
503
502
处理单元
输出单元
504

图5

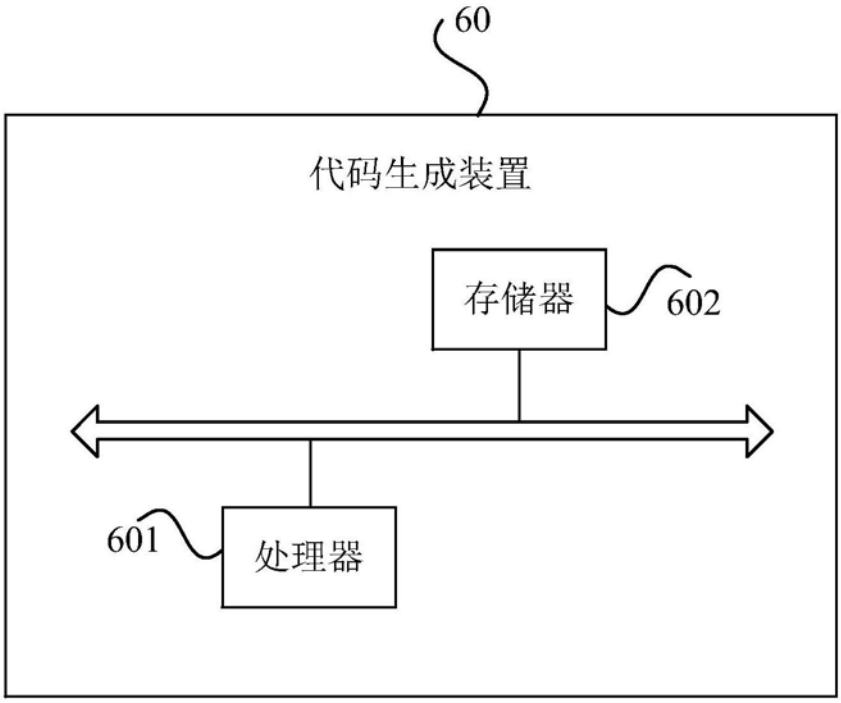

60
代码生成装置
存储器
602
601
处理器

图6