

Pyxos FT 协议介绍

美国埃施朗(Echelon)公司北京代表处

刘永生

关键词: Pyxos FT 协议, 分时复用 (TDM) 协议, 登记注册, 无处理器从设备

摘要: Pyxos FT 协议是基于时分复用(TDM)的机制。一个 Pyxos FT 系统由一个 Pilot 和最多 32 个的 Pyxos Points 组成。每一个 Point 分配有一个时间片, 用于确定何时允许该 Point 使用通信介质。Pilot 一次向每一个 Point 发送一个数据包, 然后从每一个 Point 接收一个数据包, 然后重复上述过程。这个机制本身避免了竞争和冲撞, 数据在给定的时间内可以确定性地交付到目的地。

Keywords: Pyxos FT Protocol, Time-Division Multiplexing (TDM) protocol, Registration, Unhosted Point

Abstraction: The Pyxos FT protocol uses time-division multiplexing (TDM) to manage the communications between the Pilot and the Points. TDM is a type of digital multiplexing in which two or more simultaneous bit streams are encoded as subchannels into a single bit stream by interleaving bits from the different bit streams. The combined bit stream is decoded at the receiving end. For a Pyxos FT network, the different sub-channels are the bit streams from the different Points on the Pyxos FT network, each communicating with the Pilot. Each of the sub-channels has a fixed bit rate, providing deterministic response for each of the sub-channels. The Pyxos FT Chips on the Pilot and the Points manage the interleaving of the channels.

一、概述

Pyxos 网络技术的核心是 Pyxos FT 芯片, 在芯片上内嵌有 Pyxos 通信协议。Pyxos 网络由主设备 (Pilot) 和从设备 (Point) 构成。在两种设备中均包含 Pyxos FT 芯片。

Pyxos FT 协议能够确保数据的可靠递交。Pyxos FT 芯片自动确认事务, 新的数据只有在前面的数据可靠递交后才会被处理。每一个数据包均带有自己的 18 位的 CRC 校验, 因此只有有效的数据才被递交和确认。

二、 Pyxos 芯片上的内存和寄存器

Pilot 和 Point 使用主处理器时，对于主处理器来说，Pyxos FT 芯片相当于一个通过 SPI 端口访问的串行内存。主处理器应用通过读/写 Pyxos FT 芯片上的存储器来操作该芯片，并完成 Pyxos 网络上的数据通信。

Pyxos FT 芯片的片上存储器分为两个区域：控制寄存器区和数据存储区。在 Pilot 应用中，数据存储区用于数据帧存储区，Pilot 使用数据帧存储区来缓存和 Point 之间的通信。在 Point 应用中，数据存储区用于保存 Pyxos FT 芯片的值（PCV）存储区。PCV 存储区存储有一组的 PCV，它们通过 Pyxos FT 芯片索引（PCI）来寻址。

Pilot 和 Points 通过读写 PCI 来实现它们之间的数据交换。当 Point 向 PCI 写入数据时，PCI 和 PCV 被传送到 Pilot。Pilot 可以在一个时间片寻址一个 Pyxos Point，并通过网络向该 Point 的 PCI 写入 PCV。

当通过 SPI 访问 Pyxos FT 芯片时，其存储器空间是以字节的方式来寻址的，当通过网络来访问 Pyxos FT 芯片时，存储器空间是以 PCI 来寻址。使用 PCI 访问 PCV，PCV 是四字节的数据。

三、 Pyxos 的数据帧

Pilot 和一个 Point 往返交换一个数据包的通信周期称之为一个数据帧。每一个数据帧包含有一个起始数据包（SOF），随后依次是写时间片和读时间片。写时间片分成一系列的单个数据包，每一个时间片对应一个数据包。类似地，读时间片也分成一系列的单个数据包，每一个时间片对应一个数据包。

一个数据帧最多包含 32 个时间片，标示为 0—(n-1)，n 是数据帧中的时间片的个数。时间片标识符被分配给每一个 Point，用来作为写时间片和读时间片的索引。Point 使用这个索引在相应的写时间片接收数据，在读时间片发送数据。

Pyxos FT 协议支持三种类型的数据包：SOF 数据包、写数据包和读数据包。SOF 和写数据包是由 Pilot 在一个自动传输事务中发送。读数据包由每一个 Point 发送，一个时间内只有一个发送。SOF 和读数据包均包含一个前导用于物理层实现同步。所有的数据包均包含一个 18-bit CRC 校验。SOF 实现两个主要的目的，表示一个数据帧开始和给出数据帧的长度。Points 使用这些信息来判定何处发现写给自己的数据包，何时发送自己的数据包。SOF 数据包包含一个模式标记是一个 SOF 数据包，用于区别于其它的数据包。SOF 的域值编码有当前数据帧的时间片个数。

Pyxos FT 网络的运行速率为 312.5 kHz – 或者 3.2μs 表示一个数据位。

数据帧的持续时间取决于与数据帧中的时间片的数目。时间片的数目必须是偶数，取值范围是 2-32。

四、 Pyxos 的事务

在 Pyxos FT 协议中，Pilot 和一个 Point 之间一次成功的、确认的数据传输定义为一个事务。在一个数据帧中，最多可支持 64 个执行事务。有两种基本的事务类型—从 Pilot 到 Point 的写事务，从 Point 到 Pilot 的读事务。

◆ 写事务 (Pilot 到 Point 为写)

一个写事务是从 Pilot 应用请求数据发送开始，到 Point 接收到这个数据结束。写事务具体操作步骤包括：

- 初始化，Pilot 应用通过 SPI 端口向写时间片缓冲区写入数据
- 发送，Pilot Pyxos FT 芯片将写时间片缓冲区的数据发送给 Point
- 接收，Point Pyxos FT 芯片接收数据，发回一个确认，告知 Point 应用有新的数据
- 读，Point 应用读入新的数据
- 结束，Pilot Pyxos FT 芯片接收到确认并清除该次写事务

◆ 读事务 (从 Point 到 Pilot 为读事务)

读事务和写事务不同，读事务确认需要 Pilot 应用参与—直到应用发现数据并进行读取后，相应的 ACK 才会被发出。

这些不同，给读事务带来一些细小的差别：

- 排队： Point 应用通过 SPI 接口向一个 PCV 进行写操作
- 发送： Point Pyxos FT 芯片选择至多两个 PCVs 来发送，将它们拷贝到发送缓冲区中，向 Pilot 发送读时间片
- 接收： Pilot Pyxos FT 芯片接收数据并通知 Pilot 应用，但对接收并不发确认
- 读： Pilot 应用读数据并进行处理数据
- 确认： Pilot Pyxos FT 芯片确认读事务
- 终止： Point Pyxos FT 芯片接收到确认并清除该事务

五、 Point 配置和登记注册

一个 Point 的配置信息决定了设备是否有处理器，如何进行登记注册，时间片标示符等。Pyxos 芯片控制寄存器作为时间片编号来进行发送和接收数据包的操作。Point 的应用程序，在 Point 没有配置完成的时候，不能发送数据（即向一个 PCI 进行写操作）。

Point 设备向 Pilot 注册可以使用三种方式：自动注册、手工注册和硬连线注册。

◆ 自动登记注册过程

自动登记注册由三个步骤组成：

- 1、**空闲的时间片通告**：**Pilot** 应用发布能够使用的时间片。
- 2、**注册申请**：一个未配置的 **Point Pyxos FT** 芯片识别一个空闲的时间片通告，然后申请一个时间片。
- 3、**时间片分配**：**Pilot** 应用识别注册申请并告知 **Point** 所分配的时间片。

Pilot 应用必须明了哪一个时间片当前已经分配了，哪一个时间片是空闲的。**Pilot** 应用然后应该决定何时广播空闲的时间片。这可以一直进行，也在一些特定的时间进行，这可以根据应用的需要而定。

如果一个 **Point** 的 **Pyxos FT** 芯片处在未配置状态，且被设置为自动注册模式，**Pyxos FT** 芯片会监测时间片，查询空闲的时间片广告。如果发现一个或多个空闲的时间片，**Point** 的 **Pyxos** 芯片会随机选择一个空闲的时间片，然后尝试在这个选中的时间片中发送一个注册请求。一个注册请求包括该 **Point** 的 **Pyxos FT** 芯片的 **UID1** 和 **UID2**。一旦 **Point** 的 **Pyxos FT** 芯片发送了一个注册请求，就会等待分配时间片。被分配的时间片可以是发送注册请求的时间片，也可以使其他的时间片。

如果在一个空闲的时间片中有多个 **Point Pyxos FT** 芯片尝试发送注册请求，那么 **Pilot** 就不能收到有效的注册请求。这时，注册请求不会被 **Pilot** 确认，**Point** 的 **Pyxos FT** 芯片会停止发送他们的注册请求。然后重新随机选择一个时间片，重新开始自动注册过程。

一旦 **Pilot** 接收到注册申请，就会选择一个时间片分配给发出申请的 **Point**。新分配的时间片可能与 **Point** 发出申请的时间片不一致。在实际的分配之前，**Pilot** 必须取消该时间片的空闲广告，保证没有其他的 **Point** 试图使用该时间片。进行时间片分配时，**Pilot** 在分配的时间片写入相应 **Point** 的 **UID**。**Point** 的 **Pyxos FT** 芯片识别自己的 **UID**，将分配给的时间片数值写到 **TSID** 位域中，进入配置状态（设置 **CFG** 为 1）。然后对时间片分配事务进行确认。

◆ 手工注册

当系统中有超过一个的设备具有相同的程序编码（即同一种设备）时，使用手工注册方式，但是 **Pilot** 应用程序必须将这些设备进行分别。例如，系统中有两个同类的探头位于不同的物理位置，系统必须使用其他物理位置信息来和具体的位置相关联起来。

不使用处理器的 **Point**，具有相同的设备类型，因此，如果 **Pilot** 需要分别处理这样的设备时，就必须使用手工注册方式。

这种情况下，Point 设备没有足够的信息来将自己区别开来，因此就必须要求用户的使用一些交互操作来进行识别。

手工注册和自动注册有些类似。基本的注册程序如下：

- 1、**空闲的时间片通告：**Pilot 应用发布能够使用的时间片。
- 2、**注册申请：**当安装人员指挥 Point 开始寻找空闲的时间片时，未配置的 Point 的 Pyxos FT 芯片时被一个空闲的时间片通告，并请求一个时间片。
- 3、**时间片分配：**Pilot 应用识别注册申请并告知 Point 所分配的时间片。

◆ 硬连线注册方式

硬连线的注册模式不依赖与 Pilot 或任何的网路事务。但是，Pilot 应用必须为硬连线的 Point 保留时间片。

这种注册方式用于使用其他的注册方式给 Point 分配时间片，而不通过 Pilot 进行分配。例如，Point 配线束可以使用键入的方式，这样每一个位置都有一个预先分配的时间片。

一旦 Point 应用程序定下期望得到的时间片后，必须通知 Pyxos FT 芯片。

六、 复位处理和错误恢复

可以设计一个 Pilot 或 Point 应用来预测可能的错误，从而在发生不可恢复的错误时提供错误修复和错误情况提示。

下列错误是可以检测到的：

- 失去的网络通信。
- 高的网络通信错误速率。
- 主处理器不能和 Pyxos FT 芯片通信。

所有错误情形的检测和恢复都能由 Pilot 来处理。对于 Pilot 来说，监测所有的错误情形并进行恢复是可行的，因为 Point 可能不能收集到足够的信息来进行合理的错误原因判断。

高的网络错误率可能是由于过多的电器噪声或则线路问题（例如，间歇性的短路）所导致。这样的错误就必须由人工处理来修复。

Pilot 可以简单地记录这些类型的错误并给操作人员以提示。能够允许的错误率会根据应用的不同而不同，但是对于一个设计良好的网络来说，错误应该很少出现。

Pilot 应用必须监测网络上的所有的 Point 的工作情况，并检测与这些 Point 通信时发生的通信问题。对于 Pilot 来说，最简单和最有效的探测 Point 失败的方式是跟踪记录其自身与每一个 Point 成功建立通信后有多少数据帧已经流过——这包括所有的从 Point 接收到的数据和对于写事务收到的确认。

七、 管理无处理器的从设备

一个 Pyxos FT 芯片可以单独用来构成一个设备，而不要使用任何的主处理器，这时，Pyxos FT 芯片的 SPI 管脚转换为通用的 I/O，能提供 4 路数字的输入输出和 1 路数字输入，如表一所示。

Pilot 必须管理无处理器的 Point 上的 IO—包括配置、读数据、写数据。

表一 Pyxos 芯片无主处理器模式数字 I/O

名称	管脚编号	方向
MOSI/DIO0	4	可配置
MISO/DIO1	5	可配置
CS~/DIO2	1	可配置
INT~/DIO3	2	可配置
SCLK~/DI	3	输入

前四个 DIO 管脚的方向，是由 Pilot 通过网络进行配置。每一个管脚均可以独立进行配置。可以配置成输出（Pilot 设置输出值）也可以配置成输入。配置成输入时，Pilot 可以查询输入值，或当输入发生变化时自动发送。

DIO 数据寄存器用来对 DIO 管脚进行读或写操作，如果一个管脚配置为输出，向该寄存器的相应位写数值会使得该管脚变到相应的设置值。读该寄存器会返回管脚的当前状态。

DIO 方向寄存器用来配置 DIO 管脚是用于输入或输出。向该寄存器的为写入“0”，即设置相应的位为输入，写“1”即设置相应的位为输出。DIO 管脚初始设置为输入。

DIO 配置寄存器提供 DIO 管脚的配置控制功能。且只有一个值域：AUTOUPDATE (位 8)，如果 AUTOUPDATE 设置为允许（值“1”），DIO 管脚的值在每一个数据帧均被采样，如果任意输入有变化，更新的数据就会被发送到 Pilot。因为输入的采样是在每一个数据帧一次，变化的频率超过采用频率，数据就会被丢失。另外，输入更改频率不能快于 5MHz，在管脚输入改变之间必须最少有 200ns 的时间间隔。

八、 结束语

Pyxos 控制网络技术是基于 TDM 的总线控制技术，使用该技术可以实现确定性操作的应用，该技术通信速率 312.5 Kbps，是一个完全的主/从控制模式的网络，从设备可以仅使用 Pyxos 芯片构成具有数字输入输出的设备，传输支持无极性的双绞线，且在这一对双绞线上同时传输 24V 的交直流电源，是一种低成本的网络技术。为工业自动化、智能交通、智能家庭等需要使用控制联网的领域，提供了一种新的、具有高性价比的解决方案。

参考资料

- 1、《Pyxos FT Programmer's Guide》美国埃施朗公司，2007 年
- 2、Pyxos 技术介绍，仪器仪表标准化与计量 2007-2，P33-35