计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(5):239−244 [doi: 10.15888/j.cnki.csa.007373]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 C#的 OPC 客户端设计①

龚　勋, 王淑营

(西南交通大学 信息科学与技术学院, 成都 611756)
通讯作者: 龚　勋, E-mail: 1727474808@qq.com

摘　要: 现代工业生产过程的数据采集要求高采样率和传输的高实时性, 而现有 OPC 数据采集客户端不能满足要求. 为了解决这个问题, 通过分析 OPC 标准、服务器数据访问接口和组建对象模型, 并结合当前工业控制中数据采集的特点, 设计实现了基于订阅式数据采集方式的 OPC 客户端. 该方案在解决当前工业数据采集中遇到的问题具有较高的应用价值. 并在实际的生产环境中验证了其与标准 OPC 服务器在数据传输的稳定性和实时性, 为生产过程控制提供了可靠的数据基础.

关键词: 数据采集; OPC; 组建对象模型; 订阅式; 过程控制

引用格式: 龚勋,王淑营.基于 C#的 OPC 客户端设计.计算机系统应用,2020,29(5):239–244. http://www.c-s-a.org.cn/1003-3254/7373.html

Design of OPC Client Based on C#

GONG Xun, WANG Shu-Ying

(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China)

Abstract: Data collection in modern industrial production processes requires high sampling rates and high real-time transmission, while existing OPC clients of data acquisition cannot meet the requirements. In order to solve this problem, by analyzing the OPC standard, accessing the server data interface and component object model, combined with the characteristics of data acquisition in current industrial control, the OPC client based on subscription data collection is designed and implemented. This solution has high application value in solving the problems encountered in current industrial data collection. And in the actual production environment, it verifies the stability and real-time performance of the data transmission with the standard OPC server, which provides a reliable data foundation for production process control.

Key words: data collection; OPC; building object model; subscription; process control

传统工业数据采集系统, 由于生产设备种类较多、规格不一, 缺乏统一的工业标准, 不同设备厂商提供的硬件设备在设备驱动程序的开发、设备升级更新之后的维护工作、以及各应用程序之间通信方面存在困难[1,2]. OPC (Object Linking and Embedding for Process Control, 过程控制的对象链接与嵌入技术) 技术的诞生解决了这些问题, 极大提高了数据采集系统的开放性和互通性. OPC 技术是基于微软的 COM/DCOM 技术 (Component Object Model, 组件对象模型) 发展而来, 所有符合 OPC 标准的数据采集客户端程序都可以读取 OPC 服务器采集的生产过程数据, 大大提高了控制系统、现场生产设备和企业上层应用软件之间的互操作性, 为数据采集系统的发展奠定了基础[3].

① 基金项目: 山东省重大科技创新工程 (2017CXGC0608-02)
Foundation item: Major Science and Technology Innovation Engineering of Shandong Province (2017CXGC0608-02)
收稿时间: 2019-09-24; 修改时间: 2019-10-22; 采用时间: 2019-10-30; csa 在线出版时间: 2020-05-07

近年来, 较多学者在 OPC 数据采集客户端的设计与实现上进行了相关研究, 但是这些实现的客户端难以适应复杂多变的采集环境. 文献[4]介绍了 OPC 客户端与服务器之间的接口、数据读取方式等, 对实现 OPC 客户端有一定参考价值, 但并未进行具体实现. 文献[5]基于 VB 语言、自动化接口实现 OPC 客户端, 通过使用 OPCDAAuto.dll 自动化包装库, 把定制接口转换成自动化接口. 但由于其在大数据量数据采集时, 传输实时性较低, 以及多客户端连接时, OPC 服务器压力较大等方面的限制. 文献[6]讨论了在读取 OPC 服务器数据后, OPC 客户端和数据库通信问题, 实现利用数据库存储采集数据. 但是随着数据采集量越来越多, 这种方式不支持海量数据存储, 并且会失去数据库已有检索算法的优势, 精确查询的性能会降低.

相比于传统的 OPC 客户端开发周期长, 难度大, 适应性差. 本文通过分析 OPC 客户端和服务器之间的通信接口、数据读取方式、数据存储方式, 基于 C#设计实现了采用自动化通信接口, 订阅式数据读取方式的 OPC 客户端, 在简化开发工作量的同时支持采集数据直接存储到实时数据库, 方便精确查询使用, 还支持以文本文件形式存储, 以备后续进行问题追溯时查询, 在性能方面, OPC 服务器能同时支持多个 OPC 客户端连接访问, 对服务器的性能消耗较低. 所以, 本文设计的客户端在功能支持、发开工作量、数据传输实时性、对服务器性能消耗方面都有较大改进.

1 确定与 OPC 服务器通信接口

在 OPC 标准下, 客户端与服务器之间有两种通信接口: 定制接口和自动化接口[7], 其中定制接口主要使用 C/C++等过程性编程语言开发, 使用定制接口和服务器通信的方式比较复杂; 另一种是自动化接口, 主要使用 Visual Basic, C#等语言开发, 相比于定制接口, 由于自动化接口集成了组建对象模型特征, 可以使用自动化包装库, 所以开发相对简单. 并且, 使用 C#语言实现 OPC 客户端, 不仅可以实现数据采集功能, 还可以实现其他功能. 其结构如图 1 所示.

采用定制接口方式与服务器通信需要开发人员熟练掌握相应的 COM 和 DCOM 知识的综合应用, 并且清楚的了解客户端和服务器的整个通信过程, 并且开发周期比较长. 而采用自动化接口方式则更为方便, 因为可以使用自动化包装库实现自动化接口转化为定制接口. 通过这种方式访问服务器, 降低了对开发人员的要求, 缩短了开发周期. 所以本文采用了自动化接口对方式.

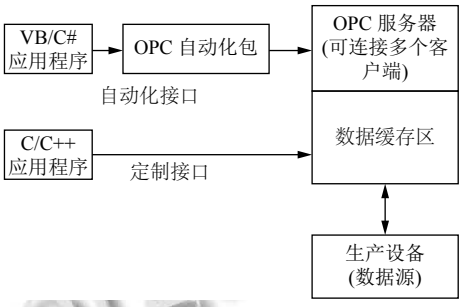

图 1 客户端与服务器通信接口

根据 OPC 标准, OPC 服务器包括 6 类对象, 分别是: 服务器对象 (OPC Server)、OPC 浏览器对象 (OPC Browser)、OPC 组集合对象 (OPC Groups)、OPC 组对象 (OPC Group)、OPC 项集合对象 (OPC Items)、OPC 项对象 (OPC Item). 开发人员需要熟悉了解这 6 个对象的属性、事件和方法[8–10].

OPC 服务器对象: OPC 标准中定义对 COM 对象, 是 OPC 服务器的实例, 也是创建其他 5 个对象的基础;

OPC 浏览器对象: 包含在服务器中存在的分支或项目名称的集合, 属于与可选部分, 如果连接的服务器不支持这个对象, 客户端则不会创建;

OPC 组集合对象: OPC Groups 与 OPC Group 之间存在一对多的关系, 是 OPC Group 对象的集合, 主要用于创建、删除和管理 OPC 组对象;

OPC 组对象: 主要用来管理客户端需要采集的数据项;

OPC 项集合对象: 与 OPC 项之间存在一对多的关系, 是 OPC 组的属性之一, 当新添加一个数据采集项时被赋默认的属性, 比如默认状态 (defaultIsActive)、默认数据更新周期 (defaultRequestedDataType);

OPC 项对象: 是客户端与服务器之间的数据访问连接. 每个项由采集 ID, 变量值和时间戳构成.

2 确定数据读取方式

OPC 数据读取规范定义了 OPC 客户端对现场生产设备产生数据对读取方式. 每个 OPC 客户端可以连接到多个不同的服务器, 只要服务器满足 OPC 规范; 多个 OPC 服务器也可以和一个客户端进行通信, 只要客户端符合 OPC 标准. 在 OPC 标准中, OPC 客户端访

问服务器的方式有 3 种: 分别是同步数据访问、异步数据访问和订阅式方式, 其中订阅式方式是特殊的异步方式[11].

2.1 同步访问方式

同步数据访问方式可用于客户端访问服务器数据, 也可以用于客户端向服务器写入数据, 当客户端读取 OPC 项对应的实时数据时, 客户端程序一直处于等待状态, 直到数据读取完毕才能继续下一阶段的工作, 当写入数据时也是如此. 所以采用这种数据访问方式会阻塞客户端线程, 适合短时间请求数据并且和服务器的数据交互量少的情况. 当出现大量数据交互或者客户访问, 容易造成网路拥塞, 性能下降[12,13].

2.2 异步访问方式

同样异步数据访问方式可用于客户端访问服务器数据, 也可以用于客户端向服务器写入数据, 当客户端向服务器发送读取或写入数据请求后, 不用等待消息的返回, 随后就可以进行其他事务的处理, 当服务器完成请求处理之后, 服务器转为客户端, 主动向原来的客户端发送异步数据访问完成事件, 并将数据访问结果返回给客户端, 客户端接收处理完成事件. 在这种方式下, 数据访问效率更高, 能避免多客户端大数据量请求造成的阻塞, 并且可以节约 CPU 和网络资源. 但是该方式需要客户端程序增加事务管理功能, 接收处理访问完成事件, 增加了编程难度, 也额外增加了开发人员工作量.

2.3 订阅式访问方式

订阅式数据访问方式是一种特殊的异步数据访问方式, 但不同于的是订阅式方式只能用于客户端读取服务器端的数据, 不能向服务器写入指令. 但是在实际工业数据采集系统中, 大部分情况是采集客户端数据. 关键在于, 进行数据交互时并不需要客户端程序实时向服务器发送请求, 而是服务器自动周期性的扫描数据缓冲区, 若缓冲区中的数据值发生变化或者变化超过一定幅度, 则更新数据缓冲区, 并向客户端发送数据变化事件, 客户端实时处理接收的数据变化事件, 所以采用订阅式方式, 需要服务器通知客户端数据更新, 客户端需要增加接收器对象, 接收器涉及到组建对象模型编程的相关知识, 具体可查阅相关文献, 这里不在赘述. 采用订阅式数据访问方式如图 2 所示.

其中数据更新的变化范围被称为敏感带 (DeadBand), 由于敏感带的存在, 可以无视模拟量的微小变化, 减少了客户端和服务器的负荷. 相比于同步和异步数据访问方式, 订阅式数据方式有效的降低了客户端访问服务器的次数, 减少了服务器的工作量, 并且避免了因传输数据量大而造成的网络阻塞情况, 所以这种方式在大数据量访问优势更为明显. 针对当前制造业生产过程数据采集特点: 采集数据量大、采样频率高并且传输实时性高, 所以本文设计的 OPC 客户端选用订阅式方式无疑是最佳的选择.

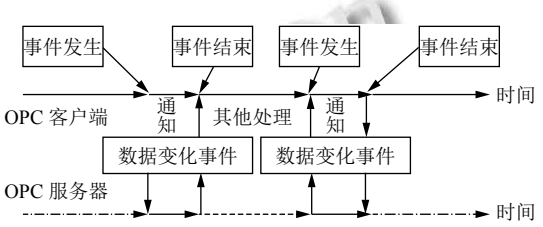

图 2 订阅式方式

3 OPC 客户端实现

3.1 总体设计

OPC 客户端总体设计如图 3 所示, 主要分为 3 个部分: 分别是建立连接、读取数据和断开连接; 其中建立连接部分需要完成 OPC 服务器与 OPC 客户端之间的身份确认, 连接信息的保存等功能, 具体包括枚举本地 OPC 服务器、初始化服务器对象、预览初始化接口、枚举地址空间等; 读取数据部分需要客户端按照定义的分组信息完成从服务器读取指定数据项, 并完成数据转储功能, 具体包括添加服务器对象、添加组对象、添加项对象、转储数据等; 断开连接部分需要客户端终止与服务器的连接, 释放资源, 同时服务器也需要确认客户端连接释放. 以上 3 个功能模块保证 OPC 客户端能顺利访问 OPC 服务器所有数据项的时间戳、采集点唯一 ID 和采集值[14,15].

其中关键部分包括创建 OPC 服务器对象、OPC 组对象和 OPC 项对象, 并管理这些对象的工作周期. 一个 OPC 服务器对象可以包含多个 OPC 组对象, 并且可以创建和删除组对象; 一个 OPC 组对象可以包含多个 OPC 项对象, OPC 组对象主要包含自身基本信息和其中所有的 OPC 项对象信息; 而 OPC 项对象是最基本的单位, 主要包含 OPC 客户端和服务器的连接信息和采集数据项信息等. 其中 OPC 服务器对象, OPC 组对象和 OPC 项对象以及每个对象需要用到的函数关系如图 4 所示.

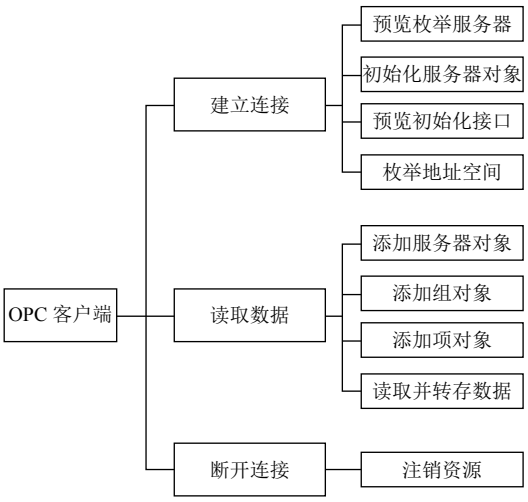

图 3 OPC 客户端总体设计

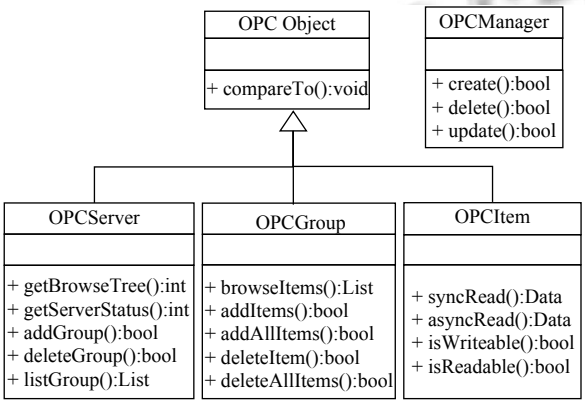

图 4 OPC 各对象之间关系

图 4 中提供的 OPC 对象是所有对象的父类, 只具有最基本的 CompareTo() 比较函数, 其他的子对象可以根据自身需要进行扩展; 同时, 提供了专门用于管理 OPC 对象的 OPCManager 管理单元, 包含 create()、add() 和 remove() 等 3 个函数分别用于新建、添加和删除 OPC 对象. OPC 服务器对象负责新增和删除一个或者多个组对象, 其中包含的 getOPCBrowseTree() 函数用于浏览地址空间并建立树型关系. OPC 组对象主要是负责新建、添加、编辑和删除 OPC 项对象, 项对象是与现场生产设备对应的最基本的数据对象; OPC 项对象调用 getItemProperty() 函数可以获取采集数据项的唯一标识 ID、采集值和时间戳.

3.2 读取服务器数据

OPC 客户端完成服务器数据读取的过程包括预览和初始化 OPC 服务器、预览地址空间以及添加和删除组对象和项对象、数据读取、数据转储.

预览 OPC 服务器名称主要是通过枚举本地所有的服务器信息并初始化需要连接的服务器对象来获取服务器名称和唯一标识符, 做好连接 OPC 服务器的准备工作; 在初始化服务器对象的过程中需要调用 OPC 规范中管理各对象的函数. 比如 OPCServerListClass() 函数得到服务器的唯一标识符, 即可获得初始化服务器, 调用 OPCServer.addGroup() 方法创建 OPC 组对象, 然后接着获取 GroupName、updateRate 和 ID 等参数信息, 其中 updateRate 表示采集数据等更新频率, ID 就是标识采集点的唯一标识符.

建立 OPC 服务器连接的过程就是初始化 COM 对象的过程, 需要根据之前获得的唯一服务器标识符来获取 COM 对象句柄地址并初始化 OPCServer 接口函数, 然后进行服务器的连接. 此阶段 OPC 服务器分为远程服务器和本地服务器, 如果连接的是本地服务器则较为简单, 若连接的是远程服务器需要设置访问类型为远程访问, 然后继续调用 COM 对象初始化函数进行服务器的远程连接.

预览地址空间包括预览通信接口并初始化和填充 TreeView 控件等功能, 为后续添加 OPC 项对象做准备, 若数据采集的量比较大, 需要手动输入采集项名称和路径则效率较低, 所以采用预览地址空间的方法避免手动添加, 直接调用 OPCBrowseAddressSpace() 函数作为预览地址空间的函数接口, 然后接着使用 ChangeBrowsePosition() 函数做为改变位置的函数接口, 使用 BrowseOPCItemIDS() 函数获取采集数据项相关参数, 方便用户添加或者删除需要的服务器对象、组对象和项对象. 其读取服务器数据的时序图如图 5.

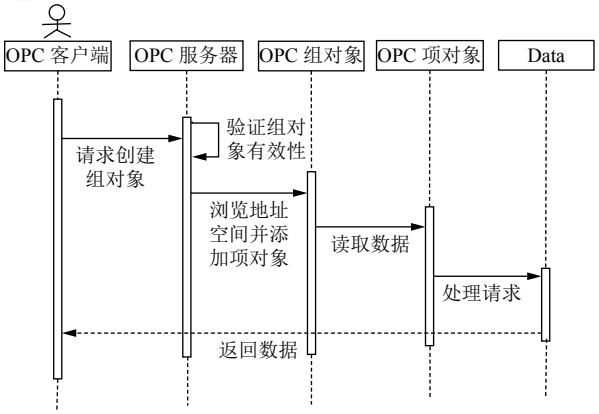

图 5 数据读取时序图

首先是创建一个 OPC 组对象并验证组对象的有效性, 如果无效则会抛出相应的异常并重新添加组对

象, 有效则继续执行, 浏览服务器的地址空间, 并调用函数 addItem() 添加项对象, 每个项对象就是需要读取的数据项基本单位, 调用项对象的 getQuality()、realItemValue() 和 dateTime() 函数开始读取数据.

3.3 数据转储

本文设计的数据转储模块可以将读取的数据转储到 MySQL 数据库, 也可以转储为文本文件, 连接 MySQL 数据库时需要使用到 MySQL.Data.dll, 这是专门用来管理连接和断开数据库连接的库. 转储数据到 MySQL 数据库时提供了数据库连接模式和断开模式, 在连接模型下数据库和客户端一直保持连接状态, 在断开模式下客户端和数据库只在有数据更新的情况下才进行连接操作. 结合现代制造业数据采集特点, 采用断开模式优势较为明显, 其中将数据转储到 MySQL 数据库时需要以下几步:

(1) 使用 Connection 连接对象连接客户端和数据库, Connection 对象包含 Open() 和 Close() 方法, 用于打开和关闭数据库连接, 只包含一个属性 ConnectionStrng 属性, 这个属性又包含了服务器名 DataSourse、数据库名 InitialCatalog、用户名 UserName 和密码 PassWord 等;

(2) 使用 Command 对象对已经建立连接的数据库发出请求操作数据命令;

(3) 将从服务器读取来的数据放到数据适配器 DataAdapter 对象中, 然后把数据转储到 DataSet 对象中;

(4) 然后在本地的 DataSet 对象中管理数据, 接着利用 DataAdapter 对象进行数据库写入操作更新数据, 最后关闭 Connection 连接.

在断开模式下, 不需要客户端和数据库服务器一直保持连接状态, 这样大大降低了服务器承受的压力, 减少了资源的浪费, 支持多个客户端同时发起连接请求的情况.

3.4 实现展示

OPC 客户端首页如图 6 所示, 在首页可以配置连接服务器的相关信息, 服务器 IP 地址和服务器名. 为了使用的方便, 该部分设计为只需一次配置的形式, 首次配置的信息保存为 XML 配置文件, 如果服务器不更改则不用重新配置; 同时提供定义分组、查看分组信息和分组管理功能的入口, 以及显示当前正在采集的信息, 包括采集组数量、采集项数量和开始采集时间等.

点击查看分组信息, 可以查看当前客户端已配置的所有分组列表, 每个分组下的所有采集数据项列表, 以及每个数据项当前的采集时间点, 采集值, 具体如图 7 所示.

图 6 客户端首页

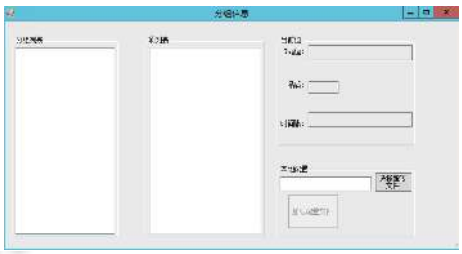

图 7 查看分组信息

点击定义分组, 可以新建分组, 定义分组名, 配置该分组需要采集的所有数据项列表, 以及该分组的数据项更新频率等, 具体如图 8 所示.

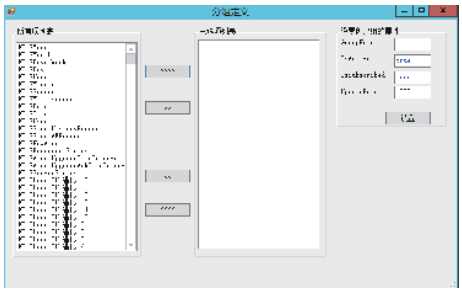

图 8 定义分组

4 结论与展望

本文通过分析现代制造业的发展趋势, 以及当前数据采集技术的弊端, 设计开发了适合当前工业控制系统需求的 OPC 数据采集客户端. 其中详细介绍了 OPC 客户端的通信接口、数据读取方式和具体的实现步骤, 并在实际生产环境投入使用, 真正解决了当前制造业在数据采集方面遇到的难点; 验证了客户端与服务器数据读取的稳定性和实时性; 对 OPC 技术在数据采集方面的应用进行了更深入的拓展. 最后, 希望本文可以为将来数据采集技术的深入研究起到抛砖引玉的作用.

参考文献

1 杜赟萌. 基于 OPC 技术的工业通讯应用. 科技风, 2019, (15): 81.

2 杨虎, 乔立慧. OPC 技术在工业网络控制系统中的应用. 电子科技与软件工程, 2019, (16): 137–138.

3 OPC Foundation. Data Access Custom Interface Specification Version 2.05. 2001: 11–15.

4 朱立军, 安娜, 陈未如. 基于 Visual C#的 OPC 客户端实现. 现代电子技术, 2009, 32(2): 171–173. [doi: 10.3969/j.issn.1004-373X.2009.02.051]

5 王绪彪, 张望, 江丹玲. 基于 VB 的 OPC 客户端软件的设计与实现. 自动化与仪器仪表, 2011, (2): 46–49.

6 张河, 鲁五一. OPC 客户端与实时数据库通信的实现. 计算机工程与科学, 2008, 30(5): 81–83. [doi: 10.3969/j.issn.1007-130X.2008.05.025]

7 Torrisi NM, Oloveira JFG. Remote control of CNC machines using the CyberOPC communication system over public networks. The International Journal of Advanced Manufacturing Technology, 2008, 39(5–6): 570–577. [doi: 10.1007/s00170-007-1244-0]

8 Burke TJ. The performance and throughput of OPC c-a rockwell software perspective. Rockwell Software Inc., 2004.

9 顾键, 王京春, 黄德先. OPC-COM 技术在工业自动化软件中的应用. 计算机工程应用, 2002, 38(12): 207–209.

10 袁小坊, 王东, 谢高岗. OPC 中订阅机制的实现分析. 计算机工程与应用, 2009, 45(1): 89–91. [doi: 10.3778/j.issn.1002-8331.2009.01.026]

11 Iwanitz F. OPC: Fundamentals, Implementation & Application. Heidelberg: Huthing, 2002. 113–115.

12 Jia ZP, Li X. OPC-based architecture of embedded web server. In: Wu Z, Chen C, Guo M, et al, eds. Embedded Software and Systems. Berlin, Heidelberg: Springer-Verlag, 2005. 362–367.

13 Son M, Yi M. A study on OPC specifications: Perspective and challenges. International Forum on Strategic Technology 2010. Ulsan, South Korea. 2010. 193–197.

14 刘会令. 基于. NET 的嵌入式系统 OPC 客户端开发[硕士学位论文]. 北京: 北京化工大学, 2013.

15 尤枫, 邵俊军, 赵恒永. 基于 WinCE 的 OPC 数据采集系统设计与实现. 计算机工程与科学, 2007, 29(6): 124–127. [doi: 10.3969/j.issn.1007-130X.2007.06.037]