TechnoCAN通信协议

版本 V1.0

深圳市泰科智能伺服技术有限公司

地 址：深圳市南山区高新技术产业园南区虚拟大学园A405

电 话：0755-26712201 邮 编：518057

传 真：0755-26712958

E-mail：info@techsoftmotion.com

网 址：www.techsoftmotion.com

版权说明

目录

1.前言

1.1.缩写术语表

缩写	定义
CAN	Controller Area Network（控制器局域网）
COB	Communication Object（通信对象）
COB-ID	Communication Object Identifier（通信对象标识符）
CiA	CAN in Automation（在自动化方面的 CAN）
TML	Technosoft Motion Language（Technosoft 运动语言）

1.2.关于这部手册

泰科智能已经开发了一系列带有 CAN 接口的智能伺服驱动器，这些驱动器可以通过 2 种不同的 CAN 协议传输：TMLCAN 和 TechnoCAN。

本手册描述了 TechnoCAN 协议。

工作在 TMLCAN 协议的驱动器和工作在 TechnoCAN 协议的驱动器的不同之处是通过驱动器不同的底层固件所产生的。因此本手册指的是带有固件版本号 F200 或更高（例如 F201，F202 等）的泰科智能伺服驱动器。

1.3.什么时候用 TechnoCAN?

TechnoCAN 协议经特殊设计允许泰科智能伺服驱动器连接在标准的 CAN 网络中使用 CANopen 协议互相交换信息。TechnoCAN 和 CANopen 彼此不会互相干扰，因此它们可以同时存在于相同的物理总线上。

如果您已经使用 CAN 总线通信和 CANopen 协议，为了在同一总线上连接，您必须使用带有 TechnoCAN 协议的泰科智能伺服驱动器。

如果您之前不是一个 CAN 用户，但是您想把泰科智能伺服驱动器连接在一个 CAN 网络中，您既可以选择带有 TMLCAN 的驱动器也可以选择带有 TechnoCAN 协议的驱动器。在这 2 种情况下，驱动器设置和运动编程类似：您需要泰科智能 IPM Motion Studio 软件在您的 PC 上运行，在 PC 和其中一个驱动器之间用 RS-232 串行连接，在这种配置中，您可以访问所有的驱动器（详见 2.2 中继轴的概念）

1.4.参考书目

下列文档详细地解释了一些主要的相关参考科目：

为工业应用的 CAN 应用层：CiA DS-201...CiA DS-207, © CiA
CANopen 应用层和通信规范：CiA DS-301, © CIA

获得这些文档，更多详细资料请访问：http://www.can-cia.de

关于 TML 和 TML 指令如何被打包和通过串口或 CAN 通信传输的更多信息被提供在：

IPM Motion Studio 帮助文件

要获得这些资料，请上 www.techsoftmotion.com 下载

2. TechnoCAN

2.1.简介

泰科智能已经开发出一系列带有 CAN 接口的驱动器，这些驱动器可用于以下通信接口：

- RS232 带 CAN 一起
- RS-485（驱动器 IDM240-5EIA,IDM640-8EIA，可取代 RS-232 和 CAN-bus）

在一个 CAN 网络中您可以连接多达 31 个泰科智能伺服驱动器。

CAN 通信是多点，半双工。当一台设备发送一个信息时，连接在 CAN 网络中的所有设备都将同时收到该信息，然而，只有通过 ID 滤波器的那些信息被认可。

CAN 的主要优点是它自动地解决冲突的能力，在 CAN 网络中，如果两个设备在同时启动传送，其中一个（有较高优先级的）将总是赢得访问网络且完成传送。另外一个设备，在丢失网络访问后，将从传送转换为接收，接收更高优先级的发送信息，然后再重新尝试去传送它自身的信息。所有这个过程通过硬件（CAN-bus 控制器）自动地完成，并且透明给用户。换句话说，在 CAN 网络中的设备能作全双工模式工作，无需担心传送冲突，因为这些是自动的解决的。

2.2.中继轴的概念

您只需用 RS232 联结您的主机或 PC 与其中任一驱动器，就能与连接在 CAN-bus 网络中的所有泰科智能驱动器通信，通过 RS-232 联结，您可以：

a) 用 EasyMotion Studio 设置和编程所有连接在 CAN-bus 总线上的泰科智能伺服驱动器。
b) 从您的主机/PC 发送命令，控制所有连接在 CAN-bus 总线上的泰科智能伺服驱动器。

这台连接到主机/PC 的泰科智能伺服驱动器：

• 执行接收来自 RS-232 联结主机的命令

• 执行接收来自 CAN-bus 联结的另一台驱动器的命令

• 动作如一个转发的中继器，所以也称中继轴：

- 通过 RS-232 为另一个轴接收来自主机/PC 的命令并且通过 CAN-bus 转发到目标轴。
- 通过 CAN-bus 接收来自另一个轴的主机/PC 所请求的数据，再通过 RS-232 发送回主机/PC。

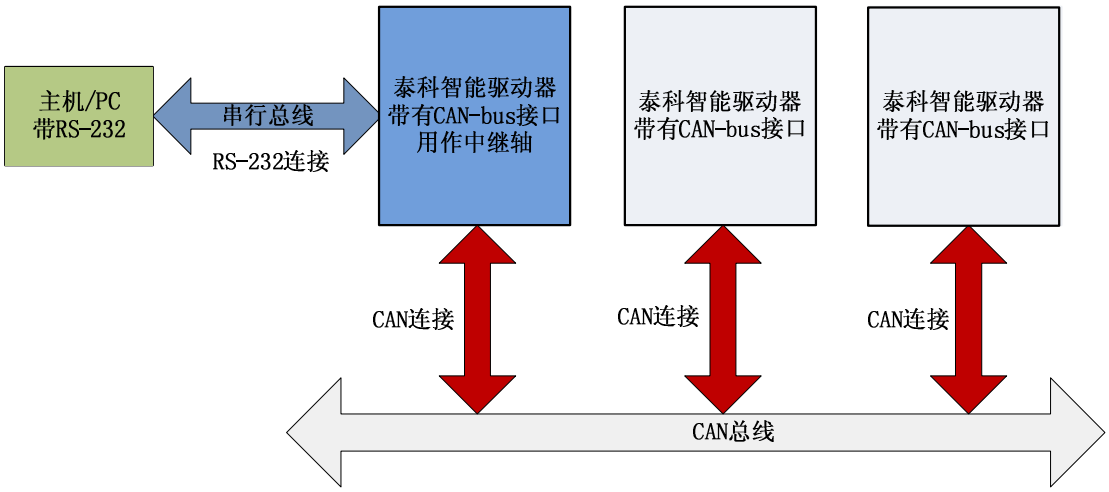


图 1.1. 中继轴概念

中继轴的概念使能主机/PC 与联结在 CAN 网络中的所有泰科智能驱动器通信，仅用一个 RS-232 连接到一台驱动器，在您的 PC 上无需 CAN-bus 接口，省掉了 CAN 接口卡的成本。**CAN-bus 协议完全透明给用户。**
任何一台泰科智能伺服驱动器当它被连接在 RS-232 与 CAN-bus 时都可做为中继轴，无需任何特殊设置，仅需要设置的是：在 EasyMotion Studio 中，主机/PC 的地址等于通过 RS-232 与其相连的驱动器的 ID 轴地址。

当 TechnoCAN 协议被使用时，通过 EasyMotion Studio 且主机与其中一台驱动器通过 RS-232 连接时设置驱动器，不会与 CAN-bus 网络上被连接的其他 CANopen 设备有任何冲突。

2.3.TML 指令格式

泰科智能伺服驱动器内嵌智能运动控制语言，使用高级 TML(Techsoft Motion Language)语言可编程复杂的运动程序，TML 指令可通过通信通道在线提供，也可以从驱动器本地 EEPROM 存储器中读取。

所有通过通信通道被接收的命令的优先级都高于从本地 EEPROM 存储器中被执行的命令。如果在执行驱动器本地 EEPROM 存储器的 TML 程序期间，驱动器通过通信通道在线接收到一个 TML 指令，这的动作好比一个中断：TML 程序执行暂停，驱动器执行在线接收到的 TML 命令，然后本地 TML 程序执行再重新开始。

所有对泰科智能伺服驱动器的操作都是通过 TML 指令来完成：驱动器设置、 下载 TML 程序、实时采集后数据上传，运动命令等等，因此，泰科智能驱动器所支持的每一个通信通道和协议的目的是为驱动器之间或主机/PC 与驱动器之间提供一个传送 TML 指令的方法。

每一个通信信息必须包含以下信息：
- 目标地址（驱动器的轴 ID 号或驱动器的组 ID 号）
- 信息接收器所能执行的 TML 指令代码

目标地址包含以下信息：
• 8-bit ID 字段，代表一个轴 ID 号（一个驱动器的地址）或一个组 ID 号（一个驱动器组的地址）
• 组 GROUP 位：0-表示 ID 字段是一个轴 ID，1-表示 ID 字段是一个组 ID
• 主机 HOST 位：0-表示中继轴，1-表示主机，当一个主机用 RS-232 与驱动器连接时，它们必须有同样的轴 ID 号，HOST 位在主机和连接到其他端如中继轴的驱动器之间做了区分，对于网络中的其他驱动器，HOST 位为 0。

TML 指令能被发送到一个驱动器/轴或一组驱动器/轴。在第一种情况，ID 字段通过唯一的 8 位 **Axis ID** 码定义。在第二种情况，目标通过 **Group ID** 码定义。对于广播发送，Group ID 表示了一种识别一组驱动器的方法。该特性允许同时控制几个驱动器，例如同时开始或停止轴的运动。每个驱动器可以被编程成为一个或几个组中的一个成员。在这个 8 组之内，驱动器将接收到所有被送入组成员的信息，例如，如果驱动器是组 1 和 3 中的成员，它将接收组 ID 中组 1 和组 3 的所有信息。组 ID 是一个 8 位的值，每组表示一个 8 位组 ID （见下表）。组 ID 中的一个轴可以是 0 到 255 之间的任意值，例如，如果一个轴的组 ID 是 11(1011 二进制)，意味着这个轴将接收所有被发送到组 1，2 和 4 的信息。

表 1.组的定义

组号	组 ID 的值
1	1 (0000 0001)
2	2 (0000 0010)
3	4 (0000 0100)
4	8 (0000 1000)
5	16 (0001 0000)
6	32 (0010 0000)
7	64 (0100 0000)
8	128 (1000 0000)

备注：上电后，所有的驱动器都被默认设置为 Group ID = 1。带有 CAN 接口的驱动器轴 ID 从地址开关中读取。

下面举例说明 HOST 位被如何使用：假设我们有两台驱动器分别为轴 ID=1 和轴 ID=2（值 1 和 2 表示 ID 字段的值）通过 CAN-bus 连接，主机通过 RS-232 连接到 ID=1 的驱动器作为中继轴，主机 ID 号也必须为 1 而且 HOST 位被设置为 1，主机发送一个数据请求信息给轴 ID=2 驱动器，该信息包含发送（**sender**）轴 ID 代码等信息，在这儿，ID=2 驱动器必须返回所请求的数据。发送(**sender**)轴 ID 代码是主机地址（ID=1 且 HOST 位设置为 1），这个请求的信息通过 RS-232 被发送给轴 ID=1 驱动器，该驱动器侦测到这个信息的目的地址是另一轴（例如 ID=2），它将通过 CAN-bus（CAN 总线）重新转发该信息给轴 ID=2 驱动器，轴 ID=2 驱动器将接收这个请求信息且通过 CAN-bus 发送回答信息给发送（**sender**）轴（例如主机），因为主机和中继轴有相同的地址，所有通过 CAN-bus 发送且把主机作为目的地址的所有信息都将由中继轴接收，中继轴看待 HOST 位：如果该位被设置，那么所接收的信息将通过 RS-232 送回主机，如果 HOST 位没有被设置，那么接收的信息将被执行（它的目的地址就

是中继轴）。

TML 指令代码有 1 到 5 个字，所有的 TML 指令至少有一个字-**操作码**。依据 TML 指令的类型，可以有 0-4 个数据字。

备注：第 3 章详细介绍了如何找查 TML 指令的二进制码。

2.4.TML 指令类型

TML 指令分为两种类型：
• **A 类型**：TML 指令无需应答（返回信息），在这类指令中，例如包含参数设置，开始或停止运动执行等明亮的 TML 指令。
• **B 类型**：TML 指令需要应答，在这类指令中，TML 指令包含要求返回数据的命令，例如 TML 参数，寄存器或变量的值的 TML 指令。

一个 B 类型 TML 指令由两部分组成：
• 一个数据请求
• 一个包含请求数据的应答

一个为 B 类型信息的典型例子是当主机或驱动器请求另一个驱动器返回它 TML 变量的一个值时，用数据请求 TML 指令“Give Me Data”和应答 TML 指令“Take Data”完成 。

“Give Me Data”数据要求有以下代码：

表 2. “Give Me Data”TML 代码

操作码：B004h 为 16 位数据，**B005h** 为 32 位数据
数据（1）：发送轴 ID
数据（2）：被请求的数据地址

“Take Data”应答有以下代码：

表 3. “Take Data”TML 代码

操作码：B404h 为 16 位数据，**B405h** 为 32 位数据
数据（1）：发送轴 ID
数据（2）：被请求的数据地址
数据（3）：被请求的数据 16LSB
数据（4）：被请求的数据 16MSB(为 32 位数据)

备注：第 3 章详细介绍了如何查找所有包含“Give Me Data”和“Take Data”TML 指令的二进制代码。

2.5. TML 指令在 TechnoCAN 中映射

在 TechnoCAN 中 TML 指令被划分为 4 类：

a) **Normal**-包含所有 A 类型和所有要求如“Give Me Data” B 类型的 TML 指令
b) **Take Data**-包含 B 类型 “Take Data”应答
c) **Group**-包含寻址一组驱动器的所有 A 类型 TML 指令
d) **Host**-包含所有除“Take Data”之外需应答的 B 类型 TML 指令

被映射到 CAN 信息的 TML 指令使用以下规则：

- 每条 CAN 信息传送一条 TML 指令
- COB-ID 的分配以允许在同一网络中与 CANopen 设备共存的方法来完成
- 8 位字段的轴 ID 或组 ID 被限制到 5 位（LSB），因为在 CAN 总线上可连接的泰科智能驱动器的最大数目是 31（轴 ID：1 到 31，轴 ID：0 被保留），因此组的数目也被从 8 减到 5。
- 除 **Take Data** 之外，所有其他 TML 指令长度最多 8 个字节（操作码+数据）且可以在一个 CAN 信息中被传送，由于在 TML 目标地址包含：地址值，HOST 位和 GROUP 位，TML 指令被划分为 3 类 TechnoCAN 信息以适应于所有的指令组合。
- TechnoCAN 信息的分类已经预先考虑了“Take Data”TML 指令。

2.5.1. Nomal 信息：COB-ID: 121h – 13Fh

	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	0	1	0	0	1	轴 ID				
字节 0	操作码[7...0]										
字节 1	操作码[15...8]										
字节 2	数据(1) [7...0]										
字节 3	数据(1) [15...8]										
字节 4	数据(2) [7...0]										
字节 5	数据(2) [15...8]										
字节 6	数据(3) [7...0]										
字节 7	数据(3) [15...8]										

2.5.2.HOST 信息：COB-ID: 141h – 15Fh

	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	0	1	0	1	0	轴 ID				
字节 0	操作码[7...0]										
字节 1	操作码[15...8]										
字节 2	数据(1) [7...0]										
字节 3	数据(1) [15...8]										
字节 4	数据(2) [7...0]										

字节 5	数据(2) [15...8]
字节 6	数据(3) [7...0]
字节 7	数据(3) [15...8]

备注：主机信息仅发生在当驱动器应答主机请求（“Give Me Data”）且仅在主机是否通过 RS-232 连接在网络中通过中继轴发送请求信息的时候，主机信息不会发生在直接连接在 CAN 总线上的驱动器或主机发送的请求信息的时候。

2.5.3. Take Data 信息：COB-ID: 161h – 17Fh

	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	0	1	0	1	1	轴 ID				
字节 0	操作码[7...0]										
字节 1	数据(1) [8...4]						H	操作码[9...8]			
字节 2	数据(2) [7...0]										
字节 3	数据(2) [15...8]										
字节 4	数据(3) [7...0]										
字节 5	数据(3) [15...8]										
字节 6	数据(4) [7...0]										
字节 7	数据(4) [15...8]										

备注：在 Take Data 信息中， 10-字节代码的“Take Data” TML 指令被压缩到 8-字节，这是以下列方法来完成的：

• 从 16 位操作码，仅开始的 10LSB 被传送，余下的 6MSB 总是常量：0x2D (101101b)且不被传送。Take Data 信息的接收者必须将 0x2D 加到已接收的 6MSB 操作码上为了恢复成完整的 16 位 Take Data 指令代码。

• HOST 位在字节 1 的位 2 被发送，在这儿无需发送 GROUP 位，因为一个数据请求不能被发送到一组驱动器。

• Take Data TML 指令的第一个数据字是发送（**sender**）轴 ID 号，这个字段的形式详见第 3 章。它包含在应答 Take Data 驱动器的 8 位轴 ID 值的 11-4 位上，由于驱动器最多轴数被限制为 31，仅 8-4 位被用且被传送。

2.5.4. Group 信息：COB-ID: 001h – 01Fh

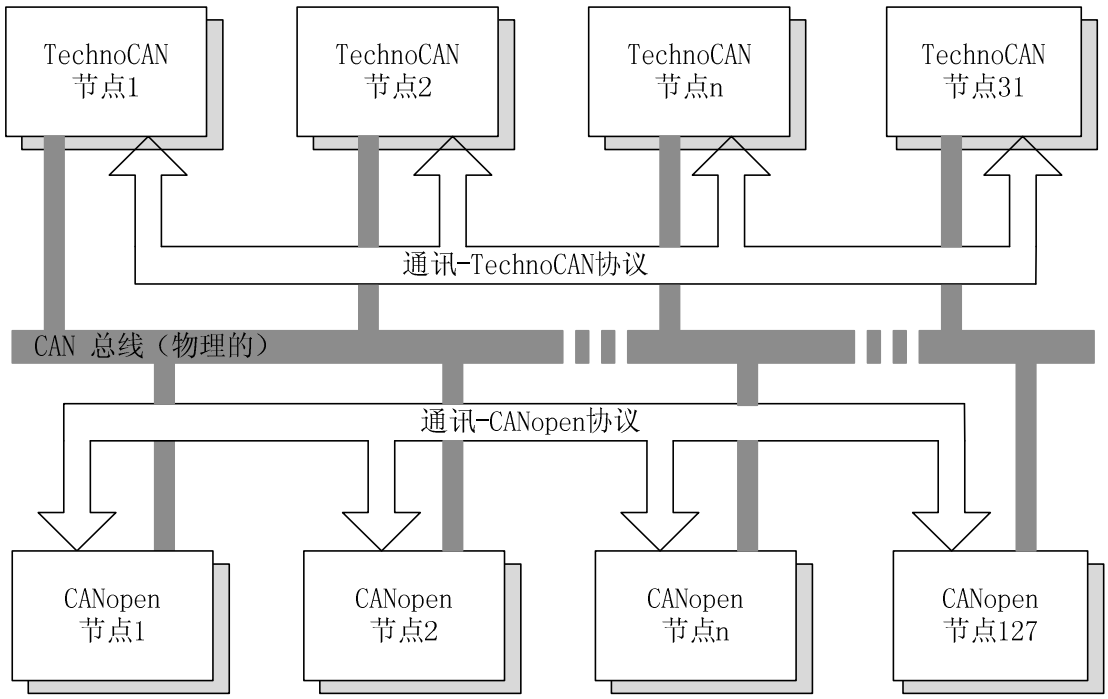
	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	0	0	0	0	0	组 ID				
字节 0	操作码[7...0]										
字节 1	操作码[15...8]										
字节 2	数据(1) [7...0]										
字节 3	数据(1) [15...8]										
字节 4	数据(2) [7...0]										
字节 5	数据(2) [15...8]										

字节 6	数据(3) [7...0]
字节 7	数据(3) [15...8]

备注：如果一台驱动器发送一个组信息给其所在的同一组，这台驱动器将不会接收到自己所发送的信息。

2.6. TechnoCAN 和 CANopen 不会彼此干扰

TechnoCAN 仅使用 CANopen 使用范围之外的 COB-IDs，因此，TechnoCAN 协议和 CANopen 协议能共存且可以在同一物理 CAN 总线上同时通信，不会互相干扰。



下列表总述了 TechnoCAN 所占用的 COB-IDs 范围，表 2 显示了这些 COB-ID 在与 CANopen 关系中是如何分配的。

表 2. TechnoCAN　COB-IDs 范围

描述	COB-ID 范围	
	Dec（十进制）	Hex（十六进制）
Group 信息	1-31	1-1F
Normal 信息	289-319	121-13F
HOST 信息	321-351	141-15F
Take Data	353-383	161-17F

表 3. TechnoCAN　COB-ID 分配

COB-ID 值	在 **CANopen** 中 COB-ID	在 **TechnoCAN** 中 COB-ID
0h	使用-NMT	不用
1 -1F	不用	使用-Group 信息
20 -7Fh	不用	不用
80h	使用- SYNC	不用
81- FFh	使用- EMERGENCY	不用
100h	使用- TIME STAMP	不用
101 -11Fh	不用	不用
120h	不用	不用
121 -13Fh	不用	使用-Normal 信息
140h	不用	不用
141 - 15Fh	不用	使用-HOST 信息
160h	不用	不用
161-17Fh	不用	使用- Take Data 信息
180h	不用	不用
181 - 19Fh	使用- PDO1 (tx)	不用
1A0 - 1FFh	使用- PDO1 (tx)	不用
200h	不用	不用
201 -21Fh	使用- PDO1 (rx)	不用
220 -27Fh	使用- PDO1 (rx)	不用
280h	不用	不用
281 -29Fh	使用- PDO2 (tx)	不用
2A0 -2FFh	使用- PDO2 (tx)	不用
300h	不用	不用
301 - 31Fh	使用- PDO2 (rx)	不用
320 -37Fh	使用- PDO2 (rx)	不用
380h	不用	不用
381 -3FFh	使用- PDO3 (tx)	不用
400h	不用	不用
401 - 47Fh	使用- PDO3 (rx)	不用
480h	不用	不用
481- 4FFh	使用- PDO4 (tx)	不用
500h	不用	不用
501- 57Fh	使用- PDO4 (rx)	不用
580h	不用	不用
581 -5FFh	使用- SDO (tx)	不用
600h	不用	不用
601- 67Fh	使用- SDO (rx)	不用
680 -6FFh	不用	不用

700h	不用	不用
701 -77Fh	使用-NMT 误差控制	不用

2.7.例子

2.7.1.例 1.设置位置控制器的比例 P 参数，驱动器轴 ID=5

主机被直接连接在一个带泰科智能驱动器的 CAN-bus 网络中，想要给轴 ID=5 驱动器发送 TML 指令“kpp = 0x1234”（设置位置控制器的比例参数值为 1234h），TML 指令的代码是：

操作码=205Eh
数据（1）=1234h

TechnoCAN 信息是：

	10	9	8	7	6	5	4	3	2	1	0
COB-ID（轴 ID=5）	0	0	1	0	0	1	0	0	1	0	1
字节 0(5Eh)				0	1	0	1	1	1	1	0
字节 1 (20h)				0	0	1	0	0	0	0	0
字节 2(34h)				0	0	1	1	0	1	0	0
字节 3 (12h)				0	0	0	1	0	0	1	0
字节 4				0	0	0	0	0	0	0	0
字节 5				0	0	0	0	0	0	0	0
字节 6				0	0	0	0	0	0	0	0
字节 7				0	0	0	0	0	0	0	0

备注：*最后 4 个字节不用且不被传送。*

2.7.2.例 2.询问轴 ID=5 驱动器的位置误差是多少

主机被直接连接在一个带泰科智能驱动器的 CAN-bus 网络中，想要从轴 ID=5 的驱动器获得位置误差值。主机 ID=3，位置误差是一个 16 位的变量名为 POSERR 位于存储器地址 0x022A 中。

请求命令“Give Me Data”包括以下内容：

操作码= B004h
数据（1）=发送轴 ID=0030 h
数据（2）=被请求的数据地址= 022Ah

TechnoCAN 信息是：

	10	9	8	7	6	5	4	3	2	1	0
COB-ID（轴 ID=5）	0	0	1	0	0	1	0	0	1	0	1
字节 0 (04h)				0	0	0	0	0	1	0	0
字节 1 (B0h)				1	0	1	1	0	0	0	0
字节 2 (30h)				0	0	1	1	0	0	0	0
字节 3 (00h)				0	0	0	0	0	0	0	0
字节 4 (2A)				0	0	1	0	1	0	1	0
字节 5 (02)				0	0	0	0	0	0	1	0
字节 6				0	0	0	0	0	0	0	0
字节 7				0	0	0	0	0	0	0	0

备注：*最后的 2 个字节不用且不被发送。*

假如位置误差值是 2，“Take Data”应答将有以下代码：

操作码= B404h
数据（1）=被请求的数据地址= 022Ah
数据（2）=数据请求= 0002h

TechnoCAN 信息是：

	10	9	8	7	6	5	4	3	2	1	0
COB-ID（轴 ID=3）	0	0	1	0	0	1	0	0	0	1	1
字节 0 (04h)				0	0	0	0	0	1	0	0
字节 1 (28h)				0	0	1	0	1	0	0	0
字节 2 (2Ah)				0	0	1	0	1	0	1	0
字节 3 (02h)				0	0	0	0	0	0	1	0
字节 4 (02h)				0	0	0	0	0	0	1	0
字节 5(00h)				0	0	0	0	0	0	0	0
字节 6				0	0	0	0	0	0	0	0
字节 7				0	0	0	0	0	0	0	0

备注：*最后 2 个字节不用且不被发送。*

3.如何查找 TML 指令二进制代码

情况 1. 您想查找不包含用户定义变量(**user-defined variables**，已经声明和命名的变量等)的 TML 指令代码。

方法：用 **IPM Motion Studio** 中的 **Binary Code Viewer** ，可以用以下菜单命令打开：**Tools | Binary Code Viewer…**在 **Source code** 编辑栏输入 TML 指令，选择 **Protocol** “TML”并且按下箭头按钮，在 **Binary code sent**，您将看到指令的 TML 代码、字结构、起始操作代码后紧跟指令数据。您也可以查找“Give Me Data” 和 “Take Data”TML 命令的 TML 代码，用于从另一个驱动器请求数据。在 **Source Code** 编辑栏输入一个问号？后面加您想要读取的变量名字，例如：?poserr（给我位置误差值），选择“Give Me Data”所请求的发送轴 ID，特别说明它是否为主机（如果它通过 RS-232 被连接到中继轴，这个主机必须被选择）。选择“Give Me Data”请求被发送的目标 **Destination** 轴 ID，保留 **Protocol** “TML”协议且按下箭头按钮，在 **Binary code sent**，您将看到包含指令数据字 **Sender** 的轴 ID 的指令“Give Me Data”的 TML 代码。在 **Binary code received**，您将看到包含带 “Take Data”应答 **Destination** 轴 ID 指令数据字的指令“Take Data”的 TML 代码，同时在指令数据中也有当作请求变量的返回值，这个值自己可以设置。

备注：*对于所有不包含用户定义的变量（user-defined variables）的 TML 指令，您可以使用 Binary Code Viewer 查找到使用 RS-232/485 或 TMLCA 通信，主机发送/接收的相关信息的精确代码。*

情况 2. 任何 **A** 类型 TML 指令，包含用户定义变量（*user-defined variables*）（指已经申明和命名的变量），A 类型 TML 指令不需要应答（返回信息），在这类指令中，如包含参数设置命令、开始或停止运动执行命令等信息。

方法：

- 打开一个 IPM Motion Studio 工程文件或创建一个新文件，如果工程中有多个应用文件，选择其中一个应用文件
- 用 Motion Wizard 编辑您想要查找其代码的 TML 指令，如果 TML 指令包含用户定义变量（*user-defined variables*），那么在第一次使用前，您必须申明
- 选择菜单命令 **Project | Settings** 和 tab **Compiler**
- 如果 compiler tab 的内容被禁止，请同时按下 **CTRL, SHIFT, ALT** 和 **A** 键。这个操作允许您修改默认设置。
- 检查选择项：**Keep ASM File** 和 **Interlist TML Source with ASM Source**
- 用菜单命令 **Build | Generate Code** 重新编译这个应用，然后 **Build |Rebuild All**，该操作将在您的应用文件夹中创建一个带扩展名***.asm** 的文件，这个文件列出了您 TML 程序的所有 TML 指令并跟随有二进制代码。
- 用菜单命令 **File | Open** 打开**.asm file**。在 **Files of type**，选择“All files (*.*)”。File | Open 命令默认指向 IPM Motion Studio 工程子文件夹，选择有与您工程同名的子文件夹， 子文件夹内有所选择同名的应用文件，在这儿您将找到与所选择的应用程序同名的***.asm**

文件。

- 用菜单命令 **Edit | Find** 查找您想找其二进制代码的指令的 TML 源代码，在每一个指令后，您将看到它的二进制代码。

例子：查找 KIP = 20(0x0014)TML 指令的二进制代码，在*.asm 文件中，二进制代码出现为：

.word 02060h
.word 00014h

因此，TML 指令有 2 个字，第一个字是操作代码且有十六进制值 2060h，第二个字是指令数据且有十六进制值：14h

备注：*一旦您了解了 TML 指令的二进制代码，您可以用任何通信协议发送该指令，只需要对各种通信协议运用打包规范建立被发送的信息包即可。*

情况 3. B 类型的 TML 指令，包含用户定义的变量（*user-defined variables*）（已经申明和命名的变量），B 类型 TML 指令要求一个应答，在这类指令中，TML 指令请求返回数据，如 TML 参数、寄存器或变量的值。

方法：

B 类型 TML 指令有两个组成部分：
• 通过 TML 指令“Give Me Data”发送请求的数据
• 通过 TML 指令“Take Data”发送数据请求应答

“Give Me Data”请求指令有以下二进制代码：

操作码：B004h 为 16 位数据 B005h 为 32 位数据
数据（1）：发送轴 ID
数据（2）：请求的数据地址

TML 指令“Give Me Data”的二进制代码

“Take Data”应答指令有以下二进制代码：

操作码：B404h 为 16 位数据 B405h 为 32 位数据
数据（1）：发送轴 ID
数据（2）：被请求的数据地址
数据（3）：被请求的数据 16LSB
数据（4）：被请求的数据 16MSB（为 32 位数据）

TML 指令"Take Data"的二进制代码

如果您已经申明了这个变量，并想要查找其整数类型 **int** (integer)值，它是一个 16 位数据，否则对于 **fixed** 或 **long** 类型，它是一个 32 位数据。

发送轴 ID 代表了要发送一系列命令的设备地址，在"Give Me Data"中它是请求一个变量值的设备，在"Take Data"中它是一个提供被请求数据的设备，在两种情况中发送轴 ID 有以下格式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	G	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	0	0	0	H

其中：
位 0-HOST 位。0-驱动器（中继轴），1-主机，当主机通过 RS-232 与驱动器相连时，两台设备必须有相同的轴 ID（位 ID7-ID0 是相等的），HOST 位区分了它们。在 RS-232 通信时,主机发出"Give Me Data"请求，驱动器用"Take Data"应答。在 RS-485 通信时，主机和驱动器有不同的轴 ID，HOST 位没有意义且必须设为 0。
位 11-8-ID7-ID0:一个轴 ID 的 8 位值
位 12-GROUP 位：0-ID7-ID0 值是一个轴 ID 值，1-ID7-ID0 是一个组 ID 值，在执行"Give Me Data"与"Take Data"TML 命令时，它总是设置为 0，因为这些命令不能被发送给一组驱动器。

Requested Data Address 代表了被请求值的变量地址，您可以用下列方法找到任何 TML 变量（预定义或用户定义）的地址：
• 用菜单命令 **View | Watch** 打开 **Watch** 窗口并且从下拉列表中选择您想要查找其地址的变量名称，按 Enter 验证您的选择，Watch 窗口自动显示变量类型和地址。

备注：*用户定义变量（user-defined variables）在Watch列表中也是可查找的，但仅在TML程序（包括用户定义变量申明）至少被编译了一次，如果您想在没有执行应用程序时（没有按下Run按钮）编译程序，您能用菜单命令**Build | Generate Code** 后再 **Build | Rebuild All**进行编译。*


TECHSOFT
Motion Control
泰科智能