

EPICS 控制系统的 Callback 机制

米清茹 刘松强

(中国科学院上海应用物理研究所 上海 201800)

摘要 EPICS 分布式控制系统提供一种 Callback 机制, 实现网络协议和设备驱动中的异步处理, 在设备数据或状态变化时设备驱动才读取数据, 并传输到网络, 减轻网络通讯负荷和控制器 CPU 负载。本文分析了 EPICS 的 Callback 机制, 并建立一个用 Callback 处理中断方式 I/O 的 ADC 数据采集的软件开发系统。

关键词 EPICS, 异步处理, Callback

中图分类号 TP335⁺.1, TP319, TP311.1

EPICS(Experimental Physics and Industrial Control System)^[1]是用于大科学装置控制系统软件开发和运行的工具集。它采用分布式结构, 兼容多种硬件设备, 应用于许多大型实验装置如加速器、射电天文望远镜等的控制系统中。EPICS 的分布式结构如图 1 所示, 在输入输出控制器(Input/Output Controller, IOC)上运行分布式动态数据库, 获取 I/O 设备数据, 控制 I/O 设备运行。在操作员接口(Operator Interface, OPI)计算机上运行图形界面, 显示并存储 I/O 设备的运行情况和运行结果。

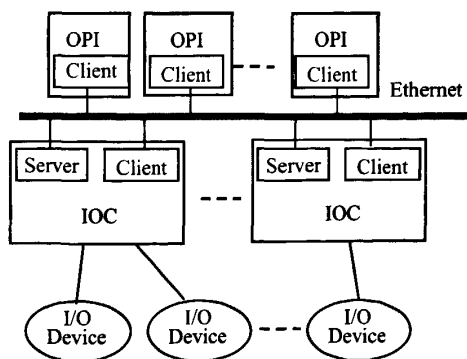


图 1 EPICS 系统结构
Fig.1 EPICS system architecture.

OPI 和 IOC 间用以太网连接, 用基于 TCP/IP 协议的通道访问(Channel Access, CA)协议。每个通道对应 IOC 数据库中的一个记录域。CA 协议采用客户-服务器模型, OPI 作为客户, 要求数据; IOC 主要作服务器, 提供数据, 有时也可作客户要求其他 IOC 上的数据。OPI 为及时掌握设备运行情况, 需不断获取 IOC 中的最新数据。当 OPI 频繁读取 IOC 数据时, 会占用极大的网络带宽, 甚至造成网

络拥堵; 而读取 IOC 数据频率太低, 则无法及时掌握设备运行情况。这种情况下就用状态变化(Change of State, COS)的获取数据方式, 即当某 I/O 设备的数据或状态变化超过指定范围时, IOC 才主动向 OPI 报送数据或状态, 以既使 OPI 实时监控设备运行情况和运行结果, 又极大地减轻网络的负荷。

IOC 通过设备驱动程序实现对 I/O 设备的控制和数据获取。但 I/O 设备操作有时要一段时间。若 IOC 不停地查询 I/O 设备的操作状况, 可能会使 IOC 在该时间段内仅能完成这一个任务, 影响 IOC 运行和对其它 I/O 设备的处理。这就希望 I/O 设备操作完成后, 通过中断方式通知 IOC 处理。这是一种异步处理的方式^[2]。EPICS 中网络 COS 方式和 I/O 设备的异步处理方式都采用 Callback 机制处理。

本文分析了 EPICS 的 Callback 机制的完整框架, 包括 CA 协议中实现 COS 方式的方法和在 I/O 设备与 IOC 动态数据库间用 Callback 任务实现异步处理的方法。并建立一个在 IOC 层使用 Callback 机制和中断进行数据采集的软件开发系统。

1 EPICS 中的 CALLBACK 机制

1.1 CA 协议中的 CALLBACK

CA 协议中的 CALLBACK 机制要求 IOC 中的记录值、报警状态等变化时由 IOC 上报给 OPI。如图 2 所示, IOC 中数据库监测模块(Monitor)专门监视状态变化事件。当扫描器(Scanner)扫描各记录时, 只要 Monitor 发现某个 I/O 通道的数据值超出设定的变化范围, 就会调用 db_post_events()函数, 把这个变化的值作为事件, 通过 CA 上传到已建立 Callback 监测机制的客户。

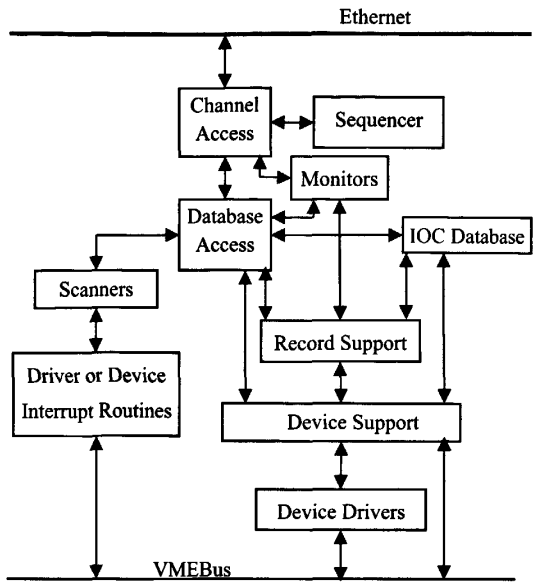


图 2 IOC 中的软件构成
Fig.2 IOC's software components.

在 CA 客户端应用程序接口中(Application Programmer Interface, API), 有一组函数专门处理 Callback 机制。用户通过调用 `ca_create_subscription()` 函数对指定的通道建立 Callback 功能。输入数据类型、通道 ID 等参数, 并设定用户定义的 Callback 函数的指针和参数指针。当 CA 客户端接收到 IOC 变化通知时就调用此自定义函数, 执行用户预设的动作。而 `ca_get_callback()`和 `ca_put_callback()`函数也用相同方式建立 get 和 put 动作的异步处理。

1.2 IOC 中的 Callback

IOCCore 是 EPICS 的核心程序, 是运行于 IOC 上的组件。它提供一个数据结构和几个 Callback 函数配合中断来完成 IOC 设备驱动中的异步处理。

CALLBACK 用来定义一个任务的私有数据。用户可用 `callbackSetUser()`和 `callbackSetCallback()`函数设定其所属数据结构和用户自定义的 Callback 函数。中断发生时, 系统会在中断服务程序(Interrupt Service Routine, ISR)中调用 `callbackRequest()`函数, 以 CALLBACK 数据为参数, 调用对应的用户自定义 Callback 函数, 执行用户预设的任务。下文介绍 Callback 如何把一个 I/O 设备以中断方式产生的数据传送到该 I/O 通道对应的动态数据库记录中。

2 软件测试平台

2.1 I/O 测试平台的硬件结构

为实现和分析 EPICS 的异步中断处理和 IOC callback 功能, 建立一个基于 VMEbus 的数据采集系统, 如图 3 所示。

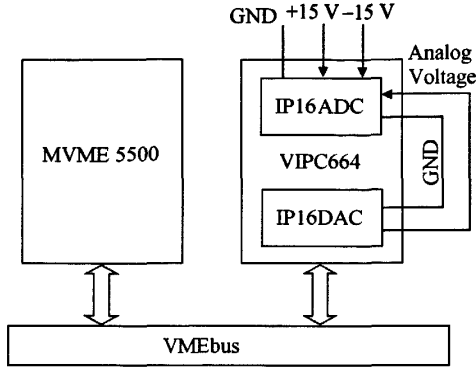


图 3 数据采集测试平台
Fig.3 DAQ test system.

系统中 IOC 的 CPU 主板用 MVME5500, 运行 vxWorks 实时操作系统。I/O 部分用夹层总线 IP(IndustryPack)结构, 由 VME 总线的 IP 载板装载 DAC 和 ADC 的 IP 模块 IP16DAC 和 IP16ADC。将 DAC 模拟输出作为 ADC 的模拟输入。IP 载板用 VIPC664, 载板上所有中断级别设置为 1。IP16DAC 和 IP16ADC 分别是 16 位数模转换器和模数转换器, 其中 IP16ADC 需外接 $DC\pm 15V$ 电源^[3]。

2.2 I/O 驱动的软件结构

由图 2 得知, IOC 设备驱动分三层: 记录支持程序(Record Support), 对某一记录类型特定处理, 如初始化、数据处理、报警等; 设备支持程序(Device Support), 从底层驱动读入数值, 放入记录相应的域中; 驱动支持程序(Device Driver), 底层驱动, 对硬件设备操作, 如设备初始化、寄存器设置及读取 I/O 通道等。

设备支持程序编写完后, 提供一个数据结构固定的函数调用入口表(Device Support Entry Table, DSET), 把编写好的初始化函数、读写函数、线性转换函数及报告函数等的函数入口地址写入 DSET, 记录支持程序不须修改可直接调用设备支持程序的函数。

IOC 的动态数据库以面向对象的思想设计, 每个记录类型就是一个类, 定义了同类数据包含的数据结构和操作方法。大部分现实能用到数据的记录

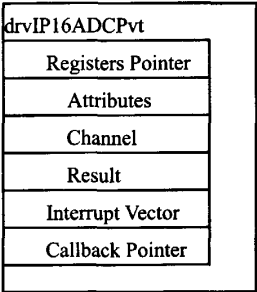
类型 EPICS 都已定义好了，包括对应的数据结构和操作方法。记录支持程序就是记录类型的操作方法。用户定义具体的数据时可采用这些记录类型，本文采用 ai 和 ao 两个记录类型，DAC 输出定义为 ao，以同步方式处理，将数据写入 DAC 模块后就退出；ADC 输入定义为 ai，以异步方式处理，触发 ADC 转换后退出，转换完毕后中断通知系统，系统获取 ADC 数值。

2.3 IP16ADC 异步中断数据采集

2.3.1 初始化

设备支持程序和驱动支持程序的初始化内容包括建立中断向量和 ISR 连接；为 Callback 分配数据区；指定 Callback 所属用户 ai 及应 Callback 请求执行的 Callback 程序 isrCallback()。

用异步中断和 IOC Callback 机制的 ADC 数据采集的各个软件模块都是在 vxWorks 操作系统下独立运行，IP16ADC 的地址、通道、控制命令、中断向量及 CALLBACK 结构数据都需要在它们与操作系统间传递信息。因此建立如下数据结构：



驱动支持程序初始化时，该数据结构被分配空间、赋值并指向到 pvt 指针；在连接中断向量和 ISR 时，pvt 指针作为参数传递给 ISR；这样，当中断后，ISR 就可通过 pvt 指针读取 A/D 转换结果，并产生 Callback 请求。设备支持程序的初始化函数中将 pvt 指针赋给 ai 记录的 dpvt 域，read_ai 等函数执行时就从 dpvt 域获取该 pvt 指针。drvIP16ADCPvt 数据区就是设备支持程序、中断服务程序 ISR 及 callback 线程间的纽带。

2.3.2 启动 I/O 数据采集

IOC 运行中，扫描器按照设定处理 ADC 记录，会调用 iocCore 提供的 dbProcess()函数。ai 记录包含一个域 PACT，原始设置为 False(=0)，标志这个记录不在被处理状态，允许 dbProcess()调用 ai 记录支持程序的 process()函数来处理记录。而 process()按照 DSET 调用设备支持程序的读取函数 read_ai()，

发现 PACT=False 时，触发 A/D 转换，同时把 PACT 域设为 True(=1)，表明该记录正被处理，然后直接退出。这样，扫描器再次调用 dbProcess()时，就不再调用 process()。read_ai()源码如下：

```
static long read_ai( struct aiRecord *pai) {
    drvIP16ADCPvt *pvt = (drvIP16ADCPvt *)pai->dpvt;
    if(pai->pact) {
        pai->rval = pvt->adc_data;
        pai->udf = FALSE; return 0;
    }
    pai->pact=TRUE;
    TriggerIP16ADC(pvt);
    return 0;
}
```

2.3.3 中断服务程序处理

当 A/D 转换完毕产生中断后，中断服务程序 ISR 就将数据从 ADC 寄存器读入 drvIP16ADCPvt 数据区，并调用 callbackRequest()函数发出运行 isrCallback()的请求。

2.3.4 自定义 Callback 函数

isrCallback() 函数是为本应用实例设计的 Callback 程序，它直接调用 ai 记录的 process()，而 process() 再次调用设备支持程序的读取函数 read_ai()，发现 PACT=True 时，将 ADC 数据读到 ai 记录的 RVAL 域中。当 process()完全处理完 ai 记录后，将 pact 域重新设置为 False。isrCallback()源码如下：

```
static void isrCallback(CALLBACK *pcallback) {
    struct aiRecord *pai;
    callbackGetUser(pai,pcallback);
    pai->rset->process(pai);
}
```

这样，通过 Callback 线程，中断 I/O 方式读得的数据就被放入该 I/O 通道的记录中。

3 结束语

本文分析和研究了 EPICS 分布式控制系统的 Callback 机制，并建立了一个数据采集系统。经测量，用 IOC 的 Callback 机制进行异步 I/O 数据采集，从启动数据采集到读取转换结果，平均时间仅为 40 μs。目前建立的这个软件开发平台只用了 1 个 I/O 通道，目的是开发和验证在 Callback 机制下的数据采集应用软件。在大型(~10,000 通道)数据采集系统中用 Callback 机制将更显其优点，今后如有机会建立 Callback 机制的大型应用系统，将对其性能做进一步测试。

软件设计方面, 目前用设置记录域 PACT 方法来防止其他任务(主要是扫描器)对该记录的重叠访问, 是否有更好的方法, 例如用 dbScanLock/dbScanUnlock 及不同优先级的 Callback 线程能最好兼顾系统的速度和效率, 这些都有待进一步研究。

参考文献

1 Kraimer M R. EPICS Input/Output Controller(IOC)

Application Developer's Guide, Release 3.14.8, Dec-2005, <http://www.aps.anl.gov/epics/base/R3-14/8.php>

2 刘松强. 计算机控制系统的原理和方法. 北京: 科学出版社, 2007

LIU Songqiang. Fundamentals and methods of computer control system. Beijing: Science Press, 2007

3 SBS: IP16ADC User's Manual, <http://www.sbs.com>

The callback mechanism in EPICS at SSRF

MI Qingru LIU Songqiang

(Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China)

Abstract The callback task in EPICS realizes asynchronous process in network transferring and device driver. In the asynchronous processing, only the data changed by device driver are collected, and transferred on the network. It alleviates the burden of both networks and the CPU load of IOC. In this paper, we analyze the callback task in EPICS, and introduce how to set up an ADC system with interrupt and callback.

Key words EPICS, Asynchronous processing, Callback

CLC TP335+.1, TP319, TP311.1