

TECHSOFT
MOTION CONTROLS

泰科智能伺服驱动器

通信协议

RS-232/RS-485/CAN-bus

技术参考手册

深圳市泰科智能伺服技术有限公司

目录

第 1 章	通信协议概述	2
第 2 章	信息结构-轴ID与组ID的概念	4
第 3 章	串行通信RS-232 与 RS-485 协议	7
第 4 章	CAN-bus通信TMLCAN协议	15
第 5 章	CAN-bus通信TechnoCAN协议	21

第 1 章 通信协议概述

本章描述了泰科智能伺服驱动器所支持的通信协议，详细介绍了TML(Techsoft Motion Language)指令为RS232/485/CAN-bus通信通道被如何压缩成数据信息的技术参考。

这些信息对于使用其他外部设备（如单片机、触摸屏等）像PC主机一样直接与泰科智能伺服驱动器通信的应用尤其有用，在这种应用中，主机打包每一个TML命令成一个二进制代码数据信息包被发送，且解压每一个所接收到的信息包，从中得到所提供的数据。

备注：一种可替代的方法是，通过**TML_LIB**运动函数库与泰科智能伺服驱动器交换数据，**TML_LIB**函数库是一个为运动编程准备的高级函数的集合，您可以把它集成到主机或主控制器的应用程序中。如果主机是一台工业PC，**TML_LIB for PC**函数库可以被集成到**C/C++**，**Delphi Pascal**，**Visual Basic**或**LabVIEW**应用中。如果主机是一台可编程逻辑控制器（**PLC**），**TML_LIB for PLC**版本，兼容**PLCopen**标准，为运动编程可以被集成到**PLC IEC 61131-3**应用中（关于**TML_LIB**库的详细资料请浏览泰科智能网站www.techsoftmotion.com）

根据不同的驱动器，您可以使用两种类型的通信通道：

- 串行RS-232或RS-485
- CAN-bus

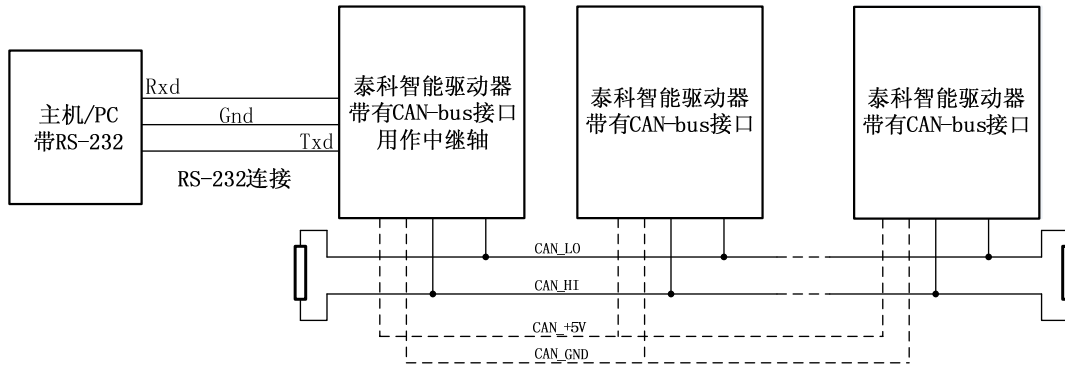
RS-232串行通信通道一般用于连接一个主机与一个驱动器，串行RS-485和CAN-bus通信通道一般可用于连接一个主机与多达32个以上的驱动器。

备注：RS-485和CAN-bus协议可接受多达255个节点。受限制的32个节点由硬件所决定，用于常规的处理。如果您的应用需要多于32个轴，请联系我们。根据您的驱动器与网络特性，我们为您提供所需用的最多轴数。

当使用CAN-bus通信时，网络中的任一驱动器也可以通过RS-232或Ethernet与主机相连接。在这种情况下，这个驱动器

- 通过RS-232联结，执行接收到来自主机的命令
- 通过CAN-bus联结，执行接收到来自网络中其他驱动器的命令
- 这个驱动器类似一个中间转播信息的作用，因此也称这个轴为**中继轴**
 - 通过RS-232，为其他轴接收来自主机的命令，再通过CAN-bus转发这些命令给目标轴
 - 通过CAN-bus，由主机向其他轴请求数据，再通过RS-232转发这些数据给主机

如下图所示：



中继轴概念

中继轴的概念能使一台主机可以与来自CAN-bus网络中的所有泰科智能伺服驱动器进行通信，只需用一个的RS-232串口或一个Ethernet与一台驱动器相连即可，而无需主机带CAN-bus接口，在这里，CAN-bus协议是完全透明的。

任一驱动器被作为中继轴使用时，当它被同时连接到RS-232与CAN-bus时，无需任何特殊设置，只需要设置主机的轴ID地址等于通过RS-232连接的驱动器的轴ID地址即可（详情请看第2章[信息结构-轴ID与组ID的概念](#)）

重要提示！ EasyMotion Studio 包含一个二进制代码浏览器 [Binary Code Viewer](#)（以下简称为二进制代码浏览器），它可以帮助您运用驱动器所支持的通信通道与协议，快速找到如何发送TML命令的二进制代码。使用该工具，您可以获得要发送信息的准确内容，同时也可获得这些被发送信息所期望的应答。

关于通信协议详细描述，详见下面章节：

- 第2章.....[信息结构-轴ID与组ID的概念](#)
- 第3章.....[串行通信RS-232和RS-485协议](#)
- 第4章.....[CAN-bus通信TMLCAN协议](#)
- 第5章.....[CAN-bus通信TechnoCAN协议](#)

第 2 章 信息结构-轴ID与组ID的概念

任何通信总线的数据交换与协议都是用信息来完成的。每条信息都包含一条由该信息接收器执行的 TML 指令。除 TML 指令附带的二进制代码之外，任何信息都包含关于它的目标的信息：一个轴(驱动器或电机)或一组轴。该信息被集合在 **Axis/Group ID Code** 中。根据所使用的通信总线与协议，轴/组 ID 代码与 TML 指令附带的二进制代码以不同的方式被压缩：

包含在一个通信信息中的信息

轴/组 ID 代码
操作码
数据 (1)
...
数据 (4)

第一个字 **Axis/Group ID Code** 识别所必须接收信息的目的轴或一组轴。第二个字表示被发送的 **TML 指令** 的编码 (操作码)。

Axis/Group ID Code 是一个 16 位的字，具有以下结构：

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-中继轴，1-主机。当一个主机用 RS-232 连接到一个驱动器时，两个设备必须有相同的轴 ID (位 ID7-ID0 是相同的)。HOST 位用于区别主机与连接其他目标驱动器时的不同。在 RS-485 通信中，主机与驱动器必须有不同的轴 ID,HOST 无任何意义且必须被设置为 0。

位 11-4(D7-ID0)：轴 ID 或组 ID 的 8 位值

位 12-GROUP(组)位：设置为 0 时 ID7-ID0 值是一个轴 ID，设置为 1 时 ID7-ID0 值是一个组 ID

根据所使用的通信总线与协议，全部 16 位轴/组 ID 代码包含在一个信息中或者只有它的一部分包含在一个信息中。该部分也可以是一个带有有用信息的 10 位：HOST 位，ID7-ID0 位和 GROUP 位或它们的子集。

备注：在以下的章节中，术语 **Axis ID Code** 或 **Group ID Cod** 被指定为上面所描述的 16 位字。术语 **Axis ID** 和 **Group ID** 被指定一个轴或组 ID 的 8 位值即位 ID7-ID0 的值。

以下例子描述了如何使用 **HOST** 位：

让我们假设有两台驱动器：轴 ID=1 和轴 ID=2 (值 1 和 2 表示位 ID7-ID0 的值)，它们之间通过 CAN-bus 连接。主机通过 RS-232 连接到轴 ID=1 的驱动器，该驱动器当作一个中继轴用。主机轴 ID (主机 ID) 也必须是 1 但是该主机位(HOST 位)被设置。主机发送一个数据请求信息给轴 ID=2 的驱动器，该请求信息的轴 ID 代码是 2 也即目标轴。该信息包含发送轴 (**sender axis**) ID 代码即轴 ID=2 的驱动器必须发送所被请求数据的来源地。发送轴 ID 代码是主机地址 (ID=1 且 HOST 位设置为 1)。请求信息通过 RS-232 发送到轴 ID=1 的驱动器，该驱动器注意到信息的目

标是另一个轴（例如 ID=2）并且通过 CAN-bus 转发信息。轴 ID=2 的驱动器将接收到请求信息并通过 CAN-bus 发送应答给发送轴（即主机）。因为主机与中继轴地址相同，所有通过 CAN-bus 发送的、且主机作为目标轴的信息都将被中继轴接收。中继轴查看 HOST 位：如果该位被设置为 1，那么所接收的信息将通过 RS-232 发送回主机。如果 HOST 位为 0，那么所接收到的信息将被执行（它的目标轴就是中继轴）。

一个信息可以被发送给一个轴或一组轴。在第一种情况中，目标通过 **Axis ID** 代码指定。在第二种情况中，目标通过 **Group ID** 代码指定。每一个驱动器都有它自己的 8 位轴 ID 与组 ID,保存在 TML 寄存器的 AAR 中。如果一个信息的目标是由一个轴 ID 代码指定，那么信息仅由与所指定的轴 ID 代码相同的 8 位轴 ID 的轴（16 位轴 ID 代码中的位 11-4）接收。如果目标通过组 ID 代码指定，每一个轴将信息的 8 位组 ID 与自己的组 ID 做比较，如果两组 ID 中有至少一个组（组位设置为 1）是相同的，那么信息将被接收。在组 ID 中，每位相对应一个组：

组的定义

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

一个驱动器可被编程为最多是 8 个组中的成员。它将接收发送给它所在成员的任何组的所有信息。例如，如果一个驱动器是组 1,2 和 4 的成员即它的 8 位组 ID=11(00001011b)，它将接收到至少有位 0,1 或 3 中被设置为 1 的一个组 ID 的所有信息。

备注：

- 带轴 ID=0 的信息都将被接收，与接收者的轴 ID 无关
- 一个广播信息的组 ID=0 且将被网络中的所有轴接收，与它们的组 ID 无关

对于每个驱动器，它的轴 ID 在上电时运用以下算法在上电时被初始设置：

- 从驱动器 EEPROM 参数设置表中读取所有设置数据的值。
- 如果这个设置表是无效的，则用读取一个有效的设置表所获得的上一次的轴 ID 值
- 如果因为一个有效的设置表无轴 ID 设置，则用读取硬件拨码开关或跳线器的状态值，设置为轴 ID 的值
- 如果驱动器没有硬件开关或跳线器为轴 ID 设置，则默认的轴 ID 值是 255

备注：如果从一个有效的设置表中读取的轴 ID 是 0（选项为 H/W），则轴 I/D 被设置为从硬件开关或跳线器状态读取的值，或当硬件开关或跳线器不存在时则设置为默认值 255。

对于每个驱动器，组 ID 被默认设置为 1，即所有的驱动器都是组 1 的成员，对于每个驱动器您可以：

- 设置或改变它的组ID，使用TML指令[GROUPID](#)
- 增加新的组号到它的组ID，使用TML指令[ADDGRID](#)
- 从它的组ID删除组号，使用TML指令[REMGRID](#)

备注：您可以在任何时刻从轴地址寄存器[ARR](#)读取一个驱动器Axis ID与Group ID的实际值。

TML instruction 指令代码可以有 1 到 5 个字长，所有的 TML 指令必须至少有一个字-操作码 **Operation Code**。根据 TML 指令的类型，操作码可以跟随 0-4 个 **Data** 数据字。

备注：使用[二进制代码浏览器](#)可获得所有通信协议下所有的TML指令的二进制代码。

第 3 章 串行通信RS-232 与 RS-485 协议

泰科智能的所有智能伺服驱动器都能通过RS-232进行通信。其中某些型号的驱动器也可以通过RS-485进行通信，替代RS-232。在下面的章节中，术语**串行通信**引用了RS-232和RS-485的共同特性。术语**RS-232通信**或**RS-485通信**用于其中一个或其他指定的特征。

RS-232通信是点到点，全双工，且可以让您在2台设备间联结进行通信。典型例子如您通过RS232连接您的PC与泰科智能驱动器进行通信。

通过RS-232通信，您可以：

- a) 使用泰科智能软件开发平台**EasySetUp**或**EasyMotion Studio**设置驱动器、电机参数或编程运动
- b) 通过RS232，从主机发送命令控制一个驱动器
- c) 通过CAN-bus连接的数个驱动器，设置驱动器、电机参数或编程运动，其中一个驱动器通过RS-232与PC连接
- d) 通过CAN-bus连接，从主机发送命令给通过RS-232连接的其中一个驱动器控制数个驱动器

在c) 与 d) 的情况下，通过RS232连接到主机的泰科智能驱动器被称为中继轴（详情请看[通信协议概述](#)）

RS-485通信是多点，半双工，可以使您在一个网络中连接多达32个驱动器。在一个RS-485网络中，在某一时刻只允许一个设备发送数据，如果两个设备在同一时刻被错误地传送，那么两个传送都会被丢失。因此为了系统正确的运行，在一个RS-485网络中必须有一个主控制器控制传送。换句话说，只有当主控制器初始化一个传送时，主控制器请求网络中所有其他设备提供数据时，其他设备才可以传送。通常您应该设置您的主机作为主控制器。

使用RS-485通信，您可以：

- a) 通过RS-485将数个驱动器与您的PC连接在一起，设置驱动器、电机参数或编程运动（在您的PC上要求带一个RS-485接口或一个RS-232转RS-485适配器）
- b) 通过RS-485连接，从主机发送命令控制数个驱动器。主机可以被看做为RS-485网络中的一个节点，且必须当作主控制器用。

备注：如果缺少主机，您也可以使用任一驱动器做为主控制器控制RS-485通信，这是由于强大的TML指令集为多轴控制提供的可能性。

串行通信设置与信息包

泰科智能伺服驱动器使用8个数据位，2个停止位，无奇偶校验位，在以下波特率：9600（复位后的默认值），19200，38400，56600和115200bit/s通过RS232联结进行通信，通过串行通信所交换的信息以下列格式打包：

串行信息结构-TML指令包

字节1：信息长度
字节2：轴/组ID代码-高字节
字节3：轴/组ID代码-低字节
字节4：操作代码-高字节
字节5：操作代码-低字节
字节6：数据（1）-高字节
字节7：数据（1）-低字节
字节8：数据（2）-高字节
...
字节13：数据（4）-低字节
最后字节：校验和

信息长度字节包含信息的整个字节数减2。换句话说，字节长度值的是以下字节的字节数：**轴/组ID码**（2个字节），**操作码**（2个字节）和**数据**字（从0到8个字节的变量）。**校验和**字节是除了校验和自身外，信息的所有字节的和的256的模。

串行通信信息类型

串行通信协议基于3种类型的TML命令压缩包的信息：

- 类型A：信息不需要一个应答（或一个返回信息）。这些信息既可以由主机发送，也可以由其他驱动器发送，包含的TML指令如：执行参数设置，运动编程，电机命令等。
- 类型B：信息需要一个应答。这些信息由主机发送且包含一个在线TML指令。这些命令请求返回数据，例如TML参数值，寄存器或变量的值。
- 类型C：无需主机请求，信息由驱动器发送给主机。当一个指定的条件发生时或跟着执行TML发送命令时，这些信息都可以被发送（详情请看[信息发送到主机](#)章节）

下一章给出了每一种信息类型的例子。

例子1-类型A信息：主机通过RS-232连接到驱动器并且发送TML指令“**KPP=5**”（设置位置控制器的比例部分为5）。主机与驱动器的轴ID是255=0FFh。轴ID代码与TML指令二进制代码是：

轴ID代码+ TML指令**KPP=5**发送到轴255的二进制代码

轴ID代码 (IDC) =0FF0h (目标轴)
操作码=205Eh
数据字 (1) =0005h (在KPP中设置的数据)

备注: 用[二进制代码浏览器](#)获取TML指令的二进制代码

主机必须发送包含有以下内容的一连串信息

连串信息: TML指令KPP=5发送到轴255

字节1: 06h-长度: IDC=2,操作代码=2, 数据=2
字节2: 0Fh-轴ID代码的高字节=0FF0h(目标轴)
字节3: F0h-轴ID代码的低字节=0FF0h (目标轴)
字节4: 20h-操作代码的高字节=205Eh
字节5: 5Eh-操作代码的低字节=205Eh
字节6: 00h-数据 (1) 的高字节=0005h (在KPP中设置的数据)
字节7: 05h-数据 (1) 的低字节=0005h (在KPP中设置的数据)
字节8: 88h-校验和

驱动器将返回一个字节0X4F作为对信息接收OK的确认。(详情请看下面的RS-232和RS-485协议细节描述)

备注:

- a) 如果另一个带轴ID=1的驱动器通过CAN-bus与带轴ID=255的驱动器相连接, 且主机想要发送相同的TML指令“KPP=5”到轴1, 那么上表中轴ID代码应该变为0010h, 替代0FF0h。
- b) 如果主机通过RS-485和一个驱动器相连, 两个设备必须有不同的轴ID值。例如如果主机轴ID等于255, 那么驱动器轴ID必须等于1, 信息与备注a)中相同

例子2-类型B信息: 主机通过RS-232和驱动器相连并且想从驱动器获得KPP参数的值(位置控制器的比例部分)。KPP在TML数据存储器中的地址是025Eh。主机和驱动器的ID是255=0FFh。主机发送一个“GiveMeData”请求并且驱动器应答一个“TakeData”信息。我们假设驱动器返回的KPP值是288(120h)。

备注: 用[命令解释器](#)获取TML数据地址。

一个“GiveMeData”请求信息为一条包含以下信息的TML数据:

“GiveMeData”请求为一个TML数据-信息描述

轴ID代码 (IDC) =目标轴
操作码: B004h为16位TML数据 B005h为32位TML数据
数据 (1): 发送轴ID代码

数据（2）：被请求的数据地址

“TakeData”应答信息包括以下信息：

“TakeData”应答-信息描述

轴ID代码（IDC）=目标轴
操作码：B404h为16位数据 B405h为32位数据
数据（1）：发送轴ID代码
数据（2）：被请求的数据地址
数据（3）：被请求的数据值16LB
数据（4）：被请求的数据值16MSB(为32位数据)

在这个例子的特定情况中，“GiveMeData”的轴ID代码与二进制代码是：

轴ID代码+“GiveMeData”请求把KPP值发送给轴ID=255的二进制代码

轴ID码（IDC）=0FF0h（目标轴）
操作码：B004h（为16位TML数据）
数据（1）=0FF1h(发送方轴ID代码=IDC+主机位设置)
数据（2）=025Eh（KPP地址）

“TakeData”的轴ID码和二进制代码是：

轴ID代码+“TakeData”（来自轴255的KPP值）的二进制代码

轴ID代码（IDC）=0FF1h（目标轴）
操作码：B404h（为16位数据）
数据（1）=0FF0h(发送轴ID码)
数据（2）=025Eh（KPP地址）
数据（3）=012h（KPP值）

主机必须发送一个包含以下内容的“GiveMeData”请求信息：

连串信息：“GiveMeData”请求把KPP值发送给轴255

字节1：08h-长度：IDC=2,操作码=2,数据=4
字节2：0Fh-轴ID码的高字节=0FF0h(目标轴)
字节3：F0h-轴ID码的低字节=0FF0h（目标轴）
字节4：B0h-操作码的高字节=B004h

字节5: 04h-操作码的低字节=B004h
字节6: 0Fh-数据 (1) 的高字节=0FF1h (发送轴ID代码)
字节7: F1h-数据 (1) 的低字节=0FF1h (发送轴ID代码)
字节8: 02h-数据 (2) 的高字节=025Eh (KPP地址)
字节9: 5Eh-数据 (2) 的低字节=025Eh (KPP地址)
字节10: 1Bh-校验和

驱动器会返回一个字节 **0X4F** 作为对信息接收 OK 的确认 (详情请看下面的 RS-232 和 RS-485 协议描述细节), 然后 “TakeData” 应答信息有以下内容:

连串信息: “TakeData” (KPP 值来自轴 255)

字节1: 0Ah-长度: IDC=2,操作码=2, 数据=6
字节2: 0Fh-轴ID代码的高字节=0FF1h(目标轴)
字节3: F1h-轴ID代码的低字节=0FF1h (目标轴)
字节4: B4h-操作码的高字节=B404h
字节5: 04h-操作码的低字节=B404h
字节6: 0Fh-数据 (1) 的高字节=0FF0h (发送轴ID代码)
字节7: F0h-数据 (1) 的低字节=0FF0h (发送轴ID代码)
字节8: 02h-数据 (2) 的高字节=025Eh (KPP地址)
字节9: 5Eh-数据 (2) 的低字节=025Eh (KPP地址)
字节10: 01h-数据 (3) 的高字节=0120h (KPP值)
字节11: 20h-数据 (3) 的低字节=0120h (KPP值)
字节12: 42h-校验和

备注:

a) 如果另一个带轴ID=1的驱动器通过CAN-bus与轴ID=255的驱动器相连接, 且主机想要从轴ID=1的驱动器获得KPP的值, 那么在“GiveMeData”中的轴ID代码应该变为0010h, 替代0FF0h。“TakeData”信息也会是0010h替代0FF0h作为发送轴ID代码。

b) 如果主机通过RS-485与一个驱动器相连, 这两个设备必须有不同的轴ID值。例如主机轴ID=255, 那么驱动器轴应该为轴ID=1, 相对于上述例子的改变是:

- “GiveMeData”: 轴ID代码-0010h替代0FF0h且发送轴ID代码-0FF0h替代0FF1h (主机位=0)
- “TakeData”: 轴ID代码-0FF0h替代0FF1h(主机位=0)且发送轴ID代码-0010h替代0FF0h

例子3-类型C信息: 主机通过RS-232连接一台驱动器, 当想要驱动器中已编程的运动序列在运行完成时通知主机。主机与驱动器的轴ID都是255=0FFh。类型C的信息是一个“TakeData2”信息被发送, 但无“GiveMeData2”的请求。它包含以下信息

“TakeData2”-信息描述

轴ID代码 (IDC) =主控制器MASTERID (目标轴)
操作码: D400h为16位数据+8位发送轴ID D500h为32位数据+8位发送轴ID
数据 (1) : 发送数据地址
数据 (2) : 发送数据值16LSB
数据 (3) : 发送数据值16MSB(为32位数据)

目标轴是由TML变量主控制MASTERID,依照公式: MASTERID=主机轴ID*16+1。在这个例子中,8位主机轴ID=255,因此MASTERID=16*255+1=4081 (0XFF1)。在C类信息中,“TakeData2”将返回:

- 2个状态寄存器SRL (位15-0) 与SRH(位31-16)的32位值, 如果所选择的其中一个位改变 (被请求的数据地址是SRL地址)
- 错误寄存器MER的16位值, 如果所选择的其中一个位改变
- PVT/PT状态PVTSTS的16位值, 如果PVT/PT缓冲状态改变
- 用TML命令SEND发送被请求的16位或32位TML数据

备注: 用命令解释器获得上述TML数据地址。SRL与SRH状态寄存器也可以当做一个单独的SR32的32位变量被访问。

位选择通过3个使能标志Masks完成,一个Masks对应一个寄存器,在TML参数中为:SRL_MASK,SRH_MASK,MER_MASK。一个位设置一个使能标志,当来自相应寄存器的同一个位改变时使能信息传送。在这个例子中,运动完成通过设置SRL.10=1发出信号。当SRL.10改变时为了激活“TakeData2”的自动发送,设置SRL_MASK=0X0400。

如果SRH=0X201且SRL=0X8400,在SRL.10从0变为1时,主机将获得一个“TakeData2”信息有以下内容:

连串信息“TakeData2”(来自轴255的状态寄存器SRL和SRH)

字节1: 0Ah-长度: IDC=2,操作码=2,数据=6
字节2: 0Fh-MASTERID的高字节=0FF1h(目标轴)
字节3: F1h-MASTERID的低字节=0FF1h(目标轴)
字节4: D5h-操作码的高字节=D4FFh
字节5: FFh-操作码的低字节=D4FFh
字节6: 09h-数据(1)的高字节=090Eh(SRL地址)
字节7: 0Eh-数据(1)的低字节=090Eh(SRL地址)
字节8: 84h-数据(2)的高字节=8400h(SRL值)
字节9: 00h-数据(2)的低字节=8400h(SRL值)
字节10: 02h-数据(3)的高字节=0201h(SRH值)
字节11: 01h-数据(3)的低字节=0201h(SRH值)

字节12: 7Ch-校验和

备注: 一个 SRL.10=1 的“TakeData2”信息发出时表示最后的运动任务已经完成。而一个 SRL10=0 的“TakeData2”信息发出时表示一个新的运动已经开始且可用于最后的运动命令确定。

RS-232 通信协议

RS-232 协议是全双工, 允许在两个方向上同时传送。在每个命令 (类型 A 或 B) 由主机发送后, 驱动器将通过发送一个应答-OK 字节确认是否接收。这个字节是 ‘O’ (ASCII 代码的大写字母的 “O”, 0x44F)。如果主机接收到 ‘O’ 字节, 这意味着驱动器已经正确地接收到了上一次发送的信息 (校验和的校验被通过), 且准备接收下一条信息。

备注: 如果接收信息的目标轴不是通过 RS-232 与主机直接连接的 (例如中继轴), 但是通过 CAN_bus 与中继轴相连接, 从中继轴接收到应答-OK 字节就不意味着信息已经被目标轴接收, 而是只被中继轴接收。依赖于 CAN-bus 波特率与该总线上的通信量, 主机在发送下一个信息给连接在 CAN-bus 上的任何一个轴前可能需要考虑引入一个延时。这个延时是必须提供给中继轴所必需的时间来通过 CAN-bus 转发信息。

如果在信息接收期间发生任何错误, 例如由驱动器轴所计算的校验和与由主机发送的校验和不匹配, 驱动器将不会发送应答-OK 字节。如果主机在校验和字节传送结束后的至少 2ms 之内还没有接收到任何应答字节, 这意味着在最后信息传送期间的某个点, 一个字节已被丢失且主机与中继轴之间的同步也已经消失了。为恢复这个同步, 主机应该进行以下工作:

- 1) 发送一个值为 0x0d 的同步 SYNC 字节 (更大的值也可以被接受)
- 2) 为一个应答等待一个可编程的超时 (典型值为 2ms) 周期
- 3) 如果驱动器发送回一个值为 0x0d 的同步 SYNC 字节, 那么同步被恢复且主机将重新发送上一个信息, 否则去到步骤 1)
- 4) 重复步骤 1)到 3)直到驱动器用一个同步 SYNC 应答或直到 15 个同步 SYNC 字节被发送。如果在发送 15 个同步 SYNC 字节后, 驱动器仍然没有应答, 那么通信问题是严重的且必须检查通信连接是否正确可靠

当一个主机通过 RS-232 发送了一个 A 类型的信息时, 它必须:

- a) 发送这个信息 (如在例子 1 中那样)
- b) 等待来自驱动器的应答-OK 字节 ‘O’

当一个主机通过 RS-232 发送了一个 B 类型的信息时, 它必须:

- a) 发送请求信息 (如在例 2 的 ‘GiveMeData’ 命令一样)
- b) 等待通过 RS-2332 连接的驱动器 (中继轴) 的应答-OK 字节 ‘O’
- c) 等待驱动器的应答信息 (如在例 2 的 ‘TakeData’ 应答一样)

当中继轴返回一个响应信息时, 它不是期望从主机接收一个应答字节, 而是主机监控通信的任务。如果主机得到一个带有错误的校验和的响应信息, 重新发送数据请求是主机的责任。

RS-485 通信协议

RS-485通信是多点，半双工，可以使您在一个网络中连接多达32个驱动器。在一个RS-485网络中，在某一时刻只允许一个设备发送数据，如果两个设备在同一时刻被错误地传送，那么两个传送都会被丢失。因此为了系统正确的运行，在一个RS-485网络中必须有一个主控制器控制传送。换句话说，只有当主控制器初始化一个传送时，主控制器请求网络中所有其他设备提供某些数据时，其他设备才可以传送。通常您应该设置您的主机作为主控制器。

在每个命令（类型 A 或 B）被主机发送到一个驱动器后，这个驱动器将通过发送一个应答-OK 字节确认是否接收。这个字节是：‘O’（ASCII 码的大写字母‘o’，0x4F）。如果主机接收到‘O’字节，意味着驱动器已经正确的接收了（校验和的验证被通过）上一个发送的信息，并且同时准备接收下一个信息。

当主机广播发送一个信息给一组驱动器时，驱动器将不发送应答-OK 字节。

如果在信息接收期间发生任何错误，例如由驱动器轴所计算的校验和与由主机发送的校验和不匹配，驱动器将不会发送应答-OK 字节。如果主机在校验和字节传送结束后的至少 2ms 之内还没有接收到任何应答字节，这意味着在最后信息传送期间的某个点，一个字节已被丢失且主机与中继轴之间的同步也已经消失了。为恢复这个同步，主机应该进行以下工作：

- 1) 发送 15 个值为 0x0d 的同步 SYNC 字节或其他大于 0xFF 的值
- 2) 等待一个可编程的超时周期（典型值为 2ms）
- 3) 重新发送上一次的命令且等待驱动器应答
- 4) 如果驱动器仍无应答，那么说明通信问题很严重并检查串行联结

当一个主机通过 RS-485 发送一个 A 类型的信息，它必须：

- a) 发送信息（如在例子 1 中那样）
- b) 等待来自驱动器的应答-OK 字节 ‘O’，仅限该信息的目标轴是否为单个驱动器

当一个主机通过 RS-485 发送一个 B 类型的信息，它必须：

- a) 发送请求信息（如在例 2 的 ‘GiveMeData’ 命令一样）
- b) 等待来自驱动器的应答-OK 字节 ‘O’
- c) 等待驱动器的应答信息（如在例 2 的 ‘TakeData’ 命令一样）

备注：

- 当使用 RS-485 协议时，不要发送类型 B 请求信息给一组轴，因为应答信息会发生重叠
- 当使用 RS-485 协议时，类型 C 信息被禁止。仅只有主机/主控制器被允许初始化传送后才可以

当驱动器返回一个应答信息，它不是期望从主机接收一个应答字节，而是主机监控通信的任务。如果主机得到一个带有错误的校验和的响应信息，重新发送数据请求是主机的责任。

第 4 章 CAN-bus通信TMLCAN协议

大多数的泰科智能伺服驱动器都可以通过 CAN-bus 通信。CAN-bus 通信为多点，半双工，而且在一个网络中您可以联结多达 32 个驱动器。

CAN-bus 的主要优点是它自动地解决冲突的能力。在 CAN-bus 网络中，如果两个设备同时开始发送，其中一个有较高优先级的设备将总是赢得网络的访问权且完成它的发送。另一个设备，在丢失网络访问权后，将从发送转变为接收，接收更高优先级的信息，然后再试图重新发送它本身所想要发送的信息。这个过程是通过硬件（CAN-bus 控制器）自动地完成，并且在更高的等级是透明的。换句话说，一个 CAN-bus 网络也能象全双工模式一样工作，确信发送冲突是否发生，且自动地解决这些冲突。

深圳市泰科智能伺服技术有限公司为扩展 CAN-bus 的优点，对公司的智能伺服驱动器进行了特殊的设计。例如，在多轴应用中您可以真正地分配主控制器与驱动器之间的控制能力，以替代一个轴运动的每一步都需要发送一个命令。您可以用 TML 编程运动控制以执行复杂的任务且在运动完成时通知主控制器，因此对于每一台驱动器轴，主控制器的任务可以被减少到：调用 TML 函数（如果需要也可能是终止执行）且等待一个信息，确认执行。如果必要，驱动器也可以被编程，定期地发送通知信息给主控制器，因此它也能监控一个任务的进程。

取决于客户所订购的产品型号，泰科智能驱动器既可以用 TMLCAN 协议交付，也可以用 CANopen 交付。根据要求，TMLCAN 协议，基于 CAN2.0B，也可以由基于 CAN2.0A 的 [TechnoCAN](#) 协议替代。TechnoCAN 被经过特殊设计允许连接一个不带 CANopen 的泰科智能驱动器在 CANopen 网络上，使用 CANopen 协议交换信息，TechnoCAN 与 CANopen 不会彼此打扰，因此可以在同一物理总线上能共同存在。

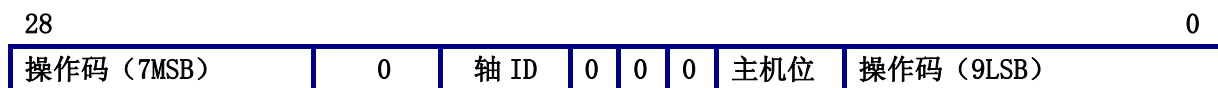
在 TMLCAN 协议中的信息打包

TMLCAN 基于 CAN2.0B 使用 29 位标识符。它可接受以下波特率：125kb，250kb，500kb（复位后默认值），800kb 与 1Mb。

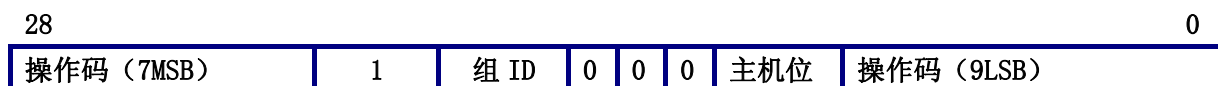
信息目标（一个轴或一组轴）且 TML 指令二进制代码被打包如下：

一个信息的 CAN 信息标识符被发送到：

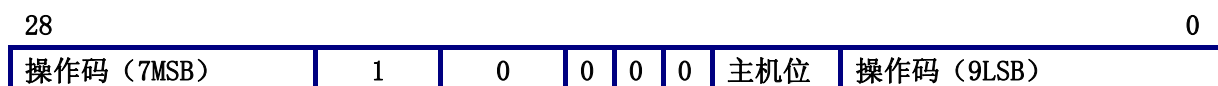
轴



组



广播



CAN信息数据字节:

CAN信息数据字节号	TML数据字
0	数据字 (1) -低字节
1	数据字 (1) -高字节
2	数据字 (2) -低字节
3	数据字 (2) -高字节
4	数据字 (3) -低字节
5	数据字 (3) -高字节
6	数据字 (4) -低字节
7	数据字 (4) -高字节

CAN信息结构

CAN-bus通信的信息类型

CAN-bus通信基于3种类型的TML命令压缩包的信息:

- 类型A: 信息不需要一个应答 (或一个返回信息)。这些信息既可以由主机发送, 也可以由其他驱动器发送, 包含的TML指令如: 执行参数设置, 运动编程, 电机命令等。
- 类型B: 信息需要一个应答。这些信息由主机发送且包含一个[在线TML指令](#)。这些命令请求返回数据, 例如TML参数值, 寄存器或变量的值。
- 类型C: 无需主机请求, 信息由驱动器发送给主机。当一个指定的条件发生时或跟着执行TML发送命令时, 这些信息都可以被发送 (详情请看[信息发送到主机](#)章节)

下一章给出了每种信息类型的一个例子。

例子1-类型A信息: 主机连接在CAN-bus上, 发送TML指令“KPP=0x1234”(用值0x1234设置位置控制器的比例部分)给轴ID=5的驱动器, TML指令的二进制代码是:

TML指令KPP=0x1234的二进制代码

操作码=205Eh
数据字 (1) =1234h (要在KPP中设置的数据)

备注: 用[二进制代码浏览器](#)获得TML指令的二进制代码

CAN信息标识符是:

CAN信息标识符: TML指令KPP=0x1234发送给轴5

28							0
操作码 (205Eh的 7MSB)	组 位	AAR (7...0)	0	0	0	主机 位	操作码 (205Eh的9LSB)
0 0 1 0 0 0 0	0	00000101	0	0	0	0	0 0 1 0 1 1 1 1 0

0400A05Eh

主机必须用以下内容发送一个CAN信息:

CAN信息: TML指令KPP=0x1234发送给轴5

	值	描述
标识符	0400A05E	CAN信息标识符
字节0	34	数据字 (1) 的低字节=1234h
字节1	12	数据字 (1) 的高字节=1234h

例子2-类型B信息: 主机想获得组1成员的2个驱动器的位置误差。主机轴ID为3且驱动器的轴ID是5和7。位置误差是16位TML变量叫做**POSERR**且它在TML数据存储器的地址是0x022A。主机为TML变量**POSERR**发送给组1一个“**GiveMeData2**”请求信息且等待“**TakeData2**”应答。

主机为POSERR请求的“**GiveMeData2**”的组ID代码与二进制代码是:

为**POERR**值发送给组1请求的“**GiveMeData2**”的二进制代码

轴ID码=B204h
数据 (1) =0030h
数据 (2) =022Ah

CAN信息标识符是:

CAN信息标识符: 为**POSERR**值发送给组1的**GiveMeData2**请求

28							0
操作码 (B204h的 7MSB)	组 位	AAR (15...8)	0	0	0	主机 位	操作码 (B004h的9LSB)
1 0 1 1 0 0 1	1	00000001	0	0	0	0	0 0 0 0 0 0 1 0 0

16602004h

主机必须用以下内容发送一个CAN信息:

CAN信息: 为**POSERR**值发送给组1的**GiveMeData2**请求

	值	描述
标识符	16602004	CAN信息标识符
字节0	30	数据字(1)的低字节=0030h
字节1	00	数据字(1)的高字节=0030h
字节2	2A	数据字(2)的低字节=022Ah
字节3	02	数据字(2)的高字节=022Ah

假设轴ID=5的驱动器返回一个位置误差POSERR=2, “TakeData2”应答的二进制代码是:

TakeData2 (来自轴5的POSERR值) 的二进制代码

轴ID码=D405h
数据(1) =022Ah
数据(2) =0002h

CAN信息标识符是:

CAN信息标识符: TakeData2 (来自轴5的POSERR值)

操作码 (D405h的7MSB)							组位	AAR (15..8)	0	0	0	主机位	操作码 (B004h的9LSB)								
1	1	0	1	0	1	0	0	00000011	0	0	0	0	0	0	0	0	0	0	1	0	1

1A806005h

主机获得带有以下内容的CAN信息:

CAN信息: TakeData2 (来自轴5的POSERR值)

	值	描述
标识符	1A806005	CAN信息标识符
字节0	2A	数据字(1)的低字节=022Ah
字节1	02	数据字(1)的高字节=022Ah
字节2	02	数据字(2)的低字节=0002h
字节3	00	数据字(2)的高字节=0002h

假设轴ID=7的驱动器返回一个位置误差POSERR=1, “TakeData2”应答的二进制代码是:

TakeData2 (来自轴7的POSERR值) 的二进制代码

操作码=D407h

数据 (1) =022Ah

数据 (2) =0001h

CAN信息标识符是:

CAN信息标识符: **TakeData2** (来自轴7的**POSERR**值)

28							0								
操作码 (D407h的 7MSB)	组 位	AAR (15...8)	0	0	0	主机 位	操作码 (B004h的9LSB)								
1 1 0 1 0 1 0	0	00000011	0	0	0	0	0	0	0	0	0	0	1	1	1

1A806007h

主机获得带有以下内容的一个CAN信息:

CAN信息: **TakeData2** (来自轴7的**POSERR**值)

	值	描述
标识符	1A806007	CAN信息标识符
字节0	2A	数据字 (1) 的低字节=022Ah
字节1	02	数据字 (1) 的高字节=022Ah
字节2	01	数据字 (2) 的低字节=0001h
字节3	00	数据字 (2) 的高字节=0001h

例子3-类型C信息: 主机通过CAN-bus被连接到一个驱动器上, 且想要当驱动器中已编程的运动完成时通知主机。主机的轴ID=255并且驱动器的轴ID=1。类型C信息是“TakeData2”信息被发送, 无一个“GiveMeData2”的信息请求。它包含以下信息:

“TakeData2”-信息描述

操作码: D400h为16位数据+8位发送轴ID
D500h为32位数据+8位发送轴ID

数据 (1): 发送数据地址

数据 (2): 发送数据值16LSB

数据 (3): 发送数据值16MSB(为32位数据)

目标轴是由TML变量主控制**MASTERID**, 依照公式: **MASTERID=主机轴ID*16+1**。在这个例子中, 8位主机轴ID=255, 因此**MASTERID=16*255+1=4081 (0XFF1)**。在C类信息中, “TakeData2”将返回:

- 2个状态寄存器**SRL** (位15-0) 与**SRH**(位31-16)的32位值, 如果所选择的其中一个位改变 (被请求的数据地址是**SRL**地址)
- 错误寄存器**MER**的16位值, 如果所选择的其中一个位改变
- PVT/PT状态**PVTSTS**的16位值, 如果PVT/PT缓冲状态改变
- 用TML命令**SEND**发送被请求的16位或32位TML数据

备注：用命令解释器获得上述TML数据地址。SRL与SRH状态寄存器也可以当做一个单独的SR32的32位变量被访问。

位选择通过3个使能标志Masks完成，一个Masks对应一个寄存器，在TML参数中为：SRL_MASK,SRH_MASK,MER_MASK。一个位设置一个使能标志，当来自相应寄存器的同一个位改变时使能信息传送。在这个例子中，运动完成通过设置SRL.10=1发出信号。当SRL.10改变时为了激活“TakeData2”的自动发送，设置SRL_MASK=0X0400。

假设轴ID=1的驱动器返回SRH=0X201且SRL=0X8400,在SRL.10从0变为1时，“TakeData2”信息的轴ID代码与二进制代码是：

TakeData2 (来自轴1的状态寄存器SRL和SRH) 的轴ID代码与二进制代码

操作码=D501h
数据(1)=090Eh
数据(2)=8400h
数据(3)=0201h

CAN信息标识符是：

CAN信息标识符：**TakeData2** (来自轴1的状态寄存器SRL和SRH)

28							0																		
操作码 (D501h的7MSB)							组位	AAR (15...8)			0	0	0	主机位	操作码 (B004h的9LSB)										
1	1	0	1	0	1	0	0	11111111			0	0	0	1	1	0	0	0	0	0	0	0	0	1	1A9FE301h

主机将获得带有以下内容的CAN信息：

CAN信息：**TakeData2** (来自轴1的状态寄存器SRL和SRH)

	值	描述
标识符	1A9FE301	CAN信息标识符
字节0	0E	数据字(1)的低字节=090Eh
字节1	09	数据字(1)的高字节=090Eh
字节2	00	数据字(2)的低字节=8400h
字节3	84	数据字(2)的高字节=8400h
字节4	01	数据字(2)的低字节=0201h
字节5	02	数据字(2)的高字节=0201h

备注：一个SRL.10=1的“TakeData2”信息发出时表示最后的运动任务已经完成。而一个SRL10=0的“TakeData2”信息发出时表示一个新的运动已经开始且可用于最后的运动命令确定。

第 5 章 CAN-bus通信TechnoCAN协议

泰科智能伺服驱动器的CAN-bus协议默认出厂设置为不带CANopen的TMLCAN协议。TechnoCAN是一个可以替代TMLCAN的协议，TechnoCAN经过特殊的设计允许不带CANopen的泰科智能伺服驱动器连接在CANopen网络中，使用CANopen协议交换信息。TechnoCAN与CANopen不会彼此干扰，因此可以在同一物理总线上共同存在。

根据客户的需求，不带CANopen的泰科智能伺服驱动器可以用带TechnoCAN协议交付给客户。工作在TMLCAN协议的驱动器与工作在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的刷新来完成：带有TechnoCAN的所有泰科智能伺服驱动器的固件号都以2开始，即固件代码为F2xxY，其中2xx是固件号，Y是固件版本。

TechnoCAN基于CAN2.0A用11位为标识符。它接收以下波特率：125kb, 250kb, 500kb（复位后默认值），800kb和1Mb。像TMLCAN一样，TechnoCAN提供通过串行RS-232连接PC到来自CANopen网络中任意泰科智能驱动器的可能性且通过它访问所有的泰科智能伺服驱动器。在这种情况下，同时连接到CANbus与RS-232的驱动器变为中继轴（详情请看[通讯协议-概述](#)）

在TechnoCAN中TML指令被分为8类：

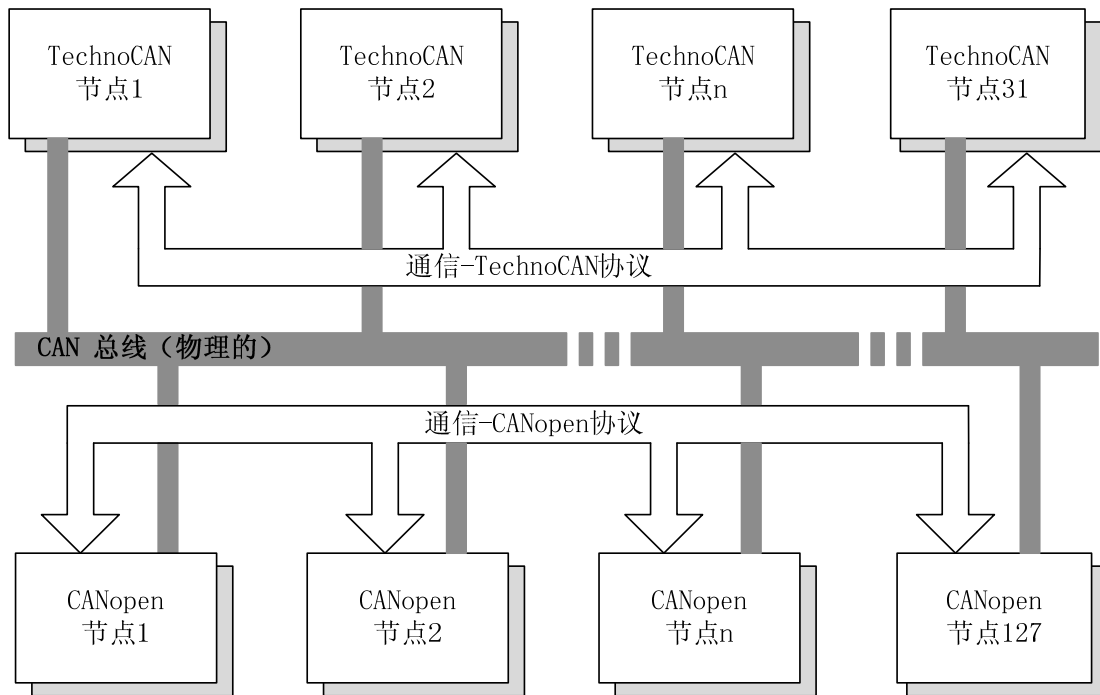
- a) **Normal**-包含寻址一个单轴驱动器的所有 TML 指令
- b) **Take Data**-包含对于请求的“GiveMeData”信息的“Take Data”应答
- c) **Group**-包含广播发送给一组驱动器的所有 TML 指令
- d) **Host**-包含除“Take Data”之外对所有[在线TML命令](#)的应答
- e) **PVT**-包含指令 PVTP(该指令作为一个 Normal 信息发送时太长)
- f) **Synchronisation**-包含为组 0 的同步信息
- g) **Broadcast**-包含除请求 GiveMeData 信息之外、寻址到组 0 的所有 TML 指令（对系统中的所有驱动器）
- h) **Take Data2**-包含对请求的“GiveMeData2”信息的“Take Data2”应答

每一类指令被映射到以下范围的 COB-ID(通信对象标识符-为一个 CAN 信息标识符的 CANopen 术语)：

COB-IDs 映射为 TechnoCAN 信息

描述	COB-ID 范围	
	十进制	十六进制
Group 信息	1-31	1-1F
Synchronization 信息	32	20
PVT 信息	65-95	41-5F
Take Data2 信息	257-287	101-11F
Normal 信息	289-319	121-13F
Host 信息	321-351	141-15F
Take Data 信息	353-383	161-17F
Broadcast 信息	512	200

TechnoCAN 仅使用由 CANopen 使用范围之外的 COB-IDs。因此，TechnoCAN 协议和 CANopen 协议可以在同一的物理总线上共同存在且同时通信而不会彼此打扰。



下表给出了如何在 TechnoCAN 与 CANopen 之间的 COB-IDs 的分配关系

CANopen 和 TechnoCAN COB-IDs

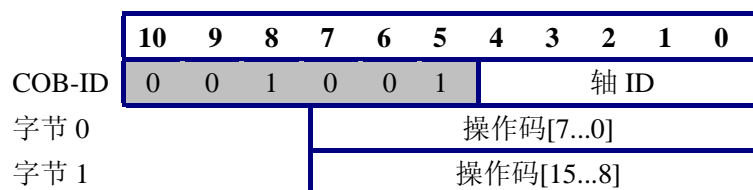
COB-ID 值	在 CANopen 中 COB-ID	在 TechnoCAN 中 COB-ID
0h	使用-NMT	不用
1 -1F	不用	使用-Group 信息
20	不用	使用- Synchronization 信息
21-40h	不用	不用
40-5Fh	不用	使用-PVT 信息
5F-7Fh	不用	不用
80h	使用- SYNC	不用
81- FFh	使用- EMERGENCY	不用
100h	使用- TIME STAMP	不用
101 -11Fh	不用	使用- Take Data2 信息
120h	不用	不用
121 -13Fh	不用	使用-Normal 信息
140h	不用	不用
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	使用-Broadcast
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

备注: 与TMLCAN 相比, TechnoCAN 有以下限制:

- 轴的最大数是 31: 轴 ID 值为: 1 到 31
- 组的最大数是 5: 组 ID 值为: 1 到 5

Normal 信息包: COB-ID:121h-13Fh



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]

Host 信息包: 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: Host 信息只有当驱动器或主控制器应答有 HOST 位设置为 1 的发送轴 ID 的一个数据请求时才会出现。这种情况例如: 当主机通过 RS-232 连接一个 PC 到一个驱动器且从另一个驱动器请求一个数据时, 驱动器的应答必须被发送给中继轴作为一个 Host 信息。当请求是由一个驱动器或由直接连在 CAN-bus 上的主机/主控制器发送时, 将不会出现 Host 信息。

Take Data 信息包: 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, Take Data TML 指令的 10 字节代码被压缩为 8 字节。通过以下方法完成:

- 从 16 位的操作码中, 只有低 10LSB 被发送。高 6MSB 总是为常数: 0x2D (101101b) 且不被发送。Take Data

信息的接收器必须在所接收的操作码的高 6MSB 增加 0x2D 以恢复整个 16 位代码为 Take Data 指令。

- HOST 位以字节 1 的位 2 被发送。不需要发送 GROUP 位，因为 GiveMeData 请求不能被发送给一组驱动器。
- TakeData TML 指令的第一个数据字是发送轴 ID 代码。因为驱动器的最大数目限制为 31，只有位 8-4 是有用的且被传输

Group 信息包: 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]					

PVT 信息包: COB-ID: 041h-05Fh

	10	9	8	7	6	5	4	3	2	1	0	
COB-ID	0	0	0	0	1	0	轴 ID					
字节 0							位置[7...0]					
字节 1							位置[15...8]					
字节 2							速度 [15...8]					
字节 3							位置 [23...16]					
字节 4							速度 [23...16]					
字节 5							速度 [31...24]					
字节 6							时间 [7...0]					
字节 7							计数器[6...0]					T[B]

备注: 在 PVT 信息中，PVT TML 指令的 10 字节代码被压缩为 8 字节。通过以下方法完成：

- 操作码不被发送。PVT 信息接收器在接收的操作码的 9MSB 与 7LSB 的计数器值上增加 0x6 以恢复完整的 16 位 PVT 指令代码。
- PVT 指令的第一个数据字包含 24 位位置值的 15LSB
- PVT 的第二个数据字包含 24 位速度值的 8LSB 与 24 位位置值的 8MSB
- PVT 指令的第三个数据字包含 24 位速度值的 16MSB
- PVT 指令的第四个数据字包含 9 位时间值

Synchronization 信息包: COB-ID: 020h

	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	0	0	0	0	1	0	0	0	0	0

备注:

- 该信息无数据字节
- 操作码为 0x1000
- 同步信息是广播信息；每一个被连接在网络中的驱动器都将接收到这个信息

Broadcast 信息包: COB-ID: 200h

	10	9	8	7	6	5	4	3	2	1	0
COB-ID	0	1	0	0	0	0	0	0	0	0	0
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]										

TakeData2 信息包: COB-ID: 101h-11Fh

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

备注:

- 一个驱动器将不会接收这个信息，这个信息主要是为其他的驱动器
- COB-ID 包含 Expeditor 轴 ID，为了主机一个接一个获得应答，主要优化 Expeditor 轴 ID 的升序
- VT 位指定了数据长度 (VT=0 为 16 位或 VT=1 为 32 位) 并且在发送第一个字节时被发送
- P 位指定如果信息是 TakeData2，给一个 GiveMeData2 信息的回复，或一个 PONG, 给一个 PING 信息的回复。PING 信息是一个广播信息，要求驱动器的轴 ID 与在网络中驱动器的固件版本。如果 P=0，则信息是 TakeData2。如果 P=1 则信息是 PONG (VT 位被自动复位且无意义)

例子 1: 连接在一个 CANopen 网络上的主机发送给轴 ID=5 的驱动器的 TML 指令 “**KPP=0x1234**” (设置位置控制器的比例部分值为 0x1234)。轴 ID 码和 TML 指令二进制代码是:

TML 指令 KPP=0x1234 的二进制代码

操作码=205Eh
数据字 (1) =1234h (在KPP中设置的数据)

备注: 用 [二进制代码浏览器](#) 获得 TML 指令的二进制代码。

主机发送一个 TechnoCAN 信息, 内容如下:

TechnoCAN 信息: TML 指令 KPP=0x1234 发送到轴 5

	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
字节 1 (20h)							0	0	1	0	0
字节 2 (34h)							0	0	1	1	0
字节 3 (12h)							0	0	0	1	0
字节 4							0	0	0	0	0
字节 5							0	0	0	0	0
字节 6							0	0	0	0	0
字节 7							0	0	0	0	0

备注: 最后 4 个字节无用且不被发送。

例子 2: 连接在一个 CANopen 网络上的主机想从轴 ID=5 的驱动器获得位置误差的值。主机轴 ID 是 3。位置误差是 16 位 TML 变量叫做 **POSERR** 且它在 TML 数据存储器中的地址是 0x022A。主机发送到轴 5 一个 “GiveMeData” 请求 TML 变量 **POSERR** 并且等待 “Take Data” 应答。

为 POSERR 的 “GiveMeData” 请求信息的轴 ID 代码与二进制代码是:

为 POSERR 值发送给轴 5 的 GiveMeData 请求信息的二进制代码

操作码=B004h
数据字 (1) =0030h
数据字 (1) =022Ah

主机必须发送一个带以下内容的 TechnoCAN 信息

TechnoCAN 信息: 为 POSERR 值发送给轴 5 的 GiveMeData 请求信息

	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	1	0	0
字节 1 (B0h)					1	0	1	1	0	0	0
字节 2 (30h)					0	0	1	1	0	0	0
字节 3 (00h)					0	0	0	0	0	0	0
字节 4 (2Ah)					0	0	1	0	1	0	1
字节 5 (02h)					0	0	0	0	0	0	1
字节 6					0	0	0	0	0	0	0
字节 7					0	0	0	0	0	0	0

备注: 最后 2 个字节无用且不被发送

假设轴 ID=5 的驱动器返回一个位置误差 POSERR=2, “TakeData” 应答的轴 ID 代码与二进制代码是:

TakeData 的二进制代码 (来自轴 5 的 POSERR 值)

操作码=D404h
数据字 (1) =022Ah
数据字 (1) =0002h

主机将得到一个带有以下内容的 TechnoCAN 信息

TechnoCAN 信息: TakeData (来自轴 5 的 POSERR 值)

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

备注: 最后 2 个字节无用且不被发送

例子 3: 一个 PVT 命令被发送给轴 ID=5 的驱动器如下 pvtp-1000L,-10, 500U, 0 (设置下一个点的位置坐标在: -1000IU=0.5 rot=FFFC18h, 速度在-10IU=300 rpm=FFF600, 时间 500IU=0.5s=01F4)

发送给轴 5 的 PVT 命令的二进制代码

操作码+计数器值=1800h+0h=1800h
数据 (1) =位置值[15...0]=FC18h
数据 (2) =位置值[23...16] 速度值[15...8]=FF 00h
数据 (3) =速度值[31...16]=FFF6h
数据 (4) =时间值[8...0]= F401h

被发送的 TechnoCAN 信息有以下内容:

TechnoCAN 信息: PVT 命令为轴 5

	10	9	8	7	6	5	4	3	2	1	0
COB-ID(轴 ID=5)	0	0	0	0	1	0	0	0	1	0	1
字节 0 (18h)				0	0	0	1	1	0	0	0
字节 1 (FCh)				1	1	1	1	1	1	0	0
字节 2 (00h)				0	0	0	0	0	0	0	0
字节 3 (FFh)				1	1	1	1	1	1	1	1
字节 4 (F6h)				1	1	1	1	0	1	1	0
字节 5 (FFh)				1	1	1	1	1	1	1	1
字节 6 (F4h)				1	1	1	1	1	1	1	1
字节 7 (01h)				0	0	0	0	0	0	0	1

例子 4: 如果一台泰科智能驱动器接收了 SETSYNC 20 TML 指令, 它将变为同步的主控制器且每隔 20ms 的时间发送一个同步信息与时间给连接在 CANbus 网络上的所有驱动器。

在某一时刻, 主控制器的时间值为 0x246C46F 且 TML 指令的代码如下:

发送给所有轴的 Set Master Time 命令的二进制代码

操作码=1401h
数据 (1) =时间值[15...0]=C46Fh
数据 (2) =时间值[31...16]=0246h

TechnoCAN 信息是:

- 当同步信息被每一个驱动器接收时, 所指定的时间变量将被保存。

TechnoCAN 信息: 为所有轴的同步 synchronization 命令

	10	9	8	7	6	5	4	3	2	1	0
COB-ID(组 ID=0)	0	0	0	0	0	1	0	0	0	0	0

备注: 最后 8 个字节无用且不被发送

- 为从轴设置主控制器时间的主控制器广播信息

TechnoCAN 信息: 给所有轴的 Set Master Time 命令

	10	9	8	7	6	5	4	3	2	1	0
COB-ID(组 ID=0)	0	1	0	0	0	0	0	0	0	0	0
字节 0 (01h)				0	0	0	0	0	0	0	1
字节 1 (14h)				0	0	0	1	0	1	0	0
字节 2 (6Fh)				0	1	1	0	1	1	1	1
字节 3 (C4h)				1	1	0	0	0	1	0	0
字节 4 (46h)				0	1	0	0	0	1	1	0
字节 5 (02h)				0	0	0	0	0	0	1	0

例子 5: 如果轴 2 发生了一个控制错误, 驱动器将发送一个错误寄存器 **MER** (0x0008) 的值的 TakeData2 指令为如下内容:

TakeData2 (来自轴 2 的 MER 寄存器值) 的二进制代码

操作码+Expeditor轴ID=D400h+2h=D402h
数据 (1)=数据请求的存储器地址[15...0]=08FCh
数据 (2)=数据请求[15...0]=0008h

备注: VT 位被设置为 0

TechnoCAN 信息发送有以下内容:

TechnoCAN 信息: 来自轴 2 的 TakeData2 命令

	10	9	8	7	6	5	4	3	2	1	0
COB-ID(Expeditor 轴 ID=2)	0	0	1	0	0	0	0	0	0	1	0
字节 0 (00h)				0	0	0	0	0	0	0	0
字节 1 (FCh)				1	1	1	1	1	1	1	0
字节 2 (08h)				0	0	0	0	1	0	0	0
字节 3 (40h)				0	1	0	0	0	0	0	0
字节 4 (00h)				0	0	0	0	0	0	0	0

备注: 最后 3 个字节无用且不被发送。