第 32 卷 第 22 期 计 算 机 工 程 2006 年 11 月
Vol.32 No.22 Computer Engineering November 2006
·工程应用技术与实现· 文章编号：1000—3428(2006)22—0218—04 文献标识码：A 中图分类号：TP311

基于 QNX 的操作监控平台的设计与实现

段菖蒲，傅 鹏
(中国科学院等离子体物理研究所，合肥 230031)

摘 要：根据 QNX 的 Photon 微图形用户接口(microGUI)的运行机理，采用自定义事件、伪终端和 telnet 技术、动态分配内存等方法，解决了 EAST 极向场电源控制系统实时网络通信、启动和终止远程进程、动态绘制波形等关键问题，实现了一个友好、高效的操作监控平台，为 EAST 极向场电源的稳定运行提供了有力保障。

关键词：超导托卡马克聚变实验装置；QNX；Photon；远程进程；事件空间；事件；区域

Design and Implementation of Operation Monitor Platform Based on QNX OS

DUAN Changpu, FU Peng
(Institute of Plasma Physics, Chinese Academy of Sciences, Hefei 230031)

【Abstract】 According to Photon microGUI's operational mechanism, the paper uses a number of techniques such as user-defined events, pseudo-terminal and telnet techniques, dynamic allocation memory method and so on. With these techniques, it solves the key questions of the operation monitor platform such as starting and canceling remote processes, dynamic rendering waves etc., offering users a friendly and simple operation and monitor platform, which provides the powerful guarantee for the EAST PF power supply stable working.

【Key words】 Experimental advanced super-conducting Tokamak (EAST); QNX; Photon; Remote processes; Event space; Event; Region

超导托卡马克聚变实验装置 (Experimental Advanced Super-conducting Tokamak, EAST)是国家“九五”重大科学工程。极向场电源系统是托卡马克装置的核心子系统之一，对于装置运行的性能与安全，物理实验的成败与效率，有着至关重要的作用。极向场电源控制系统是极向场电源系统的软件组成部分，该系统分为 3 层：Windows 监测层，QNX 实时控制层、现场总线执行层。其中，QNX 实时控制层是整个电源控制系统的灵魂，采用 QNX6.20 实时操作系统作为开发平台，由 1 台反馈运算节点(NodeFB)，1 台数据库节点(DataBase)及 12 台电源控制器(alpha)节点组成。

为了节约开支，整个控制系统采用盲机运行。对于如此庞大复杂的控制系统来说，如何在盲机运行的情况下方便地进行系统运行前各种参数设置，远程启动、终止各个节点上的对应进程，并且能够随时直观地了解整个系统的运行状态，以及在其出现异常情况时，能够及时作出响应，保证整个系统始终能够可靠地运行，成为 EAST 极向场电源系统迫切需要解决的难题之一。

1 Photon microGUI 的体系结构

QNX 构建了一个 Photon microGUI 窗口系统，其结构如图 1 所示。Photon(microGUI kernel)是该结构的核心(微内核)。其它模块在其之外，通过多种通信方式与其进行通信。

Photon 最重要的特点在于表现图形应用程序的方法。它提供了一个抽象的三维事件空间(event space)，如图 2 所示。图 2 中的每一个矩形被称作区域(region)。这些区域能够发生和接收特定的对象，这些对象被称为事件(events)，如图 2 中箭头所示，它们像光子(photon)一样，穿梭在事件空间中的区域之间，当与空间中的区域发生碰撞，对该事件类型敏感的区域就会在拥有该区域的应用程序界面上调用相应的程序，实现与用户的交互。该结构由 Photon Manager 进程负责管理。

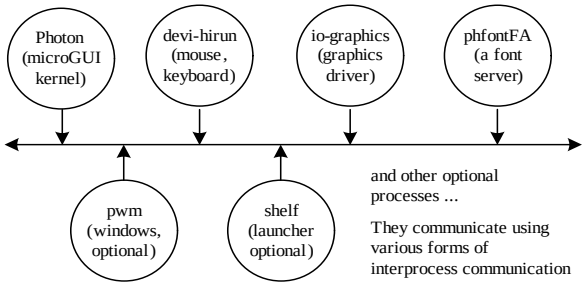

图 1 Photon microGUI 结构

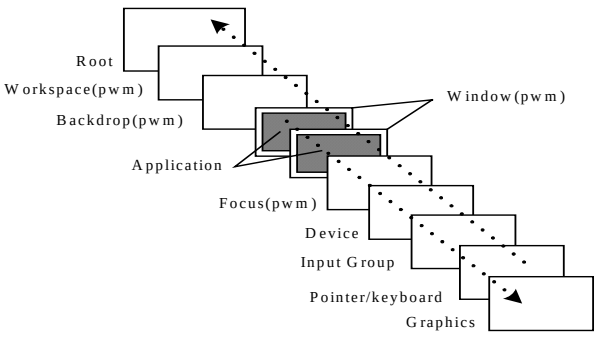

图 2 事件空间结构

基金项目：国家发展计划委员会基金资助子项目(投资(1998)1303)
作者简介：段菖蒲(1977－)，男，博士生，主研方向：EAST 极向场电源实时操作监控系统；傅 鹏，研究员、博导
收稿日期：2006-02-13 **E-mail：**changpu@ipp.ac.cn

图 2 中的 Device 区域归 Photon Manager 所有，它将事件空间分为两个部分 驱动程序部分(Device 区域至 Graphics 区域部分)和应用程序部分(Device 区域至 Root 区域部分)。Photon Manager 利用 Device 区域聚焦鼠标和键盘事件。比如，鼠标在屏幕上移动时，鼠标驱动会发出一个未被聚焦(未加工)的事件(raw event)，当该事件到达 Device 区域时，Photon Manager 会截获该事件并且在 Photon 坐标空间为其分配一个位置，这样该事件即被聚焦。然后 Photon Manager 重发该事件，此时，该事件会向两个方向运动，当到达 Graphics 区域时，绘图驱动模块收到该事件，调用绘图驱动程序在屏幕上刷新鼠标的位置。

2 操作监控平台的总体设计

2.1 平台的体系结构

电源控制系统的实时控制层的体系结构如图 3 中实线部分所示。

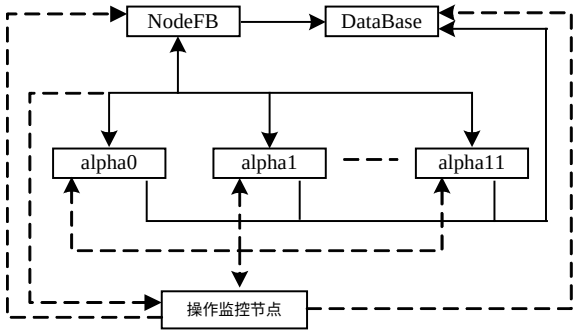

图 3 操作监控平台体系结构

NodeFB 负责在一个控制周期(1ms)内将主回路电压电流值及相关控制信号组播给 12 套电源控制器节点(alpha)，并接收 12 套 alpha 节点的状态返回信号。DataBase 节点负责对主回路信号、所有 alpha 节点以及其它相关硬件当前状态进行实时的备份。

引入操作监控节点后，它与电源控制系统各节点的通信如图 3 中虚线部分所示。通信过程包括以下两部分：(1)发送控制指令。操作监控节点负责将系统运行开始前的各种参数及相关的控制指令依次发送给 NodeFB、12 套 alpha 节点及数据库节点。(2)监测系统运行状态。操作监控节点加入 alpha 节点所在的组播组，接收 NodeFB 的组播。同时也接收 12 套 alpha 节点运行时的状态信号、电压电流及晶闸管角度信号以及系统故障信号等。

2.2 平台的主要功能

(1)控制系统运行开始前的各种设备参数设置；

(2)远程启动不同节点上的对应进程、协调各节点的运行状态、运行过程中强行终止某个/全部进程、运行结束后检查各个进程是否已经正常、安全退出；

(3)模拟总控，进行电源系统的本地调试；

(4)实时监控整个电源控制系统的运行状态，出现异常情况，及时作出响应；

(5)放电过程中，实时绘制电压、电流波形，随时查看整个控制系统的日志文件；

(6)访问数据库节点，对其进行各种本地进行的操作。

3 关键问题及解决方法

3.1 网络通信问题

根据物理实验的需求，EAST 极向场电源控制系统的控制周期定为 1ms，即在一个控制周期(1ms)内完成对每套电源设备的信号采集、运算、通信、执行、数据备份等工作。通过测量分析，得出目前控制系统每个控制周期内实际花费的时间为 573.33μs, 小于 1ms 控制周期。因此，引入操作监控节点，它与控制系统的通信是否会影响控制系统的实时性，是首先要考虑和解决的重点问题。

选择合适的网络通信协议是保证通信实时性的关键。在此，可供选择的通信协议有 UDP， TCP 和 QNX 操作系统提供的远程消息传递(Remote Message Passing, RMP)3 种。我们分别从原理上分析了这 3 种协议。

(1)TCP 协议是面向连接的协议，它实现了面向连接的、端对端的可靠的流传输。效率不高，占用资源较多。显然，它不适合于实时通信的场合。

(2)消息传递是 QNX 中进程间通信的最基本的形式。它是一个同步的 Send/Receive/Reply 的过程：

1)进程 A 通过向内核发出 MsgSend()请求向进程 B 发送一条消息。此时，进程 A 变成 Send 阻塞状态，直到进程 B 发出 MsgReceive()接收这一消息；

2)进程 A 转变为 Reply 阻塞状态；

3)进程 B 对接收到的消息进程处理，然后调用 MmsgReply()回答。回答消息被内核直接拷贝到进程 A MsgSend()所指定的缓冲区，使得进程 A 解除 Reply 阻塞状态，进入到 Ready 状态。

由上述过程可知，一旦进程 A 调用了 MsgSend()向进程 B 发送了一条消息，在未收到进程 B 的应答之前，进程 A 将一直处于 Send 阻塞状态而无法继续执行。同样，一旦进程 B 调用 MsgReceive()等待接收消息，在未收到进程 A 发送的消息之前，它也将一直处于 Receive 阻塞状态而无法继续执行。因此，它也不适合用于实时通信的场合。

(3)UDP 是一种面向无连接的、提供高效率数据传输服务的协议。因此，选用 UDP 协议作为操作监控节点与控制系统之间的网络通信协议。并对其通信性能进行了 100 000 次的统计，结果如图 4 所示。

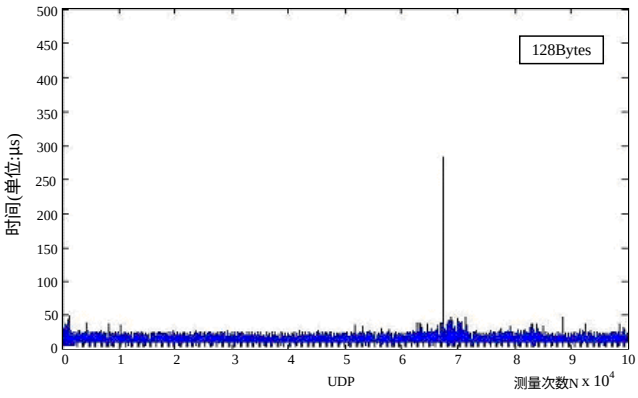

图 4 UDP 通信性能测试结果

从图 4 中可以看出，操作节点与控制系统通信所花时间大约为 20μs。因此，引入操作节点后，控制系统每个控制周期各进程所花费时间为 597.33μs，仍能够满足 1ms 控制周期的需要。

3.2 远程启动其它节点上的进程

电源控制系统一直停留在终端命令行的操作方式，即每次启动整个系统时，必须要逐一登录到每台节点，输入该节点上程序的路径及可执行文件名方可启动该节点上的程序。

这就需要用户熟记每台节点上的程序的路径、文件名以及许多相关的命令。对于如此庞大复杂的控制系统来说，采用这样的运行方式，对于用户、尤其是不熟悉该操作系统的用户来说，难以掌握和使用。对此，我们采用了伪终端与 telnet 技术，实现了“一键通”的功能，即用户只需动一下鼠标即可依次启动或终止远程节点上的进程。

使用 telnet，可以登录远程主机，但仍需要通过键盘录入命令以及远程主机的地址。登录成功后，如果要运行远程主机的程序，同样需要录入程序的路径及可执行文件名。显然这不符合我们的目标。PtTty 控件提供了解决此问题的方法。

PtTty 是 PhAB 图形界面程序开发环境提供的一个终端控件，有两个特点：

(1)能打开一个伪终端(Pseudo-Terminal)，执行 shell 命令，将使其复制为标准输入、标准输出和标准错误，使其类似一个真正的终端；

(2)它的 Pt_ARG_TTY_ INPUT 资源能够进行模拟键盘输入。

因此，我们将 PtTty 控件与 telnet 结合使用，实现了所要求的目标。

伪终端是一种类似于终端的特殊的进程间通信通道，通道的一端被称为主控终端(master pseudo-terminal device)，另一端被称为从属终端。它们分别由两个进程处理。这两个进程往往是父子进程。父进程打开伪终端的主控终端，然后调用 fork()派生子进程。子进程建立新的会话，并打开对应的从属终端，并将该从属终端复制为标准输入、标准输出和标准错误。然后子进程调用 exec()执行新的程序，该从属终端就形成了新会话的控制终端。对于子进程来说，从属终端就是它们的标准输入、标准输出和标准错误，同时也是一个终端设备。任何写入主控终端的数据将成为从属终端的输入；任何写入从属终端的数据将成为主控终端的输出。这样，主控终端上的进程(父进程)就能够为从属终端生成输入，而且还能够处理从属终端上的输出。整个过程如图 5 所示。

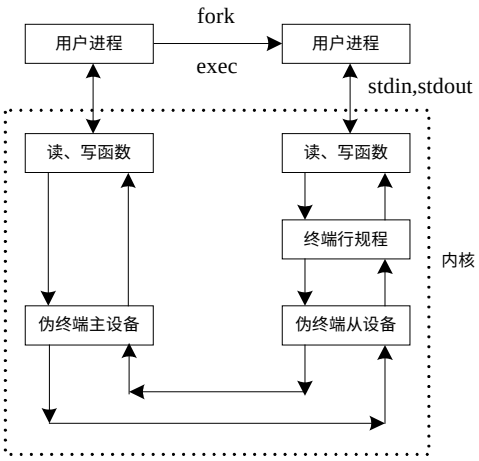

图 5 伪终端工作原理图

在 QNX 中，上述过程被巧妙地封装成一个 PtTty 控件，只须设置该控件的资源就可以完成上述过程。而不必关心底层的实现。PtTty 与 telnet 结合使用，其具体实现过程如下：

(1)设置 Pt_ARG_TTY_PSEUDO 资源，打开一个伪终端；

(2)设置 Pt_ARG_TTY_ARGV 资源，在打开的伪终端上执行 telnet 命令，连接远程主机，并且将该控件复制为标准输入、标准输出和标准错误。使其类似一个真正的终端，进行 I/O 操作。

(3)设置 Pt_ARG_TTY_INPUT 资源，模拟键盘输入远程主机的用户名、密码以及需要启动的程序的路径及可执行文件名，点击启动按钮，即可启动远程进程。

因此，二者的结合使用，使用户从烦琐的终端输入中解脱出来，真正实现了“一键通”的功能。

3.3 远程终止其它节点上的进程

一般情况下，“CTRL+C”组合键可以终止当前运行的进程。但是，该组合键不存在用字符串表示的简单方法，因此，不可能再用 Pt_ARG_TTY_INPUT 资源来简单地模拟键盘输入“CTRL+C”字符串。

本文对事件在 Photon 事件空间运行的机理(见图 6)进行了分析，模拟了一个键盘发出的事件发送给 PtTty 控件，产生组合键来终止远程进程。

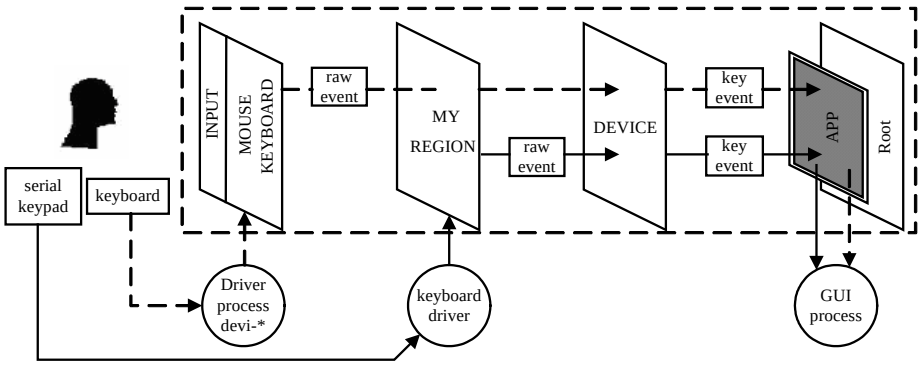

图 6 模拟键盘事件工作机理

图 6 中，虚线部分是键盘实际发出的事件。当通过键盘输入时，设备驱动模块(键盘驱动进程)会产生一个未被聚焦的事件。该事件在事件空间中穿梭，从 keyboard/mouse 区域向 Root 区域运动，当作用到 Device 区域时，被 Photon Manager 截获，Photon Manager 在 Photon 坐标空间为其分配一个位置，这样该事件即被聚焦。然后，Photon Manager 根据事件的类型重发该事件至应用程序区域(APP)，调用相应的函数，实现与用户的交互。

根据此原理，我们创建了一个区域(my region)，并将其放置于事件空间。然后，定义一个产生组合键的键盘事件(实线部分)，从该区域发出至 Root 区域，当该事件到达 Device 区域被 Photon Manager 截获，这时，Photon Manager 会认为该事件就是从键盘发出的。因此，它会用处理真实键盘事件的方法处理该事件，从而使该事件完成与真实的键盘事件同样的动作。

定义了一个键盘事件，其结构体如下：

```
struct {    PhEvent_t       event;
            PhRect_t        rect;
            PhKeyEvent_t    keyevent;
}MyEvent;
```

在此结构体内，将 event 成员变量定义为一个虚拟的键盘事件(Ph_FAKE_EVENT 和 Ph_EV_KEY)，将 keyevent 成员

变量定义为一个组合键“ CTRL+C ”。这样，MyEvent 就被定义成一个可以发生组合键“ CTRL+C ”的虚拟的键盘事件，通过 PtSendEventToWidget(PtWidget_t *widget, PhEvent_t *event); 将该事件发送至指定的控件。其中，*widget 为指定控件的指针，*event 为对所指定控件发送的事件。

这样， MyEvent 事件就会担当起真实键盘事件的功能，向 PtTty 控件输入“ CTRL+C ”组合键，终止当前运行的进程。

3.4 实时绘制电压、电流及晶闸管角度波形

原电源控制系统的电压电流及晶闸管角度都是以“ 文本 ”形式输出到屏幕，不能直观地反映电源的工作状态。操作监控平台针对此不足，定义了 3 个回调函数，实现了动态绘制电压电流及晶闸管角度波形，使用户随时都可以直观地观察放电过程中电源的工作状态。3 个回调函数如下：

(1)void Recv_Data();负责接收放电过程中 12 套极向场电源的电压电流及及晶闸管角度的值；

(2)绘图函数 void My_Draw_Func(PtWidget_t *widget, PhTile_t *damage); 负责绘制接收到的数据；

(3)定时器函数 int TimerCB(PtWidget_t *widget, ApInfo_t *apinfo, PtCallbackInfo_t *cbinfo); 负责定时发送 Ph_EV_EXPOSE 事件，调用绘图函数动态绘制波形。

绘图函数 My_Draw_Func()通过设置 PtRaw 控件的 Pt_ARG_RAW_DRAW_F 资源进行定义。

PtRaw 控件是 Photon 最基本的绘图控件，它通过调用 Pg*()系列函数 (Pg*()系列函数是 Photon 库中最底层的绘图函数) 在其画布上进行绘图。PtRaw 最具特色之处在于当其画布中的内容被毁坏时，它能够通知应用程序调用绘图函数修复被毁坏的图形。该函数的定义如下：

```
void My_Draw_Func(PtWidget_t  *widget, PhTile_t   *damage)
```

其中*widget 是指向 PtRaw 控件的指针，*damage 是指向一系列碎片的指针，这些碎片用于指示 PtRaw 控件的画布被毁坏的部分，保存于 PhTile_t 结构体中。当画布被毁坏时，PtRaw 控件就会调用 My_Draw_Func(PtWidget_t *widget, PhTile_t *damage)，通过*damage 对被毁坏的部分进行修复。至于画布的哪些地方被毁坏、重新修复什么地方，这些工作由 PtRaw 控件自动完成，编程人员无需考虑。绘图函数不能直接调用，只有当 PtRaw 控件的画布被毁坏时，该函数才被调用。PtRaw 控件的画布被毁坏的情况有 3 种：

(1)一个 Photon expose 事件到达画布所在的区域；

(2)PtRaw 控件实现时；

(3)覆盖在画布上的区域移走或被毁坏。

当上述任一情况发生时，就认为画布被损坏，PtRaw 控件就会通知应用程序调用绘图函数重画被毁坏的部分。但是，PtRaw 控件不能保存画布的原始数据。因此，Recv_Data()必须能够将每一炮放电过程中从控制系统接收的数据全部保存起来，以便在画布受到毁坏时，能够修复毁坏的图形。

操作监控平台需要实时绘制 12 套电源的电压电流波形及晶闸管角度的波形。每套电源需要绘制电压电流的预设值及实际值、两个晶闸管角度共 6 路波形。因此，操作监控节点需要实时绘制 72 路波形。

为了保证所有的波形不受毁坏，需要将 72 路波形的所有数据保存起来。因此，在程序中需要分别定义 72 个浮点型数组。但是，由于每炮放电时间不等，因此很难定义数组的大小，在此采用堆函数，根据每炮的等离子体放电时间的参数，动态地为每一路波形分配内存，将接收到的数据分别依次存入各自的数组中。当下一炮开始时，释放所有已分配的内存。这样，做到了使用多少分配多少的目的，合理地分配和使用了系统资源。

如何动态绘制波形呢？前面已经介绍，绘图函数不能直接调用，只有当画布被损坏时，该函数才被调用，才能够将 Recv_Data()接收到的数据绘制出来。因此，要调用绘图函数，至少要满足画布被毁坏的任一种情况。对此问题，我们同样根据事件在 Photon 事件空间的运行机理(见图 7)，采用画布被毁坏的第 1 种方法，实现了对绘图函数的调用。

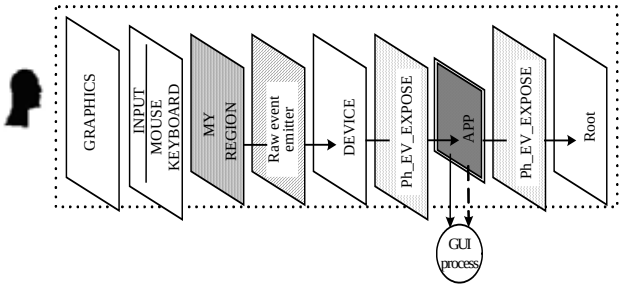

图 7 Ph_EV_EXPOSE 事件在事件空间运行图

我们在 Photon 的事件空间中创建了一个区域(my region)，让该区域发生一个矩形的 Ph_EV_EXPOSE 事件，并穿梭于事件空间。同样，当它到达 Device 区域时，Photon Manager 会将其截获，聚焦该事件，然后将其发往应用进程所在的区域(APP)。可以发现，当该矩形的 Ph_EV_EXPOSE 事件在穿过 APP 区域的前后过程中，对应用进程而言，相当于覆盖在其画布上的区域移走了，这正是画布被毁坏的第 3 种情况。因此，当矩形的 Ph_EV_EXPOSE 事件在穿过 APP 时，绘图函数即会被调用。

因此，实现动态地绘制波形，只需通过 TimerCB()定时向应用程序中的 PtRaw 发送矩形的 Ph_EV_EXPOSE 事件即可。其具体的实现方法与前面发送模拟键盘事件的方法大致相同，不再赘述。

4 结束语

操作监控平台的开发，给用户提供了一个友好、简便的操作方式。较好地克服了原电源控制系统操作方式的不足，极大地提高了控制系统的可操作性，并且使用户能够随时直观地观察电源控制系统的运行状态。当电源系统出现异常情况时，操作平台能够及时作出响应。经过在 HT-7 装置上的试运行，该平台基本能够满足实际需求，为 EAST 极向场电源系统的稳定工作提供了有力保障。

参考文献

1 北京领先实时科技有限责任公司. QNX 图形应用程序开发指南(上)[Z]. 2001.

2 北京领先实时科技有限责任公司. QNX 实时操作系统使用手册[Z]. 2001.

3 北京领先实时科技有限责任公司. QNX Neutrino 实时编程入门[Z]. 2001.

4 Stevens W R. UNIX 环境高级编程[M]. 北京：机械工业出版社, 2000.