

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2012 年 11 月 15 日 (15.11.2012)



WIPO | PCT



(10) 国际公布号
WO 2012/151886 A1

- (51) 国际专利分类号:
H04L 29/06 (2006.01)
- (21) 国际申请号: PCT/CN2011/081952
- (22) 国际申请日: 2011 年 11 月 8 日 (08.11.2011)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
201110120581.9 2011 年 5 月 10 日 (10.05.2011) CN
- (71) 申请人 (对除美国外的所有指定国): 清华大学 (TSINGHUA UNIVERSITY) [CN/CN]; 中国北京市海淀区清华园, Beijing 100084 (CN)。
- (72) 发明人: 及
- (75) 发明人/申请人 (仅对美国): 徐华 (XU, Hua) [CN/CN]; 中国北京市海淀区清华园, Beijing 100084 (CN)。
- (74) 代理人: 北京清亦华知识产权代理事务所 (普通合伙) (TSINGYIHUA INTELLECTUAL PROPERTY LLC); 中国北京市海淀区清华园清华大学照澜院商业楼 301 室, Beijing 100084 (CN)。

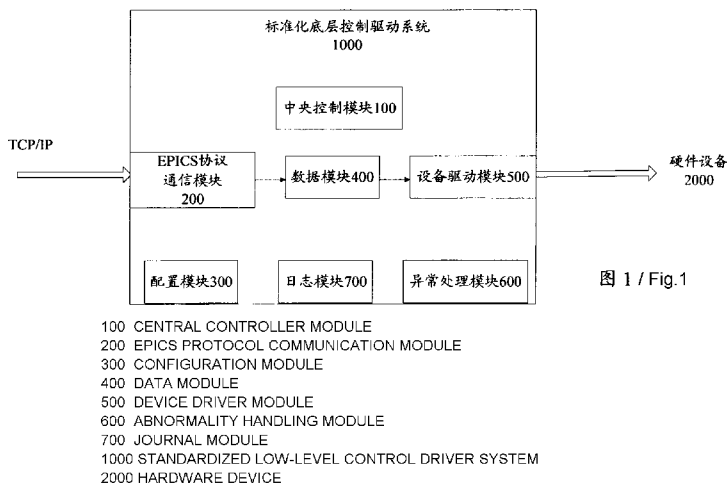
- (81) 指定国 (除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。
- (84) 指定国 (除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

(54) Title: STANDARDIZED LOW-LEVEL CONTROL DRIVER SYSTEM

(54) 发明名称: 一种标准化底层控制驱动系统



(57) Abstract: Disclosed is a standardized low-level control driver system, applicable in receiving a device operation request from a client terminal, and in controlling an action of a hardware device on the basis of the operation request. The standardized low-level control driver system comprises a central controller module for initializing the driver system and for creating a TCP/IP socket, an EPICS protocol communication module for establishing via the TCP/IP socket a communication connection to the client terminal and for communicating with the client terminal by using the EPICS protocol to receive the device operation request from the client terminal and to exchange channel data with the client terminal, a configuration module applicable in generating a variable parameter configuration file and in configuring a variable parameter in the driver system, a data module applicable in storing the channel data from the EPICS protocol communication module, a device driver module for processing the device operation request and for controlling the hardware device on the basis of the channel data stored by the data module and of the variable parameter configured by the configuration module.

[见续页]

WO 2012/151886 A1



(57) 摘要:

本发明公开一种标准化底层控制驱动系统，用于接收来自客户端的设备操作请求，并根据所述设备操作请求控制硬件设备的动作，标准化底层控制驱动系统包括中央控制模块，对驱动系统进行初始化并创建 TCP / IP 套接字；EPICS 协议通信模块，通过 TCP / IP 套接字建立与客户端的通信连接，并采用 EPICS 协议与客户端进行通信以接收来自客户端的设备操作请求及与客户端交换通道数据；配置模块，用于生成变化参数配置文件以配置驱动系统中的变化参数；数据模块，用于存储来自 EPICS 协议通信模块的通道数据；设备驱动模块，处理设备操作请求，并根据数据模块存储的通道数据以及配置模块配置的变化参数控制硬件设备。

一种标准化底层控制驱动系统

技术领域

本发明涉及计算机应用和集成电路装备技术领域，特别涉及一种标准化底层控制
5 驱动系统。

背景技术

随着微电子技术、信息技术、网络技术等的飞速发展，全球信息产业迅猛成长、
网络经济快速兴起、知识经济初见端倪，现代国防和未来战争中的尖端技术不断崛起，
10 人类进入了信息和网络时代。由于微电子技术的核心地位，微电子比以往任何时候都
更显示出其重要的战略地位，微电子产业已经成为全球经济的重要支柱和经济、社会
发展的强大推动力。

驱动控制系统作为 IC（Integrated Circuit，集成电路）产业的一部分，具有举足轻
重的作用。但是现有的集成电路的驱动控制系统通用性差，不能够满足集成电路制造
15 装备工艺模块控制器（PMC）和传输模块控制器（TMC）与常见的外部硬件设备的通
信，也从而导致对集成电路制造设备的各种硬件设备的操作和处理效率低。

发明内容

本发明的目的旨在至少解决上述技术缺陷之一，特别提出一种高效、通用的用于
20 集成电路制造设备的标准化底层控制驱动系统。

为达到上述目的，本发明第一方面的实施例提出一种标准化底层控制驱动系统，
所述标准化底层控制驱动系统用于接收来自客户端的设备操作请求，并根据所述设备
操作请求控制硬件设备的动作，所述标准化底层控制驱动系统包括：中央控制模块，
所述中央控制模块对所述标准化底层控制驱动系统进行初始化并创建所述标准化底层
25 控制驱动系统的 TCP / IP 套接字以与客户端进行通信；EPICS 协议通信模块，所述
EPICS 协议通信模块通过所述 TCP / IP 套接字建立与所述客户端的通信连接，并采用
EPICS 协议与所述客户端进行通信以接收来自所述客户端的设备操作请求及与所述客
户端交换通道数据；配置模块，所述配置模块用于生成变化参数配置文件以配置所述
标准化底层控制驱动系统中的变化参数；数据模块，所述数据模块用于存储所述来自
30 所述 EPICS 协议通信模块的通道数据；和设备驱动模块，所述设备驱动模块处理所述
设备操作请求，并根据所述数据模块存储的通道数据以及所述配置模块配置的变化参
数控制所述硬件设备。

在本发明的一个实施例中，所述中央控制模块包括：初始化单元，所述初始化单
元用于对所述 EPICS 协议通信模块、所述配置模块、所述数据模块和所述设备驱动模
35 块进行初始化，并创建所述 EPICS 协议通信模块的 TCP / IP 套接字；测试单元，所述

测试单元用于定时测试所述硬件设备的状态以更新与所述硬件设备相关的通道的参数调度单元，所述调度单元用于对所述标准化底层控制驱动系统中的各个任务进行调度以控制所述各个任务独立执行。

在本发明的一个实施例中，所述调度单元将所述各个任务放入任务优先级队列，并根据所述各个任务对应的通道的优先级通过线程池中的多个工作线程，并行执行属于同一优先级的多个任务。

在本发明的一个实施例中，所述变化参数配置文件以 XML 文件格式保存。

在本发明的一个实施例中，所述变化参数配置文件中的变化参数包括：

所述 EPICS 协议通信模块和所述客户端之间的各个通道的配置参数、所述标准化底层控制驱动系统向所述客户端定时反馈通道值的时间间隔、所述标准化底层控制驱动系统定时测试所述硬件设备的测试时间间隔、所述标准化底层控制驱动系统端的 IP 地址、所述标准化底层控制驱动系统监听网络连接的端口、所述标准化底层控制驱动系统中存在的线程的最大数量和所述标准化底层控制驱动系统中并行执行的线程数量。

在本发明的一个实施例中，所述配置模块进一步生成用于解析所述变化参数配置文件的键类。

在本发明的一个实施例中，在所述中央处理模块初始化所述标准化底层控制驱动系统后，所述 EPICS 协议通信模块监听来自所述客户端的连接请求，在接收到来自所述客户端的连接请求后，建立与所述客户端的通信连接。

在本发明的一个实施例中，所述 EPICS 协议通信模块包括：连接状态机，所述连接状态机用于维持所述 EPICS 协议通信模块与所述客户端之间的连接的状态；读数据状态机，所述读数据状态机用于读取并处理所述客户端与所述标准化底层控制驱动系统之间在通信中传输的通道数据；反馈状态机，所述反馈状态机用于在所述客户端向所述标准化底层控制驱动系统查询通道值时，向所述客户端返回请求的所述通道值，并在所述通道值变化时，向所述客户端发送通道值变化通知；监测状态机，所述监测状态机用于定时向所述客户端发送数据包以检测所述 EPICS 协议通信模块和所述客户端之间的连接状态，

在本发明的一个实施例中，当所述监测状态机发送的数据包触发到异常时，则检测到所述 EPICS 协议通信模块和所述客户端之间通信连接断开。

在本发明的一个实施例中，所述数据模块存储的通道数据包括：通道类型，所述通道类型包括连续型、离散型、字符串型；通道读写类型，所述通道读写类型包括只读型、只写型、读写型；通道类型的父类，所述通道类型的父类用于封装所述的通道的操作；节点类，所述节点类为逻辑上或物理上相关的 EPICS 通道的集合，包括多个节点，所述多个节点是连续型通道集合、离散型通道集合或字符串型通道集合；通道工厂类，所述通道工厂类用于管理和创建节点和通道。

在本发明的一个实施例中，所述通道包括：连续型只读通道、连续型只写通道、连续型读写通道、离散型只读通道、离散型只写通道、离散型读写通道、字符串型只读通道、字符串型只写通道和字符串型读写通道。

在本发明的一个实施例中，每个通道对应于一个硬件设备。

- 5 在本发明的一个实施例中，所述标准化底层控制驱动系统包括至少一个节点，其中所述标准化底层控制驱动系统任意指定作为所述节点的 EPICS 通道集合。

在本发明的一个实施例中，当创建一个通道时，将所述通道同时添加到所述通道工厂类和所述通道所属的节点的 EPICS 通道集合中。

- 10 在本发明的一个实施例中，所述设备驱动模块包括：映射单元，所述映射单元用于对所述通道和所述硬件设备的读/写接口进行映射以对所述硬件设备进行读/写操作；读写单元，所述读写单元用于读取所述数据模块存储的通道数据、所述配置模块配置的变化参数以及所述硬件设备的数据，并向所述硬件设备中写入数据；超时处理单元，所述超时处理单元用于处理对所述硬件设备的读操作超时；和设备管理单元，所述设备管理单元用于创建和维护所述硬件设备。

- 15 在本发明的一个实施例中，所述超时处理单元判断在预定时间内是否获得读写锁，如果在所述预定时间内未获得所述读写锁，则抛出超时异常，如果在所述预定时间内获得所述读写锁，则通过异步委托机制在另一个线程里判断是否超时，如果是则抛出超时异常，并释放所述读写锁，如果否则读取所述硬件设备的数据，并释放所述读写锁。

- 20 在本发明的一个实施例中，设备管理单元将所述硬件设备分为四类，包括：随机模拟类设备，所述随机模拟类设备中的硬件设备的每次读操作均分别返回一个随机的正数；连续型类设备，所述连续型设备中的硬件设备保存一个双精度浮点型数值，并对所述双精度浮点型数值进行读/写；离散型设备，所述离散型设备中的硬件设备保存一个整型数值，并对所述整型数值进行读/写；字符串型设备，所述字符串型设备中的
25 硬件设备保存一个字符串型数值，并对所述字符串型数值进行读/写。

在本发明的一个实施例中，进一步包括异常处理模块，所述异常处理模块用于在所述标准化底层控制驱动系统抛出异常时，查找异常处理向量表以获得所述异常的异常处理函数，并根据所述异常处理函数对所述异常进行处理。

- 30 在本发明的一个实施例中，所述异常处理向量表包括异常类型以及每种异常类型对应的异常处理函数。

在本发明的一个实施例中，进一步包括日志模块，所述日志模块以日志的形式记录所述标准化底层控制驱动系统运行过程中的信息。

- 35 本发明实施例提供的用于集成电路制造设备的标准化底层控制驱动系统基于 Windows xp 操作系统，采用 Visual C#实现。该标准化底层控制驱动系统包括中央控制模块、EPICS 协议通信模块、配置模块、数据模块和设备驱动模块。其中，中央控制

模块可以对整个驱动系统进行初始化和调度，包括创建驱动系统的 TCP / IP 套接字以与客户端进行通信。

EPICS 协议通信模块可以根据创建的 TCP / IP 套接字建立与所述客户端的通信连接，并采用 EPICS 协议与所述客户端进行通信。对 EPICS 协议的处理包括 EPICS 通信的管理和 EPICS 数据包的处理，提供可靠的通信连接。

配置模块可以生成变化参数配置文件以配置所述标准化底层控制驱动系统中的变化参数。当驱动系统需要这些参数时，可以在运行时通过配置模块获取这些参数。当系统中这些参数经常变化时，可以直接修改配置文件，而不用重新编译整个系统，从而大大提高了驱动系统的可移植性。

数据模块包括驱动系统常用的一些数据结构的抽象，包括数据项和通道等。数据模块存储 EPICS 协议通信模块的通道数据，采用面向对象的相关技术，减小对象间的耦合性，从而提高数据的组织效率。

设备驱动模块和硬件设备进行通信，处理设备操作请求，根据数据模块存储的通道数据以及配置模块配置的变化参数控制硬件设备。

本发明实施例提供的标准化底层控制驱动系统可以使得集成电路制造设备中的工艺模块控制器（PMC）、集群控制器（CTC）和传输模块控制器（TMC）和外部硬件设备进行通信，高效调度和处理各个控制子系统对集成电路制造设备的各种硬件设备的大量操作请求。

本发明附加的方面和优点将在下面的描述中部分给出，部分将从下面的描述中变得明显，或通过本发明的实践了解到。

附图说明

本发明上述的和/或附加的方面和优点从下面结合附图对实施例的描述中将变得明显和容易理解，其中：

图 1 为根据本发明实施例的标准化底层控制驱动系统的示意图；

图 2 为根据本发明实施例的标准化底层控制驱动系统与客户端及硬件设备交互的示意图；

图 3 为中央控制模块的结构示意图；

图 4 为中央控制模块对 GotPacketDelegate 的事件处理流程；

图 5 为驱动系统中各种任务的继承关系图；

图 6 为调度单元的调度示意图；

图 7 为任务工厂类的继承类示意图；

图 8 为 EPICS 协议通信模块的通信示意图；

图 9 为 EPICS 协议通信模块中各种对象的创建过程示意图；

图 10 为连接状态机的状态转换图；

图 11 为读数据状态机的状态转换图；

图 12 为反馈状态机的状态转换图；

图 13 为监测状态机的状态转换图；

图 14 为 EPICS 协议通信模块的各个状态机之间的协作图；

5 图 15 为通道类的继承关系图；

图 16 为通道与硬件设备驱动的关系图；

图 17 为 AbstractDevice 类内部的函数调用关系；

图 18 为设备驱动模块的结构示意图；

图 19 为设备读操作时处理的流程图；

10 图 20 为 DeviceFactory 的 UML 描述示意图；

图 21 为异常处理模块的结构示意图；

图 22 为异常处理模块的处理流程图；和

图 23 为根据本发明实施例的驱动系统整体运行状态机模型。

15 具体实施方式

下面详细描述本发明的实施例，所述实施例的示例在附图中示出，其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施例是示例性的，仅用于解释本发明，而不能解释为对本发明的限制。

如图 1 所示，本发明实施例提供的标准化底层控制驱动系统 1000，包括中央控制
20 模块 100、EPICS 协议通信模块 200、配置模块 300、数据模块 400 和设备驱动模块 500。其中，中央控制模块 100 对整个驱动系统 1000 进行初始化并创建驱动系统 1000 的 TCP/IP 套接字以与客户端进行通信；EPICS 协议通信模块 200 通过 TCP/IP 套接字建立与客户端的通信连接，并采用 EPICS 协议与客户端进行通信以接收来自客户端的设备操作请求及与客户端交换通道数据；配置模块 300 用于生成变化参数配置文件以配
25 置驱动系统 1000 中的变化参数；数据模块 400 用于存储来自 EPICS 协议通信模块 200 的通道数据；设备驱动模块 500 用于处理设备操作请求，并根据数据模块 400 存储的通道数据以及配置模块 300 配置的变化参数控制硬件设备。本发明实施例提供的标准化底层控制驱动系统 1000 可以用于对集成电路制造设备的控制。

图 2 示出了本发明实施例提供的标准化底层控制驱动系统 1000 与客户端 3000 和
30 硬件设备 2000 的交互过程。在集成电路制造装备的控制系统中，存在大量需要被控制的硬件设备 2000（例如串口设备和数据采集卡）。标准化底层控制驱动系统 1000 和客户端 300 采用图 2 所示的 C/S 型体系结构。如图 2 所示，标准化底层驱动系统 1000 相当于数据服务器，采用 EPICS 通信协议与客户端 3000 进行通信。标准化底层控制驱动系统 1000 与客户端 3000 采用 EPICS 通信协议交换通道的数据项（DataItem）。标准
35 化底层控制驱动系统 1000 负责监听网络，处理各种大量的并发的客户端 3000 的设备

操作请求, 根据客户端 300 的请求控制硬件设备 2000, 例如各种串口设备 (Serial) 和 PCI 数据采集卡。采用上述 C/S 型体系结构, 可以弥合各种不同设备的不同操作方式, 由驱动系统 1000 向外提供硬件设备 200 的统一的控制接口。

数据项 (DataItem) 是客户端 3000 与标准化底层控制系统 1000 之间数据交换的逻辑单位。数据项 (DataItem) 是一种抽象的数据结构, 在驱动系统 1000 中主要用来保存硬件设备 2000 的各种控制单元 (I/O 管脚) 的状态, 也可以用来在对象之间传递数据。硬件设备 2000 的各种控制单元 (I/O points) 被映射为 DataItem, 驱动系统 1000 通过控制 DataItem 来控制各种实际的硬件设备 2000。客户端 200 和驱动系统 1000 之间通过 Epics 通信协议来交换通道的数据信息。

EPICS 通信协议为一个基于消息的位于 TCP/IP 套接字之上的通信协议, 在驱动系统 1000 的上下文中用来在客户端 3000 和服务端 (驱动系统 1000 端) 之间交换通道数据。EPICS 协议负责建立一种可靠地、高效的通信连接, 以数据项 (DataItem) 为单位在通信双方之间进行信息交换。

驱动系统 1000 为服务器端, 客户端 3000 向驱动系统 1000 请求操作特定的通道。驱动系统 1000 和客户端 3000 之间建立通信连接包括如下步骤:

- 1) 驱动系统 1000 在计算机特定的端口建立一个服务器端套接字连接;
- 2) 驱动系统 1000 在上述端口监听来自客户端 3000 的连接请求;
- 3) 运行客户端 3000;
- 4) 客户端 3000 设置端口号等环境变量;
- 5) 客户端 3000 生成一个客户端套接字连接, 然后尝试连接驱动系统 1000;
- 6) 驱动系统 1000 接收连接请求, 通信建立。

在连接建立之后, 客户端 3000 可以与驱动系统 1000 进行通讯, 驱动系统 1000 接收并且处理客户端 3000 的操作请求。

驱动系统 1000 采用 EPICS 通信协议来和客户端程序进行通信, 负责处理各种大量的并发的客户端 3000 的设备操作请求。客户端 3000 和驱动系统 1000 之间的主要交互如下:

当客户端 3000 需要从一个通道读取值时, 客户端 3000 会向驱动系统 1000 发送一个 P_READ 型的 EPICS 消息, 驱动系统 1000 在接收到消息之后, 将调用和这个通道相关联的 Read 方法查询设备的特定通道的状态值, 如果查询成功, 则驱动系统 1000 向客户端 3000 返回相应的通道的数值; 如果失败, 则返回一个错误消息。

如果客户端 3000 绑定到驱动系统 1000 的一个通道, 则当通道的数值改变时, 驱动系统 1000 将通知客户端 3000。驱动系统 1000 会定时查询设备, 来检测通道数值是否改变。

当客户端 3000 需要向一个通道写数值时, 客户端 3000 会向驱动系统 1000 发送一个 P_WRITE 型 EPICS 消息。驱动系统 1000 接收到该消息之后, 将调用和这个通道相

关的 Write 方法把设备通道的状态设置为指定的值。

下面参考图 3 至图 7 描述中央控制模块 100。中央控制模块 100 是整个驱动系统 1000 的核心，负责对整个驱动系统 1000 的初始化和调度。在驱动系统 1000 启动时，首先由中央控制模块 1000 初始化相关的对象，创建服务器端套接字，等待客户端的连接。在驱动系统 1000 中，对通道的读写、轮循、对硬件设备的定时测试等动作都被封装成任务，由中央控制模块 100 进行调度。

如图 3 所示，中央控制模块 100 包括初始化单元 110、测试单元 120 和调度单元 130。其中，初始化单元 110 用于对 EPICS 协议通信模块 200、配置模块 300、数据模块 400 和设备驱动模块 500 进行初始化，并创建 EPICS 协议通信模块 200 的 TCP / IP 套接字；测试单元 120 用于以定时测试硬件设备 2000 的状态以更新与硬件设备 2000 相关的通道的参数，换言之，测试单元 120 以预定的测试频率测试硬件设备 2000 的状态；调度单元 130 用于对驱动系统 1000 中的各个任务进行调度以控制各个任务独立执行。

当驱动系统 1000 启动时，初始化单元 110 对相关的对象进行初始化，包括对 EPICS 协议通信模块 200、配置模块 300、数据模块 400 和设备驱动模块 500 进行初始化，创建服务器端套接字，等待客户端 3000 的连接。由 EPICS 协议通信模块 200 维护和管理网络连接，当客户端 3000 和驱动系统 1000 之间的连接建立之后，EPICS 协议通信模块 200 管理网络连接，根据网络状态和接收到的数据包创建相应的任务，采取对应的动作。当中央控制模块 100 接收到来自客户端 3000 的 P_SHUTDOWN 消息时，退出循环。

在中央控制模块 100 的循环中，对消息进行下述过程的处理。

由 EPICS 通信协议模块 200 的状态机监听网络，EpicsReader 接收到 Epics 数据包以后，在方法 ProcessEpicsPacket() 中触发 EpicsReaderEventAdapter 的 GotPacketDelegate 事件，在 GotPacketDelegate 的处理函数中处理这些 Epics 数据包。其中 EpicsReaderEventAdapter 是 EpicsReader 向外提供的一个 Façade，用于对 GotPacketDelegate 事件的注册和处理。在中央控制模块 100 中对 GotPacketDelegate 事件进行处理。

如图 4 所示，对 GotPacketDelegate 事件进行处理包括如下步骤：

S401: 触发 GotPacketDelegate 事件；

S402: 判断消息类型；

当消息类型为 P_READ 类型时，创建 ReadForEpicsTask 型的任务；

当消息类型为 P_WRITE 类型时，创建 WriteForEpicsTask 型的任务；

当消息类型为 P_POLL 类型时，注册 Poller；

当消息类型为 P_UNPOL 类型时，取消注册 Poller。

为了封装的需要，系统中 P_SHUTDOWN 类型的消息已经在类 EpicsReader 的

ProcessEpicsPacket()方法中进行了处理，所以图 4 中未示出 P_SHUTDOWN 类型的消息。

测试单元 120 可以分配单独的定时器线程以轮循地检查硬件设备 2000 的状态。具体而言，硬件设备 2000 的状态需要定时进行测试，以便及时更新与硬件设备 2000 相关的通道的值。针对每个通道，测试单元 120 可以分配单独的定时器线程，从而定时产生通道的读任务，以便从硬件设备 2000 读取值并且设置与硬件设备 2000 相关的通道的当前状态。在本发明的一个实施例中，硬件设备 2000 的测试频率从配置文件中获得。

调度单元 130 为整个中央控制模块 100 的核心。驱动系统 1000 对 Epics 数据包的 10 处理、对硬件设备的测试、通道的读写等操作均封装成任务，由调度单元 130 进行调度。采用这种方式，可以使得各个任务的执行相对独立，便于长时间任务的执行，有利于提高系统的并发性和吞吐量。每个任务的设计，参考了 Command 模式。所有任务的抽象父类 Task 定义如下：

```
protected object id; //任务 ID
protected bool active = true; //任务是活动的，可执行
public Task() {} //构造函数
//参考 Command 模式，任务执行的核心内容
public abstract void Execute(ManagedThreadPool tp); //主要用于在任务执行
完成之后，将任务对象归还到系统对象池中
public abstract void Done();
```

图 5 示出了系统中各种任务的继承关系。其中常用的有 ReadForEpicsTask、WriteForEpicsTask、ReadTask 等任务。

驱动系统 1000 的核心是任务调度和控制系统，其核心框架如图 6 所示。

在驱动系统 1000 中，对通道的读写、轮循（polling）以及对硬件设备的定时测试 25 等均被封装成任务。如图 6 所示，调度单元 130 将各个任务放入任务优先级队列中，并根据优先级通过线程池中的多个工作线程，并行处理各个任务。其中，任务例如为 ReadTask、ReadForEpicsTask 和 WriteForEpicsTask。

在驱动系统 1000 中共有 4 个优先级队列，其中优先级队列 0 和 3 供驱动系统 1000 自身使用，队列 1 供只读通道相关的任务使用，队列 2 供只写和读写型通道相关的任务使用。任务的优先级只与该任务相关的通道的优先级有关。

在本发明实施例的驱动系统 1000 中，实际使用如下三种任务：ReadTask、ReadForEpicsTask 和 WriteForEpicsTask。

下面分别对上述三种任务进行说明。

ReadTask：主要用于在定时测试通道相关的硬件设备时，为每个通道定时分配一个 ReadTask 以读取通道的值。

ReadForEpicsTask: 当驱动系统 1000 (服务器端) 接收到一个 P_READ 消息时, 将分配一个 ReadForEpicsTask 任务, 以读取通道值, 然后向客户端 3000 返回相应的通道值。

WriteForEpicsTask: 当驱动系统 1000 (服务器端) 接收到一个 P_WRITE 消息时, 将分配一个 WriteForEpicsTask 任务, 以向通道写入指定值。

对于一个长时间执行的 C/S 型系统, 客户端 3000 不断请求驱动系统 1000 (服务器端) 的服务, 服务器端根据请求创建任务对象、执行任务、然后销毁任务对象。调度单元 130 中采用了对象池 (ObjectPool) 的概念。不同的任务对象的结构成员不同, 因此每种任务工厂分别维护一个对象池, 从而可以解决大量任务对象的创建和销毁对系统的性能造成负面影响。

对象池 (ObjectPool) 采取一种单链表型的数据结构, 两个指向 PoolableObject 对象的指针 (poolHead 和 poolTail) 分别指向链表的表头和表尾。为了对链表操作的方便, poolTail 指向链表中最后一个任务对象, 但是 poolHead 所指向的表头对象只是起标志作用, 该表头不存储可用的任务对象。对链表的访问是互斥进行的。当需要从对象池取得一个任务对象时, 如果对象池为空, 而且可以创建对象, 则调用任务工厂创建一个对象并返回。如果对象池非空, 则取出链表中 poolHead 的下一个对象, 初始化并且返回该对象。当任务执行完成之后, 任务工厂会把该任务对象添加到链表尾。

在调度单元 130 采用工程模式创建和分配每种任务。在驱动系统 1000 中, 任务工厂类的继承关系如图 7 所示。

在本发明的一个实施例中, 调度单元 130 为了便于管理和调度线程, 将线程封装成 IOCPHandle 对象。

ManagedIOCP 实现了端口完成模型的核心, ManagedIOCP 负责创建和注册线程对象 (IOCPHandle), 将新的任务对象放到合适的优先级队列中, 并且根据任务对象的优先级取出下一个合适的任务对象供系统调度。

ManagedThreadPool 类参考 Facade 模式, 是调度系统对外提供的使用接口。ManagedThreadPool 类初始化 ManagedIOCP 对象, 根据定义创建一个线程对象池, 将线程对象注册到 ManagedIOCP 对象中, 同时也将新的任务对象放到 ManagedIOCP 的优先级队列中。

下面参考图 8 至图 14 描述 EPICS 协议通信模块 200。

EPICS 协议通信模块 200 通过中央控制模块 100 建立的 TCP/IP 套接字建立与客户端 3000 的通信连接, 并采用 EPICS 协议与客户端 3000 进行通信, 接收来自客户端 3000 的设备操作请求及与客户端 3000 交换通道数据。EPICS 通信协议是一个基于 TCP/IP 套接字之上的通信协议。EPICS 协议的处理主要包括 EPICS 通信的管理和 EPICS 数据包的处理, 从而提供可靠的通信连接。

EPICS 协议通信模块 200 通过类 EpicsAdapter 向外提供使用接口。EpicsAdapter

采用 Singleton 模式。在驱动系统 1000 中只有一个 EpicsAdapter 实例管理系统的 Epics 连接。EpicsAdapter 类的设计参考了 Facade 模式，EpicsAdapter 类的定义如下：

```

public class EpicsAdapter
{
5      private EpicsReaderEventAdapter
      readerEventAdapter=EpicsReaderEventAdapter.getInstance();
      private EpicsEventAdapter eventAdapter = EpicsEventAdapter.getInstance();
      private EpicsManager epicsManager;
      public ManualResetEvent manualEvent = new ManualResetEvent(false);
10     private int portNumber = 5001;
      private string hostname = "127.0.0.1";
      private static EpicsAdapter instance=new EpicsAdapter();

      //初始化连接,当 IsServer为true时初始化的是一个服务器端,否则初始化的
15     是一个客户端
      public void Init(bool IsServer);

      public void InitEpicsServer(); //初始化并且启动服务器端
      public void InitEpicsClient(); //初始化并且启动客户端
20     public void Disconnect(); //断开连接
      void NotifyOtherSideToShutDown(); //通知通信的另一端关闭连接
      public void Shutdown(); //关闭连接,清理资源
    }
}

```

25 中央控制模块 100 采用 EpicsAdapter 的 Init 方法来初始化 Epics 连接，当 Init 方法的参数为 True 时，驱动系统 1000 被初始化为服务器端，否则驱动系统 1000 被初始化为客户端。

在中央处理模块 100 初始化后，EPICS 协议通信模块 200 监听来自客户端 3000 的连接请求，在接收到来自客户端 3000 的连接请求后，建立与客户端 3000 的通信连接。

30 具体而言，EPICS 协议通信模块 200 在初始化之后，监听网络。在连接建立之后，Epics 模块的多个状态机分别在单独的线程中执行，多个状态机之间相互作用，共同维护连接状态和处理 Epics 通信。如图 8 所示，与 Epics 数据包处理直接相关的就是 EpicsReader 类，当通信端接收到 Epics 数据包之后，将调用 EpicsReader 的 ProcessEpicsPacket 方法，在这个方法中处理接收到的 Epics 数据包，并且触发相应的
35 GotPacketDelegate 类型的事件（Event）。

图 9 示出了 EPICS 协议通信模块 200 中各种对象的创建过程。其中，

EpicsManager: 对 EpicsConnection 进行一次封装，向外提供操作接口；

EpicsConnection: 负责 EPICS 连接的建立和管理；

EpicsStateMachine: 负责管理连接的状态；

40 EpicsPoller: 主要负责处理 P_POLL 数据包；

EpicsReader: 负责该 EPICS 通信端的数据包的接收和处理；

EpicsMonitor: 定时向通信的对方发送数据包，以检测当前连接状况；

EpicsPacketFactory: Epics 数据包的创建工厂。

Epics 消息类型包括:

P_INIT: 客户端 3000 请求服务器 (驱动系统 1000) 建立 EPICS 通信连接;

P_READ: 客户端 3000 请求服务器 (驱动系统 1000) 返回一个 EPICS 通道的值;

5 P_WRITE: 客户端 3000 请求服务器 (驱动系统 1000) 设置一个 EPICS 通道的 setpoint;

P_POLL: 客户端 3000 请求服务器 (驱动系统 1000) 查询一个 EPICS 通道的值; 当 Control IO 驱动接收到这个消息后立即返回这个通道的值, 并且当该通道的值发生变化时, 将向客户端发送一个 P_REFRESH 消息通知所请求的 EPICS 通道的值已经发生

10 P_UNPOLL: 客户端 3000 请求服务器 (驱动系统 1000) 取消 POLL 一个 EPICS 通道;

P_REFRESH: 当客户端 3000 请求服务器 (驱动系统 1000) 返回一个 EPICS 通道; 且当数据条目值变化时, 服务器向客户端 3000 发送一个 P_REFRESH 消息。

15 P_SHUTDOWN: 客户端 3000 请求服务器 (驱动系统 1000) 关闭建立的 EPICS 通信连接。

表 1 示出了驱动系统 1000 与客户端 3000 交互的 EPICS 数据包结构。

表 1

LEN	DTYPE	OPCODE	NUM1	NUM2	CHID	CHATTR	PEERID
0	4	6	8	12	16	18	20
REQUESTOR		VALUE	EXTRA				
22		24	32				

如表 1 所示, 标准的 EPICS 数据包是一个 32 字节的数组。如果通道属性值可以存放在 8 字节的 VALUE 字段中 (如离散--整型、连续--双精度浮点型), 则 EPICS 数据包无 EXTRA 字段, 此时数据包的第一个字段 LEN 为 0; 如果通道属性值无法存放在 8 字节的 VALUE 字段中 (如字符串型), 则将通道属性值存放在 EXTRA 字段中, 字段长度由数据包的第一个字段 LEN 指定, 此时 EPICS 数据包长度大于 32 字节。

对 EPICS 数据包各字段的意义说明如下:

LEN: EXTRA 字段长度。如果为 0, 则数据包长度为标准的 32 字节; 如果非 0, 则数据包长度为 32+LEN 字节。

DTYPE: 通道数据类型。其中, 通道数据类型包括:

DISCRETE (0): 离散型;

CONTINUOUS (1): 连续型;

ASCII (2) : 字符串型;

ALIST (3) : 字符串列表型;

VARIANT (4) : 通用型。

OPCODE: 操作码。其中, 操作码包括:

5 E_EVENT (-6) : 错误事件;

E_REFRESH (-4) : 错误更新;

P_READ (0) : read 请求;

P_WRITE (1) : write 请求;

P_POLL (2) : poll 请求;

10 P_UNPOLL (3) : unpoll 请求;

P_INIT (4) : 初始化 epics 连接;

P_SHUTDOWN (5) : 关闭 epics 连接;

P_REFRESH (6) : 通道属性值更新;

P_GETIDBYNAME (8) : 通过 name 获取 id 请求;

15 P_MESSAGE (11) : 消息;

P_HEARTBEAT (27) : peer 存活;

P_EVENT (28) : 事件。

EPICS 服务器支持的最小服务集对应: READ、WRITE、POLL、UNPOLL、INIT、SHUTDOWN、REFRESH 及 E_REFRESH。

20 NUM1: Board Number (Node Number, 节点号);

NUM2: Channel Number (通道号);

CHID: Channel ID (通道 ID);

CHATTR: Channel Attribute (通道属性), 包括:

NAME (0) : 名称;

25 UNITS (1) : 单位;

DT_VALUE (2) : 离散型通道值;

DT_SETPOINT (3) : 离散型通道设置点;

CT_VALUE (4) : 连续型通道值;

CT_SETPOINT (5) : 连续型通道设置点;

30 ST_SETPOINT (6) : 字符串型通道设置点;

ST_VALUE (19) : 字符串型通道值;

DESCRIPTOR (21) : 描述字符串;

ACCURACY (22) : 精度;

MIN (23) : 最小值;

35 MAX (24) : 最大值。

PEERID: 对等实体 ID (EPICS 数据包接收方 ID)。

REQUESTOR: Request ID (请求 ID, 和一个 EPICS 请求相对应)。

VALUE: 用于存放通道属性值 (非字符串类型, 可存放在 8 字节字段中)。

EXTRA: 存储字符串型通道属性值。

5 EPICS 协议通信模块 200 包括四个状态机, 每个状态机都以一个单独的线程进行。
四个状态机包括连接状态机 (EpicsStateMachine)、读数据状态机 (EpicsReader)、反
馈状态机 (EpicsPoller) 和监测状态机 (EpicsMonitor)。其中, 连接状态机用于维持
所述 EPICS 协议通信模块 200 与客户端 3000 之间的连接的状态, 状态为枚举值
ConnectionStates; 读数据状态机用于读取并处理客户端 3000 与标准化底层控制驱动系
10 统 1000 之间在通信中传输的通道数据, 状态为枚举值 ReaderStates; 在客户端 3000 向
标准化底层控制驱动系统 1000 查询 Epics 通道的通道值时, 反馈状态机向客户端 3000
返回请求的 Epics 通道的通道值, 并在该 Epics 通道的通道值变化时, 向客户端 3000
发送 P_REFRESH 消息来通知该 Epics 通道的通道值已经发生变化, 状态为枚举值
PollerStates; 监测状态机用于定时向客户端 3000 发送数据包以检测 EPICS 协议通信模
15 块 200 和客户端 3000 之间的连接状态, 态为枚举值 MonitorStates。连接状态机、读数
据状态机、反馈状态机和监测状态机均以单独的线程运行。

图 10 示出了连接状态机 (EpicsStateMachine) 的状态转换过程, 其中连接状态机
负责维持连接的状态。

图 11 示出了读数据状态机 (EpicsReader) 的状态转换过程, 其中读数据状态机负
20 责该 EPICS 通信端的数据包的接收和处理。在连接建立以后, EpicsReader 处于
STANDBY 状态。在这个状态中, EpicsReader 不停地执行 DoRead() 方法。在 DoRead()
方法中, EpicsReader 调用 EpicsConnection 的方法从 Epics 网络连接中读取数据包, 然
后调用 ProcessEpicsPacket() 处理读到的数据包。在 ProcessEpicsPacket() 方法中, 为了
封装的需要, 首先需要对 P_SHUTDOWN 类型的消息进行处理, 然后触发
25 GotPacketDelegate 事件。

图 12 示出了反馈状态机 (EpicsPoller) 的状态转换过程, 其中反馈状态机负责 EPICS
通信端的数据包的接收和处理。

图 13 示出了监测状态机 (EpicsMonitor) 的状态转换过程, 其中监测状态机定时
向通信的对方发送数据包, 以检测当前连接状况。在客户端 3000 和驱动系统 1000 成
30 功建立连接之后, EpicsMonitor 将处于 STANDBY 状态, 在这个状态, EpicsMonitor
定时向通信的对方发送 HeartBeat 型数据包, 以检测网络状态。当网络断开时, 监测状
态机发送 HeartBeat 型数据包将触发一个异常, 系统就会检测到 EPICS 协议通信模块
120 和客户端 3000 之间的网络连接已经断开。

图 14 示出了 EPICS 协议通信模块 120 的各个状态机的协作工作图。

35 配置模块 300 负责驱动系统 1000 中易变参数的处理。具体而言, 配置模块 300 用

于将驱动系统 1000 中经常变化的参数放置到变化参数配置文件中。在本发明的一个实施例中，变化参数配置文件以 XML 文件格式保存。

配置模块 300 主要配置如下两类参数：

- 5 a) EPICS 协议通信模块 200 和客户端 3000 之间的各个通道的配置参数：包括描述符列表、节点、各种通道等条目

1.1)描述符列表：保存驱动系统 1000 中各种描述符列表，以及每种列表所包含的条目。描述符列表示例如下：

//描述符列表，名称为“OpenStatus”，该类描述符可以有两种取值：Open 和 Close

<Descriptors Name="OpenStatus">

```
10      <Descriptor>
          <StringValue>Open</StringValue>
          <IntValue>1</IntValue>
      </Descriptor>
      <Descriptor>
15      <StringValue>Close</StringValue>
          <IntValue>2</IntValue>
      </Descriptor>
  </Descriptors>
```

1.2)节点：对节点进行配置，节点的主要参数是节点号，每个节点下面的通道被分为三类，分别为 ContinuousChannels、DiscreteChannels、和 StringChannels。在此三类 Xml 节点下的为具体的通道。节点示例如下：

```
  <node NodeNumber="1">
      <ContinuousChannels>
          <channel>
25          ...
          </channel>
      </ContinuousChannels>
      <DiscreteChannels>
          <channel>
30          ...
          </channel>
      </DiscreteChannels>
      <StringChannels>
          <channel>
35          ...
```

</channel>

</StringChannels>

</node>

1.3) 通道: 通道保存有每个 Channel 对象的关键信息。通道分为三类
5 ContinuousChannels、DiscreteChannels、和 StringChannels, 每类的关键信息都不同。

1.3.1) 离散型通道 DiscreteChannel:

离散型通道包括通道的 ID、通道号、通道名、描述符列表名、最小值、最大值、
优先级、读写类型。离散型通道示例如下:

<channel>

```

10         <ID>2003</ID>
           <ChannelNumber>3</ChannelNumber>
           <ChannelName>channel8</ChannelName>
           <DescriptorsName>OpenStatus</DescriptorsName>
           <Min>1</Min>
15         <Max>10</Max>
           <Privilege>3</Privilege>
           <Type>Write</Type>
           </channel>

```

1.3.2) 连续型通道 ContinuousChannel:

20 连续型通道包括通道的 ID、通道号、通道名、单位、最小值、最大值、精确度、
优先级、读写类型。连续型通道示例如下:

<channel>

```

           <ID>2001</ID>
           <ChannelNumber>1</ChannelNumber>
25         <ChannelName>channel6</ChannelName>
           <Units>0.1</Units>
           <Min>1.0</Min>
           <Max>10.0</Max>
           <Accuracy>0.1</Accuracy>
30         <Privilege>3</Privilege>
           <Type>Write</Type>
           </channel>

```

1.3.3) 字符串型通道 StringChannel:

字符串型通道包括通道的 ID、通道号、通道名、最大长度、优先级、读写类型。

35 字符串型通道示例如下:

```

<channel>
    <ID>2005</ID>
    <ChannelNumber>5</ChannelNumber>
    <ChannelName>channel10</ChannelName>
5    <MaxLength>20</MaxLength>
    <Privilege>3</Privilege>
    <Type>Write</Type>
</channel>

```

b) 其他可配置的易变参数包括:

10 1)PollingFrequency: 轮循通道时, 服务器(驱动系统 1000)向上层(客户端 3000)定时反馈通道值的时间间隔;

2)deviceTestFrequency: 服务器(驱动系统 1000)定时测试硬件设备 2000 的测试时间间隔;

3)IP: 服务器端(驱动系统 1000 端)的 IP 地址;

15 4)Port: 服务器端(驱动系统 1000 端)监听网络连接的端口;

5)MaxThreads: 驱动系统 1000 中存在的线程的最大数量;

6)ConcurrentThreads: 驱动系统 1000 中可以并行执行的线程数量。

在本发明的一个实施例中, 配置模块 300 可以进一步生成用于解析变化参数配置文件的键类 ConfigProvider。

20 键类 ConfigProvider 负责解析变化参数配置文件, 取得通道等配置参数, 保存到 ConfigProvider 的成员对象中。为了对配置文件解析的一致性, 驱动系统 1000 中只能有一个 ConfigProvider 对象, ConfigProvider 采用 Singleton 模式。在 ConfigProvider 的 Init 方法中对 XML 文件进行解析,

在对 XML 配置文件进行解析中, 关键是使用类 XmlNode 的如下几个方法或属性:

25 SelectSingleNode(string path): 获得满足路径 path 的第一个节点对象。

SelectNodes(string path): 获得满足路径 path 的所有节点对象组成的列表。

InnerText 属性: 该节点的文本内容, 如对于节点

<ChannelName>channel10</ChannelName>, 其 InnerText 就是"channel10"

Attributes 属性: 该节点的属性集合,

30 如对于节点<Descriptors Name="OpenStatus">, "Name"就是该节点的一个属性。

当驱动系统 1000 需要上述参数时, 可以在运行时通过配置模块 3000 获取这些参数。当驱动系统 1000 中上述参数经常变化时, 可以直接修改变化参数配置文件, 而不用重新编译整个驱动系统 1000, 这样大大提供了驱动系统 1000 的可移植性。

35 数据模块 400 用于存储来自 EPICS 协议通信模块 200 的通道数据。数据模块 400 可以存储驱动系统 1000 中常用的一些数据结构的抽象, 包括数据项 DataItem、通道

Channel 等。数据结构是一个完善的系统的基础，数据结构、数据对象设计的好坏直接关系到系统整体的运行效率和控制算法的实现。在数据结构、数据对象的设计过程中，要采用面向对象的相关技术，减小对象间的耦合性，提供数据的组织效率。

下面列出了几个主要类所包含的一些最基本的数据成员：

- 5 (1) 通道类型：标识一个通道是连续的、离散的或是字符串型的。

CONTINUOUS = 0, //连续型

DISCRETE = 1, //离散型

STRING = 2, //字符串型

- (2) 通道读写类型：标识一个通道是只读的、只写的或是读写的。

10 Read = 0, //只读型

Write = 1, //只写型

ReadWrite = 2, //读写型

(3) 通道抽象类：所有具体的通道类（离散型通道、连续型通道、字符串型通道）的父类，用于封装通道的常见操作。

- 15 由上述(1)(2)可知，根据通道相关的数据类型，通道可以分为：连续型，离散型和字符串型。根据通道的读写类型，通道可以分为：只读型，只写型和读写型。综合这两种分类，通道分为如下六种：连续型只读通道、连续型只写通道、连续型读写通道、离散型只读通道、离散型只写通道、离散型读写通道、字符串型只读通道、字符串型只写通道、字符串型读写通道。

- 20 在本发明实施例的驱动系统 1000 中，通道类 Channel 的子类只有三种：连续型通道 ContinuousChannel、离散型通道 DiscreteChannel 和字符串型通道 StringChannel。这三种类型都包含一个 ReadWriteTypes 型的成员，以表示该对象是只读型、只写型或是读写型通道。图 15 为通道类的继承类图。

- 在本发明实施例的驱动系统 1000 中，通道对象是底层硬件 I/O point 的一次封装。
25 每个通道分别对应底层的一个硬件设备，即对应一个硬件控制点。通道的值即为对应的 I/O point 的值。硬件设备 2000 作为通道 Channel 的一个成员。通道和硬件设备 2000 的关系如图 16 所示，其中，通道可以包括连续型通道、离散型通道和字符串型通道。图 17 示出了 AbstractDevice 类内部的函数调用关系，即通道读写操作的顺序。

- (4) 节点类：节点是指逻辑上或是物理上相关的 EPICS 通道的集合。节点类包括
30 多个节点，其中多个节点是连续型通道集合、离散型通道集合或字符串型通道集合。每个驱动至少要有一个节点。服务器（驱动系统 1000）驱动可以任意指定作为节点的 EPICS 通道集合。

public class DeviceNode //节点

{

- 35 private int nodeNum; //节点号

```
private int channelIndex; //通道索引号, 用于创建通道  
private Hashtable channels; //该节点的所有通道  
}
```

5 每当创建一个新的通道时, 就会把通道同时添加到通道工厂和该通道所属的节点中的集合里, 以方便对通道的管理可操作。

(5) 通道工厂类: 全局唯一实例, 管理和创建节点与通道: 采用 Singleton 模式, 以确保整个 Control IO 端只有一个工厂实例。这个实例负责驱动系统 1000 中所有节点和通道的创建, 并且有两个 Hashtable, 分别保存驱动系统 1000 中所有可用的节点和通道。

10 下面参考图 18 至图 21 描述设备驱动模块 500。设备驱动模块 500 和硬件设备 2000 进行通信, 处理设备操作请求, 向外提供设备相关接口。在标准化通用接口软件包中, 对于各类设备具有共性的读写操作均提供一致的接口, 因此需要对不同类型设备已有的驱动程序进行相应的封装。设备驱动模块 500 中包含一些设备抽象类, 对实际的硬件设备 2000 进行抽象, 封装了对常用硬件设备 2000 的操作方法。

15 如图 18 所示, 设备驱动模块 500 包括映射单元 510、读写单元 520、超时处理单元 530 和设备管理单元 540。

映射单元 510 用于对通道和硬件设备 2000 的读/写接口进行映射以对硬件设备 2000 进行读/写操作。具体而言, 映射单元 510 封装对实际硬件设备 2000 的抽象, 完成通道和实际硬件的 IO point 的映射。

20 读写单元 520 用于读取数据模块 400 存储的通道数据、配置模块 300 配置的变化参数以及硬件设备 2000 的数据, 并向硬件设备 2000 中写入数据。读写单元 520 对硬件设备 200 的读写操作的原则: 可以有多个客户同时对硬件设备 2000 进行读操作, 但每个时刻只能有一个客户对硬件设备 2000 进行写操作。这种原则是由读者/写者模型来保证。AbstractDevice 类是所有实际设备的父类, 在 AbstractDevice 内部提供了对设备
25 操作的抽象接口, 同时对设备操作超时等情况进行了处理。

超时处理单元 530 用于处理对硬件设备 2000 的读操作超时。图 16 示出了设备读操作超时处理的流程。设备读操作的超时, 包括两种情况:

1) 在对硬件设备 2000 的实际读操作之前, 如果有其他客户对硬件设备 2000 进行写操作, 则此时该客户需要等待, 出现超时。

30 这种超时通过读者/写者锁的 AcquireReaderLock 方法处理。

执行 S1901: 超时处理单元 530 判断在预定时间内是否获得读写锁, 如果在所述预定时间内未获得所述读写锁, 则执行 S1902; 否则执行 S1905。

S1902: 抛出超时异常;

S1903: 返回 False;

35 S1904: 释放读写锁。

2) 在对硬件设备 2000 进行实际操作时, 硬件设备 2000 的读操作可能要花费很长时间, 出现超时。对硬件设备 2000 进行实际的读写是通过异步委托机制在一个新的线程里完成的, 可以取得等待异步委托完成的 WaitHandle 句柄, 使用该句柄的 WaitOne() 方法处理设备操作超时。

5 S1905: 对硬件设备 2000 进行实际的读写是通过异步委托机制在一个新的线程 BeginInvoke () 里完成;

S1906: 取得等待异步委托完成的 WaitHandle 句柄, 判断是否超时, 如果是则执行 S1907, 抛出超时异常, 否则执行 S1908, 读取硬件设备 2000 的数据。

S1909: 返回 True, 执行 S1904, 释放读写锁。

10 AbstractDevice 类中对底层设备的实际读写操作是由 InternalReadDevice 函数和 InternalWriteDevice 函数处理的, 而高层的 Read 和 Write 函数封装了底层的这两个函数并且对超时等异常情况进行处理以供通道等对象调用。AbstractDevice 的子类, 即对各种实际的硬件设备 2000 的封装类, 必须重写 InternalReadDevice 方法和 InternalWriteDevice 方法以处理各种具体设备的读写情况。AbstractDevice 类内部的函数调用关系如图 17 所示。

设备管理单元 540 用于创建和维护硬件设备 2000。驱动系统 1000 中包括一个设备工厂类 DeviceFactory, 完成对驱动系统 1000 中所有硬件设备的创建和维护。图 20 示出了 DeviceFactory 的 UML 描述 (Singleton 模式)。

设备管理单元 540 的设备工厂类可以创建如下四类模拟设备:

20 随机模拟类设备 (RandomSimulatedDevice): 对所述随机模拟类设备的每次读操作均分别返回一个随机的正数;

连续型设备 (StaticContinuousDevice): 在所述连续型设备中的硬件设备保存一个双精度浮点型数值, 并对所述双精度浮点型数值进行读/写;

25 离散型设备 (StaticDiscreteDevice): 在所述离散型设备中的硬件设备保存一个整型 (int 型) 数值, 并对所述整型数值进行读/写;

字符串型设备 (StaticStringDevice): 在所述字符串型设备中的硬件设备保存一个字符串型 (string 型) 数值, 并对所述字符串型数值进行读/写。

30 为了提高驱动系统 1000 的可靠性和鲁棒性, 需要对各种异常进行处理。本发明实施例的驱动系统 1000 进一步包括异常处理模块 600, 用于在驱动系统 1000 抛出异常时, 接管所述异常, 并查找异常处理向量表以获得所述异常的异常处理函数, 根据所述异常处理函数对所述异常进行处理。其中, 异常处理模块 600 主要处理与底层数据结构以及与 Epics 协议相关的一些比较重要的异常。当驱动系统 1000 的某个部分出现异常时, 由异常处理模块 600 接管这个异常, 并对该异常进行相关的处理。

35 在异常处理模块 600 中包括一个相对独立的异常处理类 AlarmServer。AlarmServer 类采用 Singleton 模式。在驱动系统 1000 中只允许存在一个 AlarmServer 实例, 以便对

系统中的所有异常进行集中处理。AlarmServer 维护一张异常处理向量表。异常处理向量表保存了每种预定义的异常类，以及每种异常类型对应的异常处理函数。

如图 21 所示，当系统抛出特定的异常时，可以使用这个 AlarmServer 接管异常，AlarmServer 查找异常处理向量表。异常处理向量表类似 PC 底层的中断向量表，保存了各种可以处理的异常类型，以及每种异常类型对应的异常处理函数。当 AlarmServer 接收到一个 Alarm 时，将查询异常处理向量表，找到对应的处理函数。通过查找异常处理向量表获得这种异常的处理类的类型，进而对异常进行处理，并且进行日志记录等相关操作。其中，异常日志处理包括调用日志模块 700，进行相应的异常记录。日志模块 700 将在下文中进行详细说明。

在本发明的一个实施例中，驱动系统 1000 抛出异常主要有两种方式。图 22 示出了对两种异常的处理流程。

1) 对于一般的异常可以使用 throw new Exception (Message) 方式。采用这种方式抛出的异常在驱动系统 100 中显示地用 try{} catch 语句捕获。

2) 对于与底层数据结构以及与 Epics 通信协议相关的一些比较重要的异常，可以使用 AlarmServer 处理，AlarmServer 会查找异常处理向量表，找出预先指定的异常处理类来处理该种异常。在异常处理类里可以集中进行一些指定的处理，如日志记录，向通信对方发送消息等等。

本发明实施例的标准化底层控制驱动系统 1000 进一步包括日志模块 700，用于以日志的形式记录驱动系统 1000 运行过程中的信息。

日志模块 700 只有一个类 LoggingService，用户需要进行日志记录时，只需要调用 LoggingService 的成员函数就可以了。LoggingService 类的设计参考了 Adapter 模式和 Bridge 设计模式。LoggingService 的定义如下：

```
public static class LoggingService
{
    //创建日志记录对象
    static readonly ILog log = LogManager.GetLogger(typeof(LoggingService));

    //使用配置文件初始化 log4net 的日志模块
    static LoggingService()
    {
        XmlConfigurator.ConfigureAndWatch(new
        FileInfo(AppDomain.CurrentDomain.SetupInformation.ConfigurationFile));
    }

    //参考 Adapter 模式和 Bridge 模式,把调用转加给 log4net 的日志对象
```

```

        public static void Debug(object message)
        {
            log.Debug(message);
        }
5      .....
    }

```

日志模块 700 可以提供一个灵活高效的日志系统来记录驱动系统 1000 的运行情况，以便当驱动系统 1000 出现问题时可以及时发现问题所在，从而大大提高驱动系统 1000 的可维护性，并且也可以使用日志模块 700 进行程序的调试和维护

10 图 23 示出了驱动系统 1000 整体运行状态机模型。如图中所示，首先进行初始化驱动系统 1000，包括创建自定义通道、修改已有通道的读写委托、读取配置文件、创建通道和节点、初始化 EPICS 协议通信模块等任务。在初始化完成后，运行驱动系统 1000，包括运行系统框架，进入消息循环。在处理完成后，驱动系统 1000 停止处理并执行清理任务。在清理完成后，重新启动系统并执行新的处理任务，在处理完成后进行相关的清理任务。

在本发明的一个实施例中，驱动系统 1000 的整体启动和调度采用类 Core。

Core 类是整个驱动系统 1000 调度和使用的接口。在 Core 类中，主要执行下述任务：

- a)初始化各个任务对象的工厂，以在处理 Epics 数据包时创建相应的任务
- 20 b)初始化通道工厂，以在需要的时候创建通道和节点
- c)初始化 EpicsAdapter，以便使用 Epics 协议进行通信
- d)初始化 ConfigProvider，以便系统使用配置文件
- e)初始化设备工厂，以创建各种设备
- f)对 Epics 数据包进行处理

25 Core 类也采用了 Singleton 模式，驱动系统 1000 中只有一个 Core 型对象。

本发明实施例提供的用于集成电路制造设备的标准化底层控制驱动系统基于 Windows xp 操作系统，采用 Visual C#实现。标准化底层控制驱动系统可以使得集成电路制造设备中的工艺模块控制器(PMC)、集群控制器(CTC)和传输模块控制器(TMC)和外部硬件设备进行通信，高效调度和处理各个控制子系统对集成电路制造设备的各种硬件设备的大量操作请求。

35 在本说明书的描述中，参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特点包含于本发明的至少一个实施例或示例中。在本说明书中，对上述术语的示意性表述不一定指的是相同的实施例或示例。而且，描述的具体特征、结构、材料或者特点可以在任何的一个或多个实施例或示例中以合适的方式结合。

尽管已经示出和描述了本发明的实施例，对于本领域的普通技术人员而言，可以理解在不脱离本发明的原理和精神的情况下可以对这些实施例进行多种变化、修改、替换和变型，本发明的范围由所附权利要求及其等同限定。

5

10

15

20

权利要求书

1、一种标准化底层控制驱动系统，所述标准化底层控制驱动系统用于接收来自客户端的设备操作请求，并根据所述设备操作请求控制硬件设备的动作，所述标准化底层控制驱动系统包括：

中央控制模块，所述中央控制模块对所述标准化底层控制驱动系统进行初始化并创建所述标准化底层控制驱动系统的 TCP / IP 套接字以与所述客户端进行通信；

EPICS 协议通信模块，所述 EPICS 协议通信模块通过所述 TCP / IP 套接字建立与所述客户端的通信连接，并采用 EPICS 协议与所述客户端进行通信以接收来自所述客户端的设备操作请求及与所述客户端交换通道数据；

配置模块，所述配置模块用于生成变化参数配置文件以配置所述标准化底层控制驱动系统中的变化参数；

数据模块，所述数据模块用于存储所述来自所述 EPICS 协议通信模块的通道数据；和

设备驱动模块，所述设备驱动模块用于处理所述设备操作请求，并根据所述数据模块存储的通道数据以及所述配置模块配置的变化参数控制所述硬件设备。

2、如权利要求 1 所述的标准化底层控制驱动系统，其中，所述中央控制模块包括：

初始化单元，所述初始化单元用于对所述 EPICS 协议通信模块、所述配置模块、所述数据模块和所述设备驱动模块进行初始化，并创建所述 EPICS 协议通信模块的 TCP / IP 套接字；

测试单元，所述测试单元用于定时测试所述硬件设备的状态以更新与所述硬件设备相关的通道的参数；和

调度单元，所述调度单元用于对所述标准化底层控制驱动系统中的各个任务进行调度以控制所述各个任务独立执行。

3、如权利要求 1 所述的标准化底层控制驱动系统，其中，所述调度单元将所述各个任务放入任务优先级队列，并根据所述各个任务对应的通道的优先级通过线程池中的多个工作线程，并行执行属于同一优先级的多个任务。

4、如权利要求 2 所述的标准化底层控制驱动系统，其中，所述变化参数配置文件以 XML 文件格式保存。

5、如权利要求 4 所述的标准化底层控制驱动系统，其中，所述变化参数配置文件中的变化参数包括：

所述 EPICS 协议通信模块和所述客户端之间的各个通道的配置参数、所述标准化底层控制驱动系统向所述客户端定时反馈通道值的时间间隔、所述标准化底层控制驱动系统定时测试所述硬件设备的测试时间间隔、所述标准化底层控制驱动系统的 IP 地址、所述标准化底层控制驱动系统端监听网络连接的端口、所述标准化底层控制驱动

系统中存在的线程的最大数量和所述标准化底层控制驱动系统中并行执行的线程数量。

6、如权利要求 1 所述的标准化底层控制驱动系统，其中，所述配置模块进一步生成用于解析所述变化参数配置文件的关键类。

5 7、如权利要求 1 所述的标准化底层控制驱动系统，其中，在所述中央处理模块初始化所述标准化底层控制驱动系统后，所述 EPICS 协议通信模块监听来自所述客户端的连接请求，在接收到来自所述客户端的连接请求后，建立与所述客户端的通信连接。

8、如权利要求 7 所述的标准化底层控制驱动系统，其中，所述 EPICS 协议通信模块包括：

10 连接状态机，所述连接状态机用于维持所述 EPICS 协议通信模块与所述客户端之间的连接的状态；

读数据状态机，所述读数据状态机用于读取并处理所述客户端与所述标准化底层控制驱动系统之间在通信中传输的通道数据；

15 反馈状态机，所述反馈状态机用于在所述客户端向所述标准化底层控制驱动系统查询通道值时，向所述客户端返回请求的所述通道值，并在所述通道值变化时，向所述客户端发送通道值变化通知；和

监测状态机，所述监测状态机用于定时向所述客户端发送数据包以检测所述 EPICS 协议通信模块和所述客户端之间的连接状态，

20 其中，所述连接状态机、读数据状态机、反馈状态机和监测状态机均以单独的线程运行。

9、如权利要求 7 所述的标准化底层控制驱动系统，其中，当所述监测状态机发送的数据包触发到异常时，则检测到所述 EPICS 协议通信模块和所述客户端之间通信连接断开。

25 10、如权利要求 1 所述的标准化底层控制驱动系统，其中，所述数据模块存储的通道数据包括：

通道类型，所述通道类型包括连续型、离散型、字符串型；

通道读写类型，所述通道读写类型包括只读型、只写型、读写型；

通道类型的父类，所述通道类型的父类用于封装所述的通道的操作；

30 节点类，所述节点类为逻辑上或物理上相关的 EPICS 通道的集合，包括多个节点，所述多个节点是连续型通道集合、离散型通道集合或字符串型通道集合；和

通道工厂类，所述通道工厂类用于管理和创建节点和通道。

11、如权利要求 10 所述的标准化底层控制驱动系统，其中，所述通道包括：

35 连续型只读通道、连续型只写通道、连续型读写通道、离散型只读通道、离散型只写通道、离散型读写通道、字符串型只读通道、字符串型只写通道和字符串型读写通道。

12、如权利要求 10 所述的标准化底层控制驱动系统，其中，每个通道对应于一个硬件设备。

13、如权利要求 10 所述的标准化底层控制驱动系统，其中，所述标准化底层控制驱动系统包括至少一个节点，其中所述标准化底层控制驱动系统任意指定作为所述节点
5 的 EPICS 通道集合。

14、如权利要求 13 所述的标准化底层控制驱动系统，其中，当创建一个通道时，将所述通道同时添加到所述通道工厂类和所述通道所属的节点的 EPICS 通道集合中。

15、如权利要求 1 所述的标准化底层控制驱动系统，其中，所述设备驱动模块包括：

10 映射单元，所述映射单元用于对所述通道和所述硬件设备的读/写接口进行映射以对所述硬件设备进行读/写操作；

读写单元，所述读写单元用于读取所述数据模块存储的通道数据、所述配置模块配置的变化参数以及所述硬件设备的数据，并向所述硬件设备中写入数据；

15 超时处理单元，所述超时处理单元用于处理对所述硬件设备的读操作超时；和设备管理单元，所述设备管理单元用于创建和维护所述硬件设备。

16、如权利要求 15 所述的标准化底层控制驱动系统，

其中，所述超时处理单元判断在预定时间内是否获得读写锁，

如果在所述预定时间内未获得所述读写锁，则抛出超时异常，

如果在所述预定时间内获得所述读写锁，则通过异步委托机制在另一个线程里

20 判断是否超时，如果是则抛出超时异常并释放所述读写锁，如果否则读取所述硬件设备的数据并释放所述读写锁。

17、如权利要求 15 所述的标准化底层控制驱动系统，其中，所述设备管理单元将所述硬件设备分为四类，包括：

25 随机模拟类设备，所述随机模拟类设备中的硬件设备的每次读操作均返回一个随机的正数；

连续型类设备，所述连续型设备中的硬件设备保存一个双精度浮点型数值，并对所述双精度浮点型数值进行读/写；

离散型设备，所述离散型设备中的硬件设备保存一个整型数值，并对所述整型数值进行读/写；

30 字符串型设备，所述字符串型设备中的硬件设备保存一个字符串型数值，并对所述字符串型数值进行读/写。

18、如权利要求 16 所述的标准化底层控制驱动系统，其中，进一步包括异常处理模块，所述异常处理模块用于在所述标准化底层控制驱动系统抛出异常时，查找异常处理向量表以获得所述异常的异常处理函数，并根据所述异常处理函数对所述异常
35 进行处理。

19、如权利要求 18 所述的标准化底层控制驱动系统，其中，所述异常处理向量表包括异常类型以及每种异常类型对应的异常处理函数。

20、如权利要求 1 所述的标准化底层控制驱动系统，其中，进一步包括日志模块，所述日志模块以日志的形式记录所述标准化底层控制驱动系统运行过程中的信息。

5

10

15

20

25

30

35

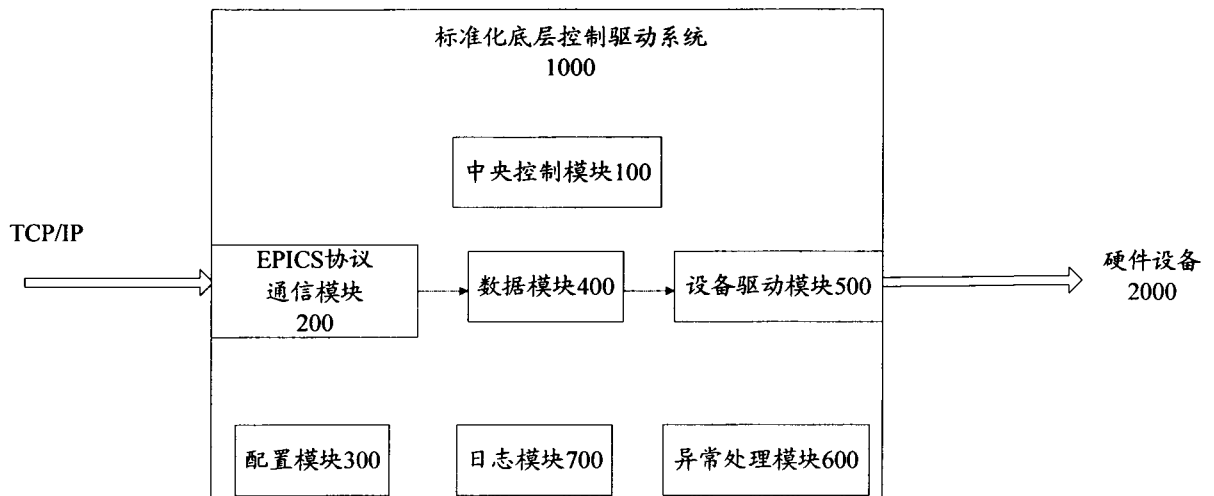


图 1

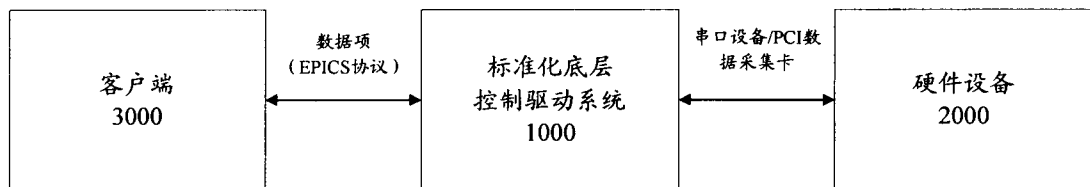


图 2

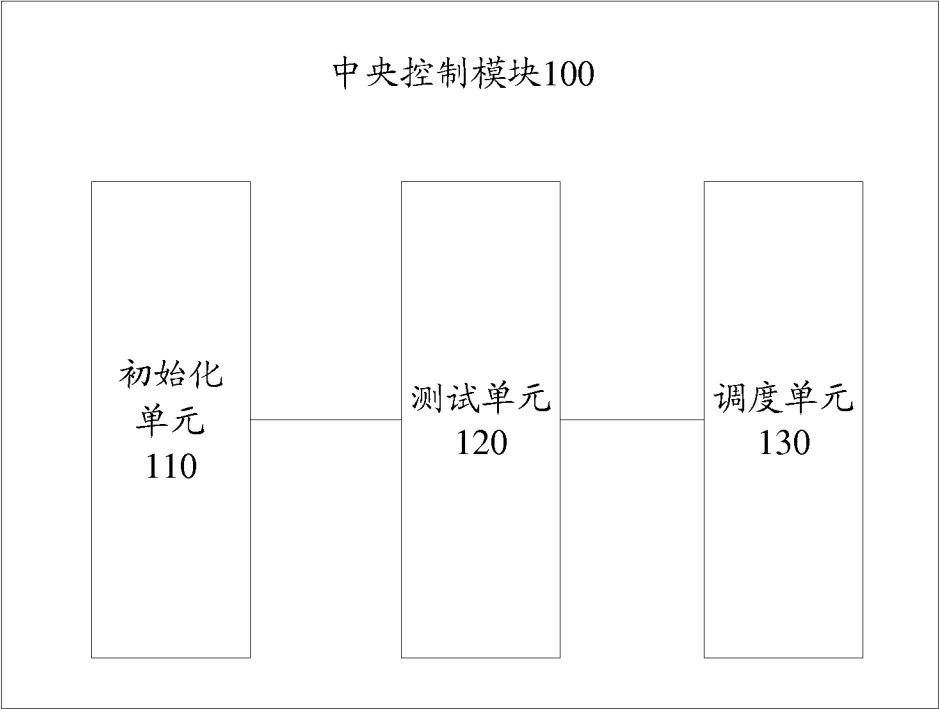


图 3

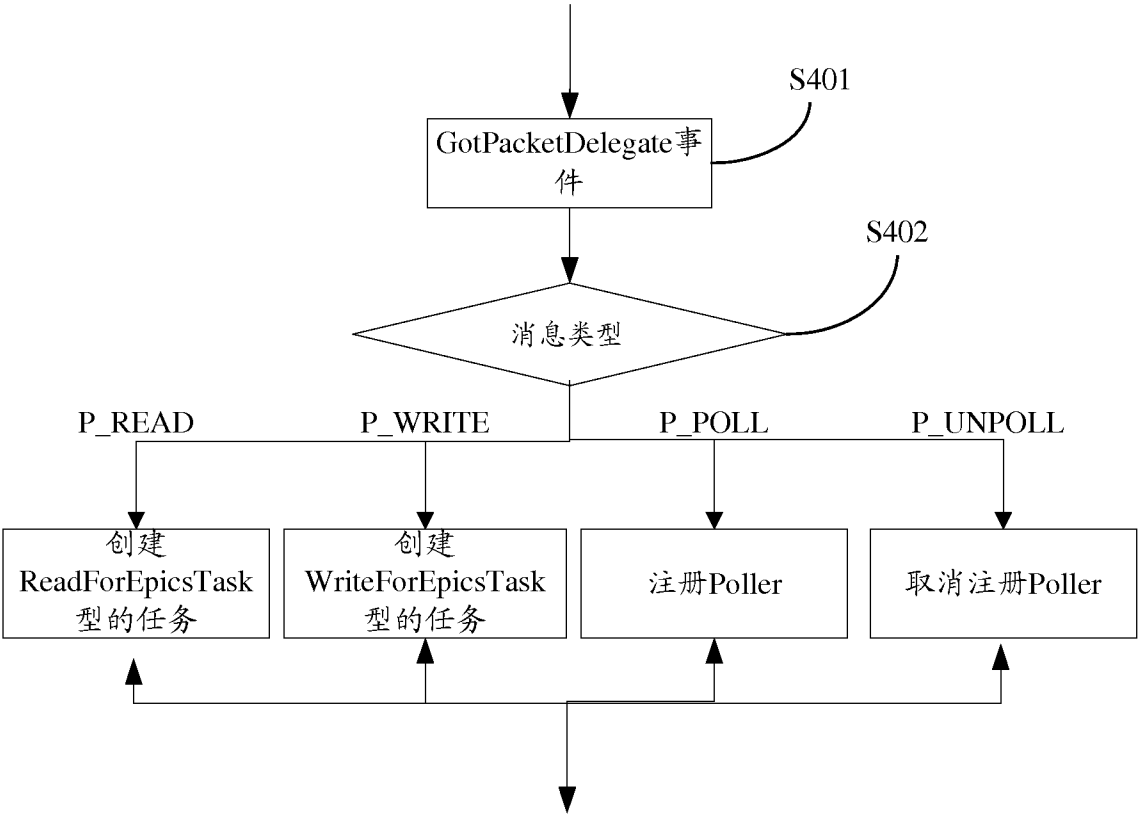


图 4

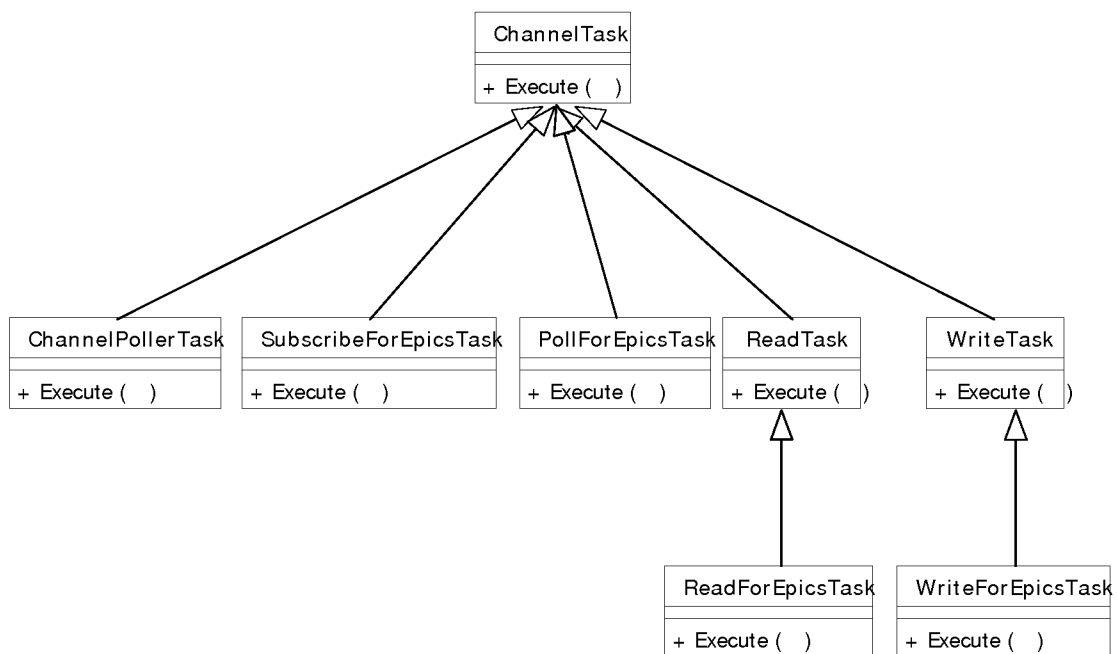


图 5

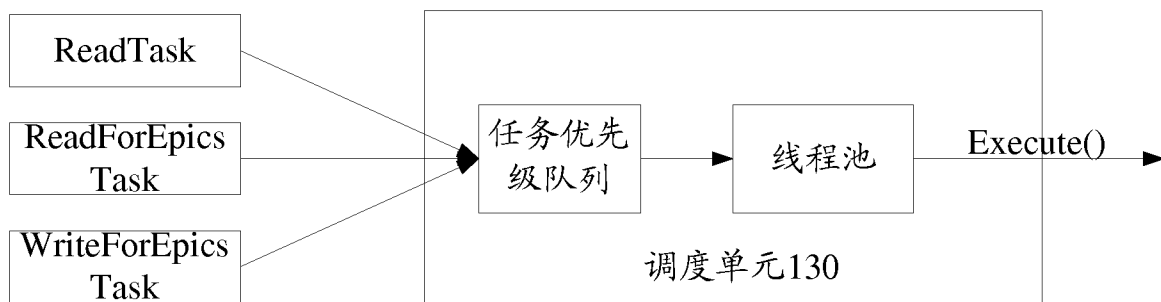


图 6

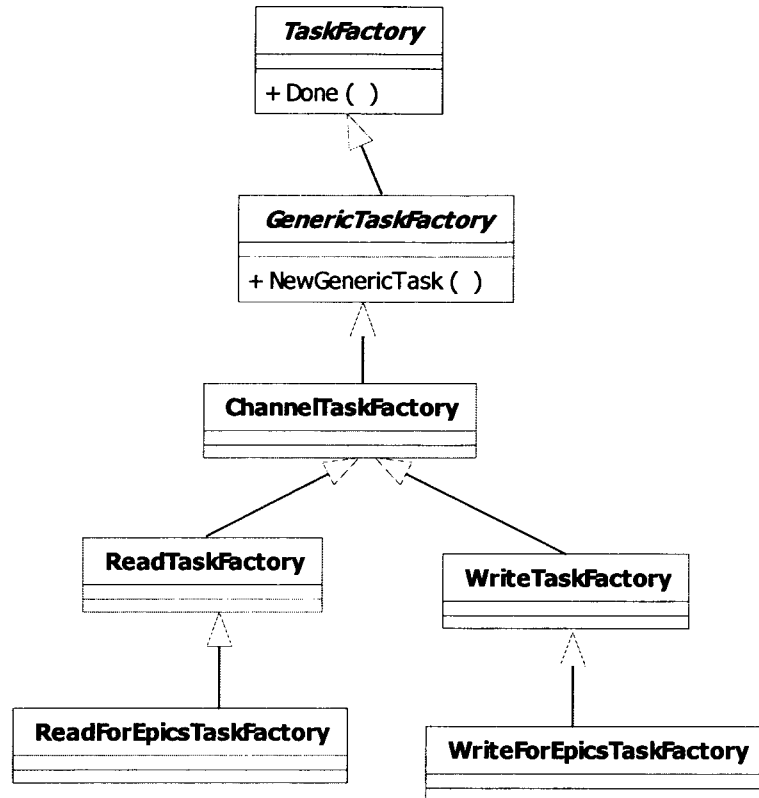


图 7

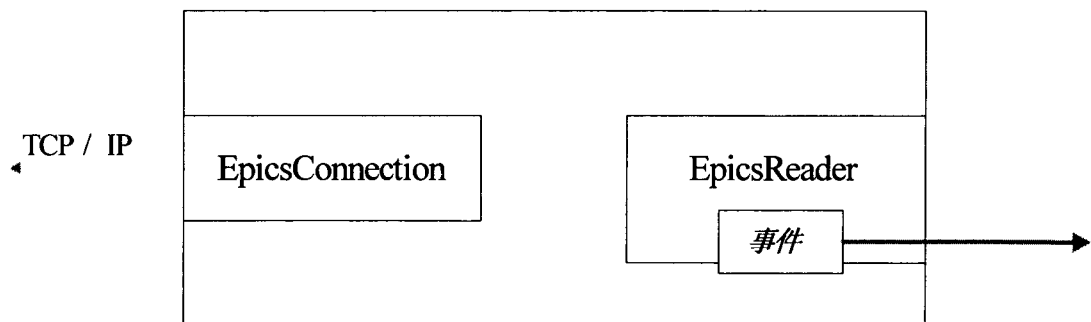


图 8

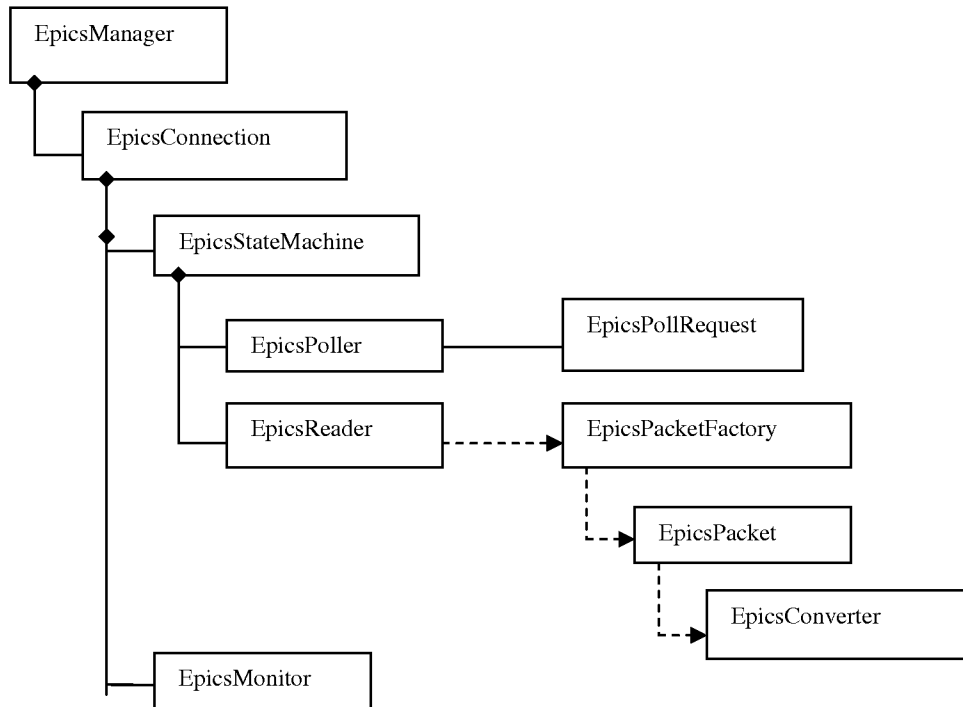


图 9

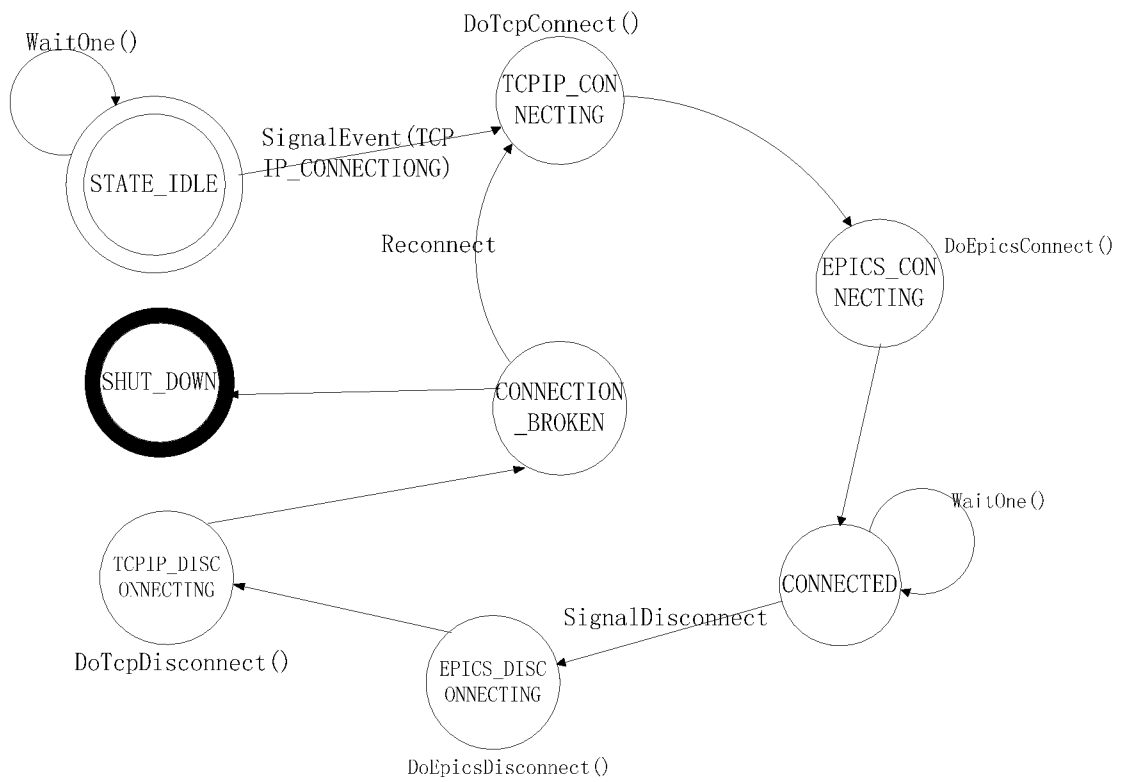


图 10

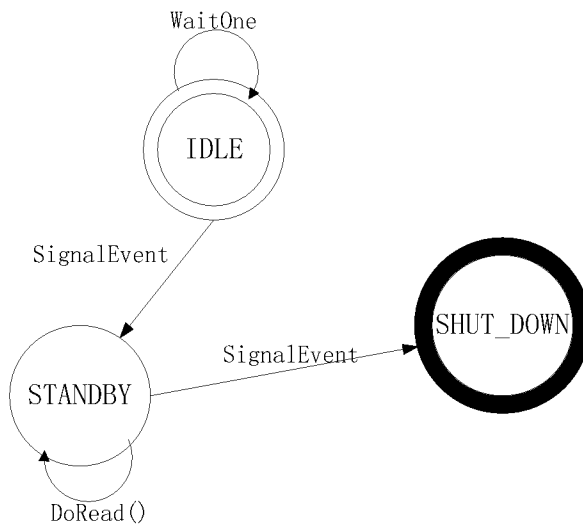


图 11

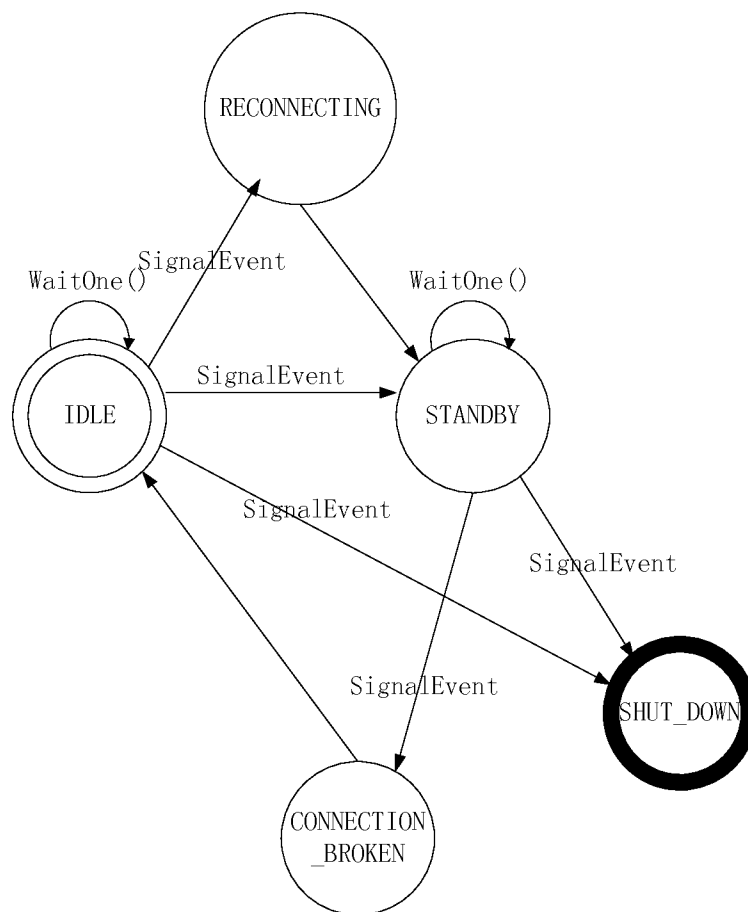


图 12

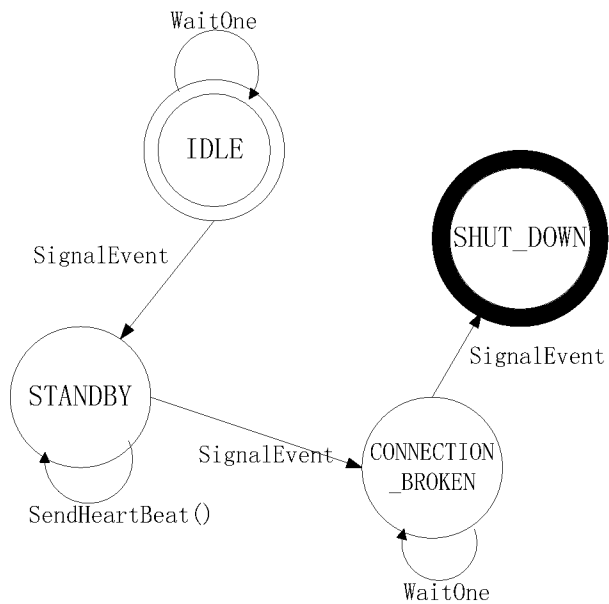


图 13

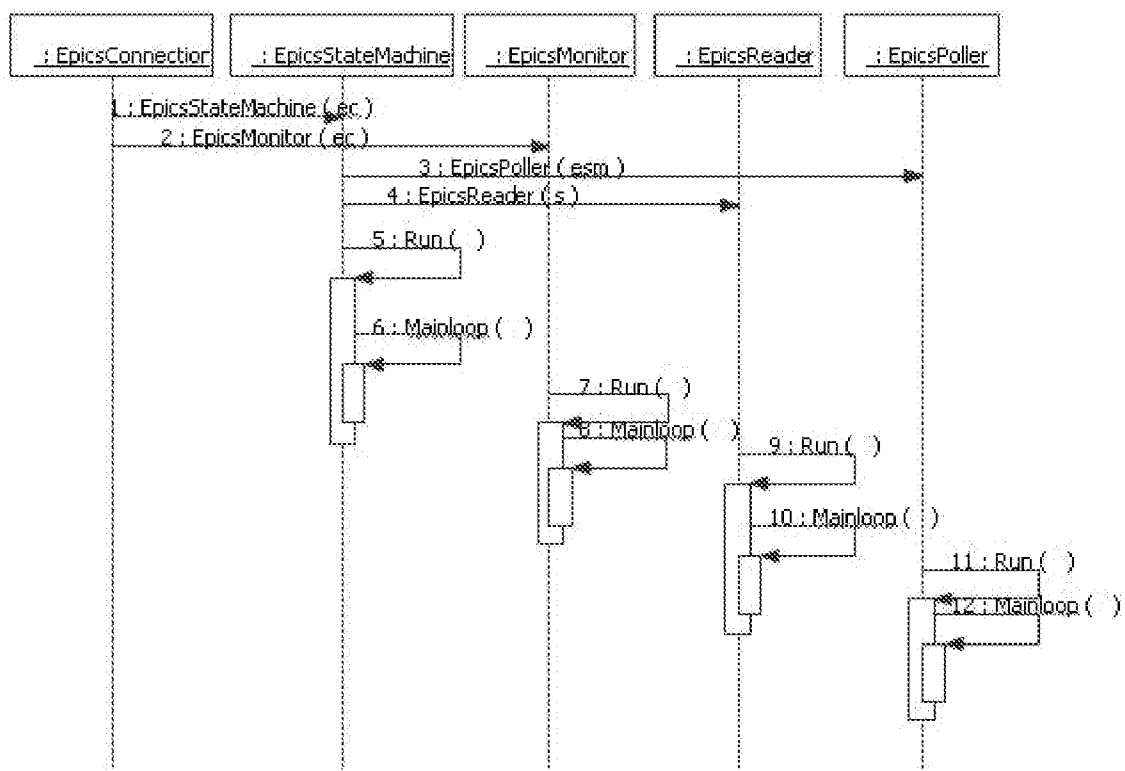


图 14

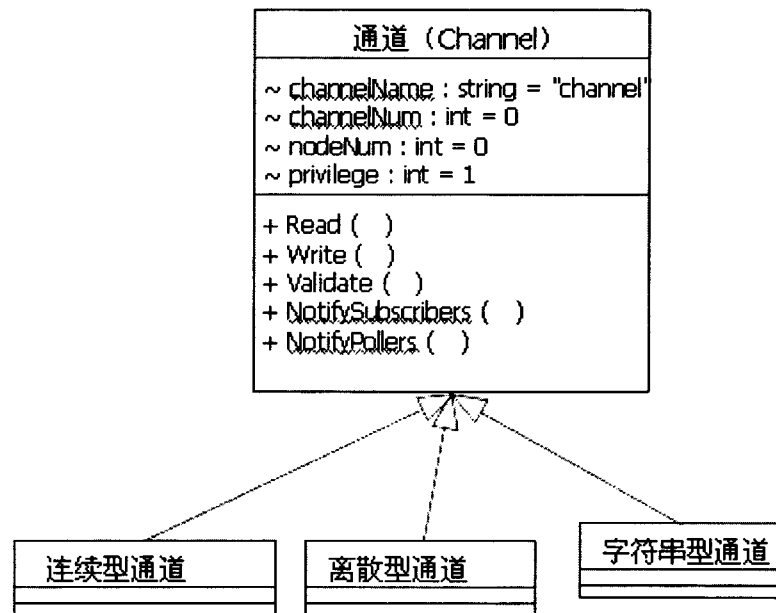


图 15

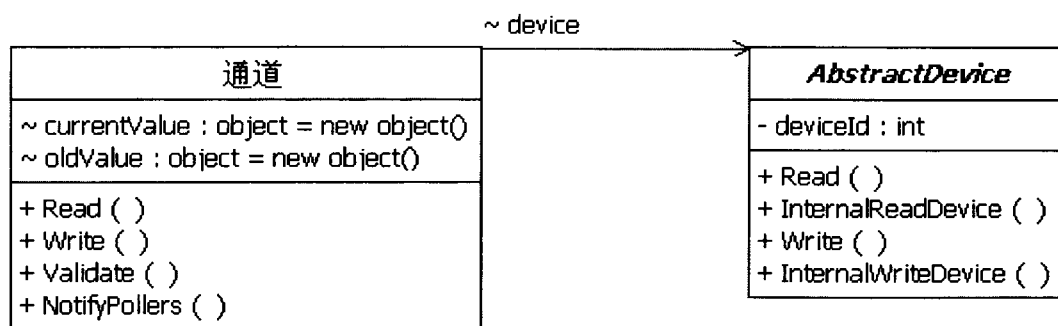


图 16

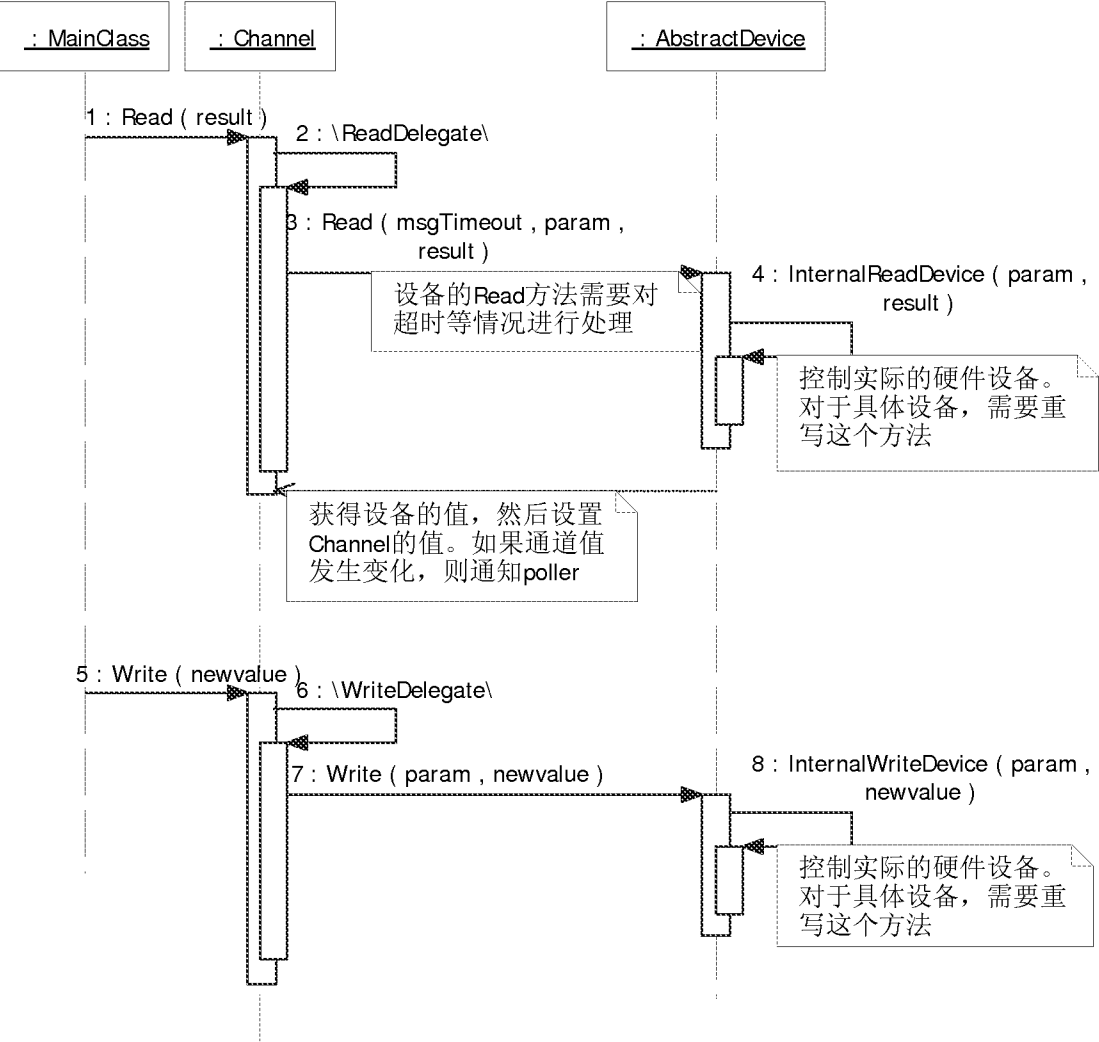


图 17

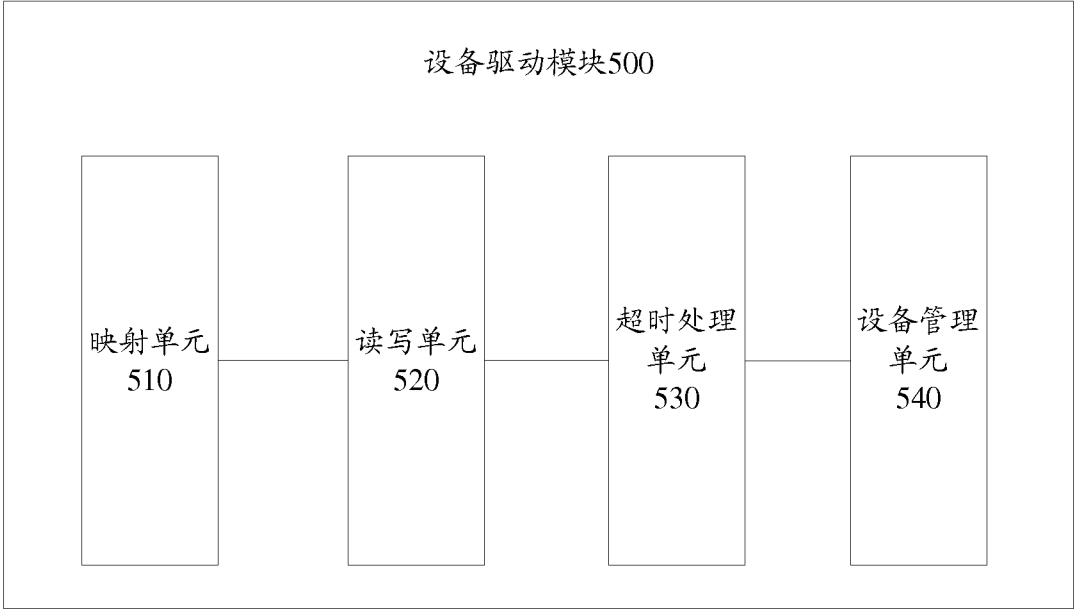


图 18

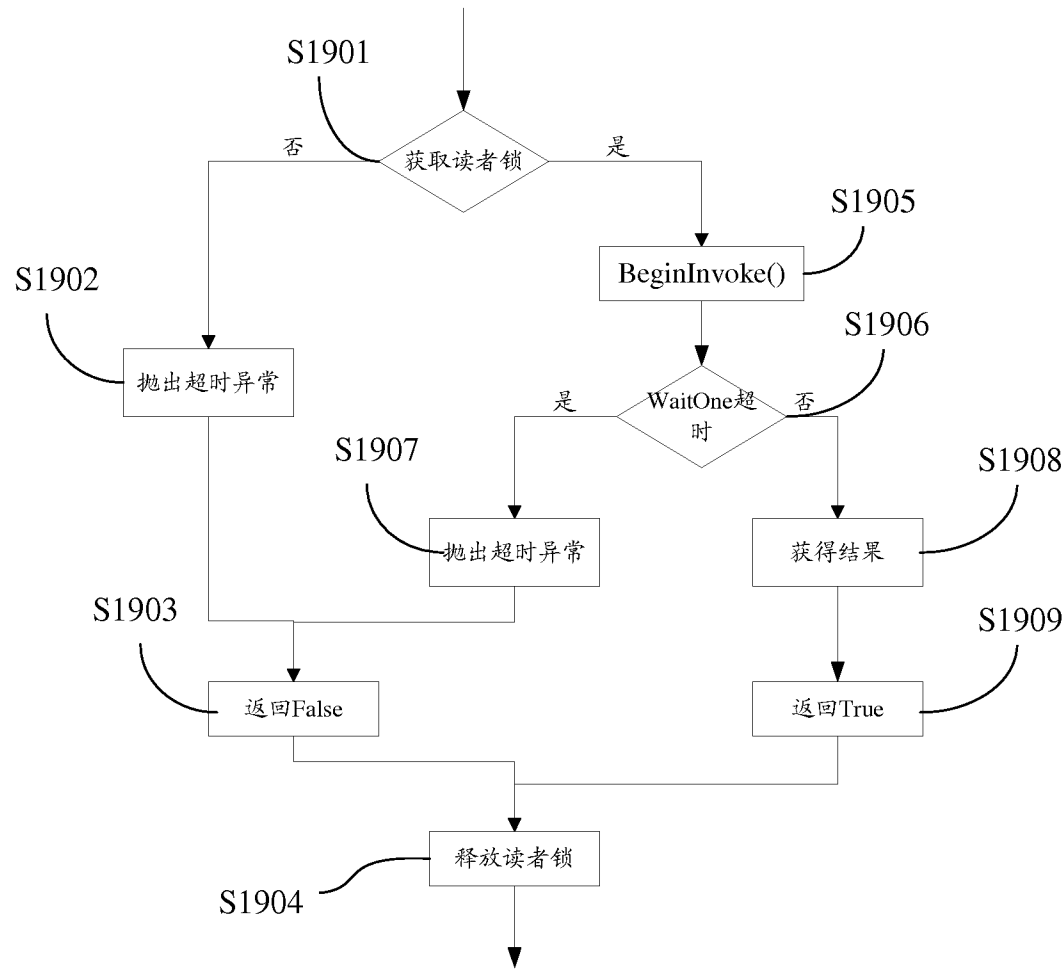


图 19

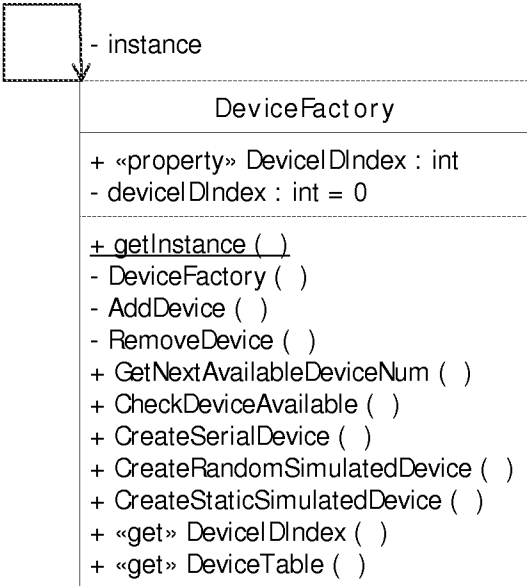


图 20

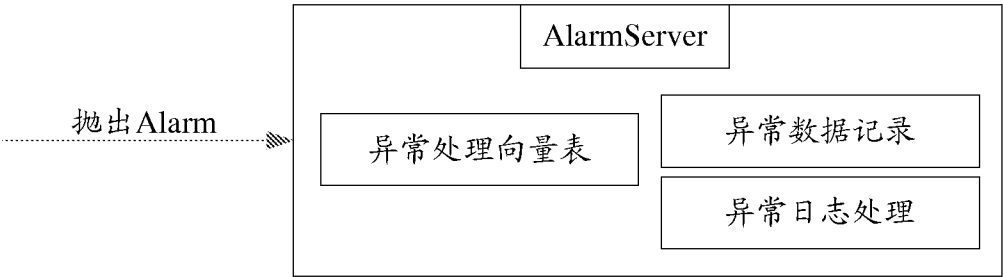


图 21

12/12

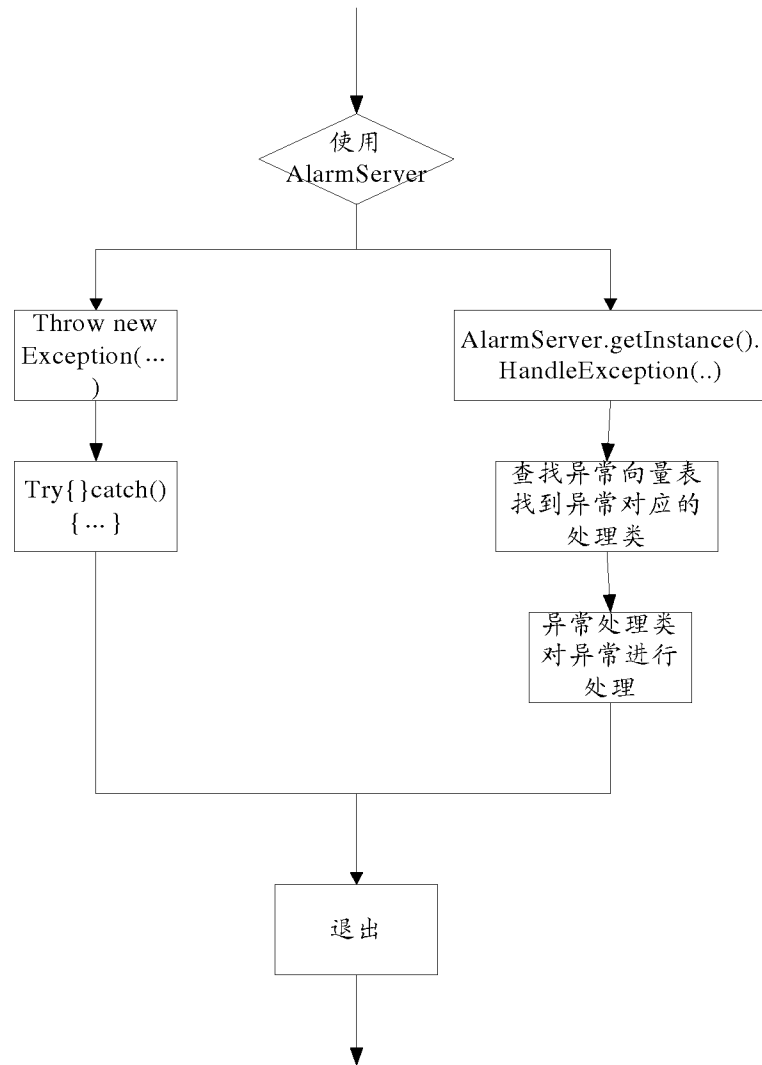


图 22

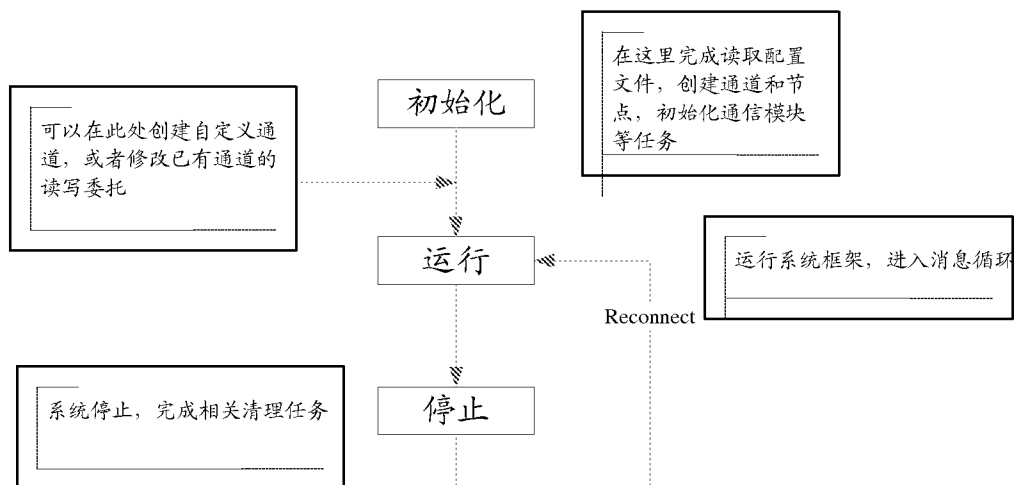


图 23

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2011/081952

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/06 (2006.01) i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: H04L G06F

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

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

CPRSABS, VEN, CNKI, IEEE: EPICS, socket, driv+, channel, session, parameter, data, hardware, device, circuit, apparatus, request+, command, instruct+, operat+, control+, connect+, communicat+

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN101848152A (HUAWEI DEVICE CO LTD) 29 Sep. 2010 (29.09.2010) see the description paragraphs 0086-0093	1-20
A	CN101895544A (SHENZHEN TP LINK TECHNOLOGY CO LTD) 24 Nov. 2010 (24.11.2010) see the whole document	1-20
A	US20110004790A1 (HEIDASCH R et al) 06 Jan. 2011 (06.01.2011) see the whole document	1-20
A	SHEN Guobao et al. PC/104 based EPICS Device Controller. Nuclear Electronics & Detection Technology. March 2003, Volume 23, Issue 2, pages 151-154, ISSN 0258-0934	1-20
PX	CN102185860A (UNIV QINGHUA) 14 Sep. 2011 (14.09.2011) see the claims 1-20	1-20

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

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

Date of the actual completion of the international search 02 Feb. 2012 (02.02.2012)	Date of mailing of the international search report 23 Feb. 2012 (23.02.2012)
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10)62019451	Authorized officer WANG, Xinyi Telephone No. (86-10) 62412027

International application No.
PCT/CN2011/081952

Form PCT/ISA/210 (patent family annex) (July 2009)

A. 主题的分类

H04L 29/06 (2006.01) i

按照国际专利分类(IPC)或者同时按照国家分类和 IPC 两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

IPC: H04L G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称, 和使用的检索词(如使用))

CPRSABS, VEN, CNKI, IEEE: 套接字, EPICS, 驱动, 通道, 线程, 参数, 数据, 硬件, 设备, 电路, 器件, 请求, 命令, 指令, 操作, 控制, 连接, 通信, socket, driv+, channel, session, parameter, data, hardware, device, circuit, apparatus, request+, command, instruct+, operat+, control+, connect+, communicat+

C. 相关文件

类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
A	CN101848152A (华为终端有限公司) 29.9 月 2010 (29.09.2010) 参见说明书第 0086-0093 段	1-20
A	CN101895544A (深圳市普联技术有限公司) 24.11 月 2010 (24.11.2010) 参见全文	1-20
A	US20110004790A1 (HEIDASCH R 等) 06.1 月 2011 (06.01.2011) 参见全文	1-20
A	沈国保等. 基于 PC/104 的 EPICS 设备控制器. 核电子学与探测技术. 3 月 2003, 第 23 卷, 第 2 期, 第 151-154 页, ISSN 0258-0934	1-20
PX	CN102185860A (清华大学) 14.9 月 2011 (14.09.2011) 参见权利要求 1-20	1-20

☐ 其余文件在 C 栏的续页中列出。☒ 见同族专利附件。

* 引用文件的具体类型:

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件(如具体说明的)

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期

02.2 月 2012 (02.02.2012)

国际检索报告邮寄日期

23.2 月 2012 (23.02.2012)

ISA/CN 的名称和邮寄地址:

中华人民共和国国家知识产权局

中国北京市海淀区蓟门桥西土城路 6 号 100088

传真号: (86-10)62019451

受权官员

王心一

电话号码: (86-10) 62412027

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2011/081952

检索报告中引用的 专利文件	公布日期	同族专利	公布日期
CN101848152A	29.09.2010	无	
CN101895544A	24.11.2010	无	
US20110004790A1	06.01.2011	无	
CN102185860A	14.09.2011	无	