BEAR™ 开发和使用手册

Westwood Robotics® Corporation

v 0.3.3

目录

1 简介

1.1 关于本手册

- 数据类型用 “<>” 标明. 例如：<list> 是 Python 语言中的一种数据类型.
- 本手册中，BEAR 还可以被称为 “BEAR 电机”, “BEAR 驱动器” 或 “BEAR 模块”.
- “注意” 标签包含重要建议和说明，如果不正确遵守，可能会导致 BEAR 损坏或故障。

> **注意**
>
> 这是 “注意” 标签的例子。
> “注意” 标签包含重要建议和说明，如果不正确遵守，可能会导致 BEAR 损坏或故障。

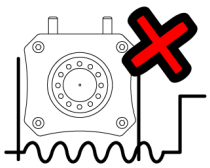
- “警告” 标签包含重要限制和说明，如果不正确遵守，肯定会对您的 BEAR 造成严重损坏，并可能导致危险的发生。

> **警告**
>
> 这是 “警告” 标签的例子。
> “警告” 标签包含重要限制和说明，如果不正确遵守，肯定会对您的 BEAR 造成严重损坏，并可能导致危险的发生。

1.2 警告

警告

为避免结构变形，请不要从侧面夹持 BEAR。



警告

为防止漏液带来的短路风险，请勿随意松动或紧固冷却盖板螺丝或拆卸冷却盖板。一经拆卸，需更换液冷密封圈。

警告

切忌将电源极性反接。

警告

为避免火灾所带来的危险，请在并联使用多个 BEAR 时估算每个 BEAR 的工作电流，并相应地选择合适的电源线，特别是预计会使用较高电流的情况。

警告

不要在驱动器使能时保存配置。驱动器在写入闪存时可能会暂时停止响应。

警告

当在 BEAR 仍处于使能状态下切换运行模式时请格外当心。BEAR 会在完成模式切换后立即执行新运行模式下相应的 goal_xxx 设定。

警告

直接力控模式下 BEAR 的实际输出速度不受设置的最高速度 limit_velocity_max 的限制。

1.3 初步了解 BEAR

a) **特征概述**

为了方便比较，表. 1和 2列出了所有当前 BEAR 产品的基本机械、电气和性能特性。

属性	Koala BEAR	Koala BEAR Muscle Build
当前版本	KB02	KBMB01
重量	250g	285g
输入电压	$9 \sim 33.6$V (3 ∼ 8S)	
电源接口	XT30	
信号接口	Molex PicoBlade 53047 6Pin	
等效转动惯量	$1.82 \times 10^{-3}\,\mathrm{kg/m^2}$	
速度常数 K_V	27.3 RPM/V	9 RPM/V
扭矩常数 K_T	0.35 Nm/A	1.16 Nm/A
峰值扭矩（15 秒）	3.5 Nm	8 Nm
峰值扭矩（15 秒，液冷）	4.2 Nm	N.A.[2]
峰值扭矩（1.5 秒)	10.5 Nm	20 Nm

表 1: BEAR Specification[1]

属性	Panda BEAR	Panda BEAR Plus	Kodiak BEAR
当前版本	PB02	PB02P	CB01
重量	685g	925g	2500g
输入电压	$9 \sim 50.4$V (3 ∼ 12S)		
电源接口	XT60		XT90
信号接口	Molex PicoBlade 53047 6Pin		
等效转动惯量	$7.44 \times 10^{-3}\,\mathrm{kg/m^2}$		
速度常数 K_V	14.3 RPM/V	7.1 RPM/V	
扭矩常数 K_T	0.67 Nm/A	1.3 Nm/A	
峰值扭矩（15 秒）	13.4 Nm	26.5 Nm	180 Nm
峰值扭矩（15 秒，液冷）	16.8 Nm	33 Nm	240 Nm
峰值扭矩（1.5 秒)	33.5 Nm	67 Nm	$\geq$ 350 Nm

表 2: Panda and Kodiak BEAR Specification[1]

b) Koala BEAR™ V2 (KB02)

Koala BEAR V2（产品代号 KB02）适用于各种小型移动机器人、机械手或小型机械臂等低负载高动态应用场景。下面结合图1对 KB02 的机械、电气以及液冷等特征进行介绍。

- 机械属性　KB02 的整体尺寸和外部安装孔尺寸与位置已在图1中详细标注。输出轴端面设有 6 个 3mm 直径销钉孔和 6 个 M3 螺纹孔用于定位和连接负载；正面设有 4 个 2.5mm 直径销钉孔和 8 个 M2.5 螺纹孔以及三个侧面均设有 4 个 2mm 销钉孔和 4 个 M3 螺纹孔用于定位和固定，此外，背面还设有关于输出轴中心对称的四个 M3 螺纹孔可用于固定或安装附加轴承。KB02 的机械性能详见表1。

[1]“峰值扭矩”是 BEAR 保持温升小于 55°C 的情况下可在给定的时间内持续输出的堵转扭矩；“液冷”代表使用了液态冷却功能。

[2]KBMB01 暂时没有液冷选项。

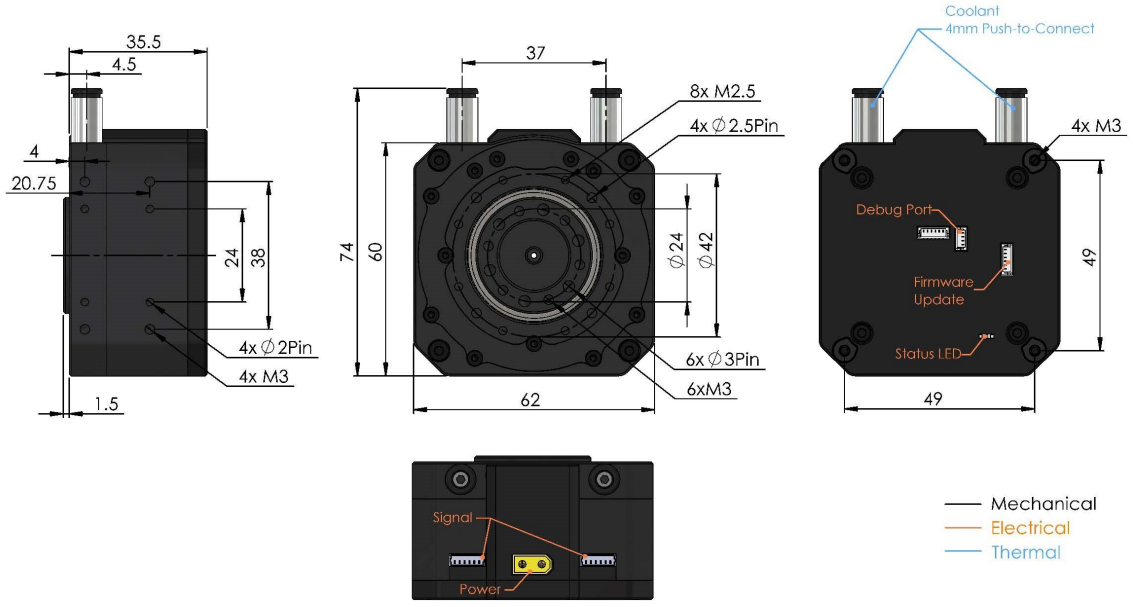
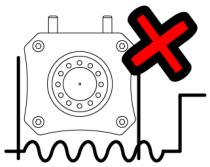


图 1: Koala BEAR V2 Specs

警告

为避免结构变形，请不要从侧面夹持 BEAR。



注意

当负载将对 BEAR 模块所驱动的关节施加较大轴向载荷时，需为负载额外添加轴向支撑，而不可使用 Koala BEAR 直接为负载提供轴向支撑。

- 电气属性　KB02 有一个电源接口，接头型号为 XT30。为方便多个 BEAR 模块的串联，KB02 配有两个信号接口，接头型号为 Molex PicoBlade 53047 6Pin。KB02 可接受的输入电压范围为：9 至 33.6V。

注意

当驱动高动态特性负载时，BEAR 可能在被驱动/冲击的情况下产生一个较大的反向电动势。此种应用场景建议使用 Li-Po 电池供电，或在动力电路中加入带有较大电容的反电动势吸收电路以进行缓冲从而保护供电单元。

- 液态冷却　KB02 的液冷接口为 M5 螺纹孔，您可以任意选择可匹配的各种形状、型号与材料的冷却管接头。冷却涵道可接受的最高液压为 1MPa。请使用蒸馏水作为冷却剂，可适当添加抑菌剂，不建议添加其他添加剂或染料。

注意

避免使用硫酸铜（CuSO4）添加剂（常见商品名“Nuke Cu”或“Biocide Cu”）。硫酸铜倾向于与液冷回路中常见的金属发生反应，尤其是散热水排（含锌、铜、锡）以及 BEAR (含铝) 从而促进腐蚀。另外，使用硫酸铜还会加速有碍观瞻的铜变色现象。

注意

为防止冷却剂漏液而导致短路，每次拆装冷却接头时请在液冷接头与壳体连接的螺纹处使用 PTFE 密封胶带进行密封，并在使用前进行密封确认。

注意

对于长期使用液冷的应用，请每周定期检查冷却系统密封性，谨防漏液。

警告

为防止漏液带来的短路风险，请勿随意松动或紧固冷却盖板螺丝或拆卸冷却盖板。一经拆卸，需更换液冷密封圈。

c) Koala BEAR™ Muscle Build V1 (KBMB01)

Koala BEAR Muscle Build V1（KBMB01）是 Koala 系列扭矩密度最高的执行器。它保持了 Koala 系列的紧凑和超轻设计，但提供高达标准 Koala BEAR V2 模块的 3 倍的扭矩。它非常适合对尺寸和重量非常严格要求的动态应用，同时更偏向于高负载能力而非高速度，比如可穿戴机器人设备、康复机器人和设备、机器人手或台式机器人。KBMB01 不支持液态冷却。

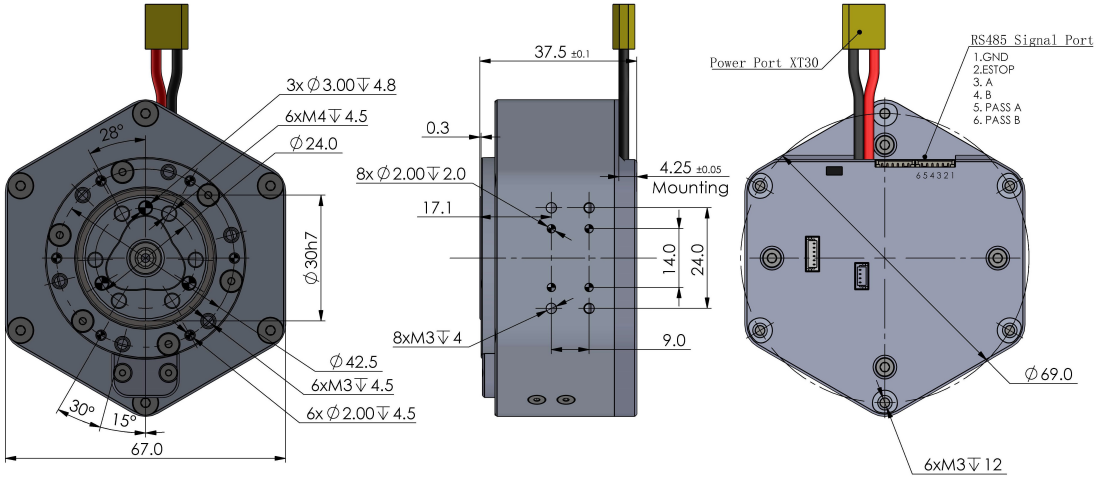
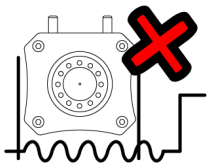


图 2: Koala BEAR Muscle Build V1 Specs

- 机械属性　KBMB01 的整体尺寸和外部安装孔尺寸与位置已在图2中详细标注。输出轴端面设有 3 个 3mm 直径销钉孔和 6 个 M4 螺纹孔用于定位和连接负载；正面设有 6 个 2mm 直径销钉孔和 6 个 M3 螺纹孔以及两个侧面对称地设有 4 个 2mm 销钉孔和 4 个 M3 螺纹孔用于定位和固定，此外，背面还设有关于输出轴中心对称的六个 M3 螺纹孔可用于固定或安装附加轴承。KBMB01 的机械性能详见表1。

警告

为避免结构变形，请不要从侧面夹持 BEAR。



注意

当负载将对 BEAR 模块所驱动的关节施加较大轴向载荷时，需为负载额外添加轴向支撑，而不可使用 Koala BEAR 直接为负载提供轴向支撑。

- 电气属性　KBMB01 有一个电源接口，接头型号为 XT30。为方便多个 BEAR 模块的串联，KBMB01 配有两个信号接口，接头型号为 Molex PicoBlade 53047 6Pin。KB02 可接受的输入电压范围为：9 至 33.6V。

注意

当驱动高动态特性负载时，BEAR 可能在被驱动/冲击的情况下产生一个较大的反向电动势。此种应用场景建议使用 Li-Po 电池供电，或在动力电路中加入带有较大电容的反电动势吸收电路以进行缓冲从而保护供电单元。

- 液态冷却　KBMB01 不支持液态冷却。

d) Panda BEAR™ V2 (PB02)/Panda BEAR™ V2 Plus (PB02P)

具备扭矩、自重和体积之间完美平衡的全能驱动器，Panda BEAR V2（产品代号 PB02）是 BEAR 系列中的全能选手，可以轻松胜任从轻型机械臂到足式机器人平台等各种应用场合。Panda BEAR V2 Plus（产品代号 PB02P）是 PB02 的升级高功版，其机械、电气以及液冷等特征与 PB02 相同。下面结合图3对 PB02 的机械、电气以及液冷等特征进行介绍。

- 机械属性　PB02 的整体尺寸和外部安装孔尺寸与位置已在图3中详细标注。输出轴端面设有 8 个 3mm 直径销钉孔和 8 个 M3 螺纹孔用于定位和连接负载；正面设有 8 个 3mm 直径销钉孔和 8 个 M3 螺纹孔以及三个侧面均设有 6 个 2mm 销钉孔和

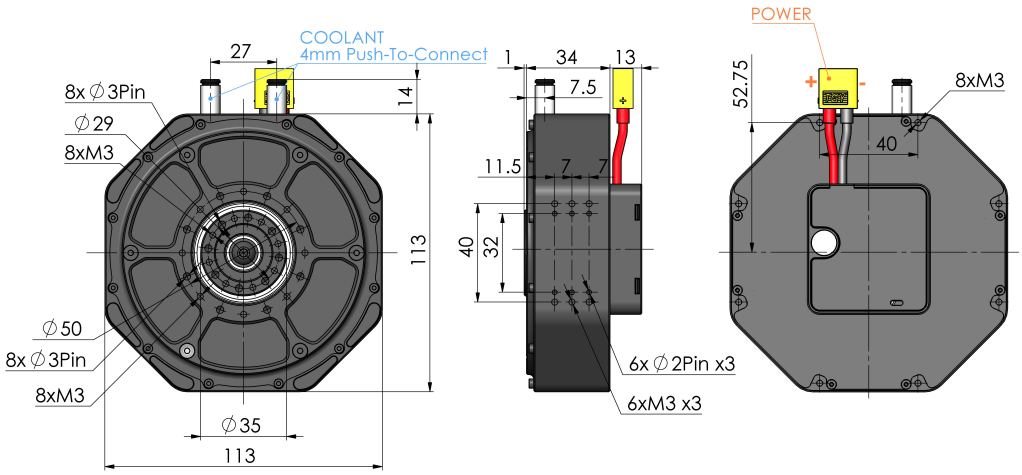
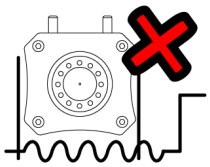


图 3: Panda BEAR V2 Specs

6 个 M3 螺纹孔用于定位和固定，此外，背面还设有关于输出轴中心对称的 8 个 M3 螺纹孔可用于固定或安装附加轴承。PB02 以及 PB02P 的机械性能详见表??。

警告

为避免结构变形，请不要从侧面夹持 BEAR。



注意

当负载将对 BEAR 模块所驱动的关节施加较大轴向载荷时，需为负载额外添加轴向支撑，而不可使用 Panda BEAR 直接为负载提供轴向支撑。

- 电气属性　PB02 有一个电源接口，接头型号为 XT60。为方便多个 BEAR 模块的串联，PB02 配有两个信号接口，接头型号为 Molex PicoBlade 53047 6Pin。请参照图4所示撬开背盖来连接线号接口。PB02 可接受的输入电压范围为：9 至 50.4V。

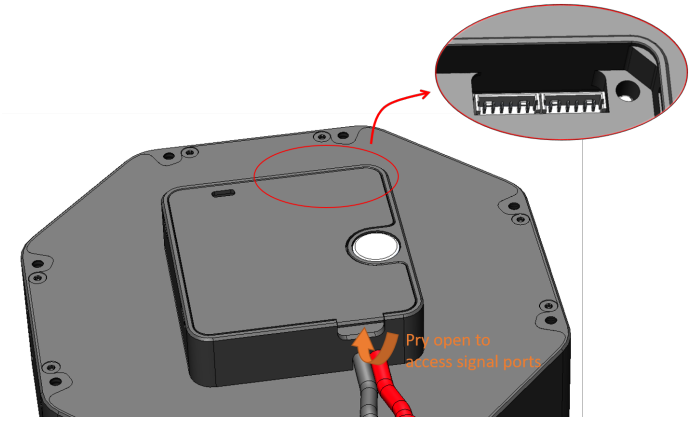


图 4: Panda BEAR Signal Ports

注意

当驱动高动态特性负载时，BEAR 可能在被驱动/冲击的情况下产生一个较大的反向电动势。此种应用场景建议使用 Li-Po 电池供电，或在动力电路中加入带有较大电容的反电动势吸收电路以进行缓冲从而保护供电单元。

- 液态冷却　PB02 的液冷接口为 M5 螺纹孔，您可以任意选择可匹配的各种形状、型号与材料的冷却管接头。冷却涵道可接受的最高液压为 1MPa，但如果您的液冷系统处于长期加压状态下，则建议您将压力控制在 0.7MPa 以避免冷却剂缓慢渗出的可能性。请使用蒸馏水作为冷却剂，可适当添加抑菌剂，不建议添加其他添加剂或染料。

注意

避免使用硫酸铜（CuSO4）添加剂（常见商品名“Nuke Cu”或“Biocide Cu”）。硫酸铜倾向于与液冷回路中常见的金属发生反应，尤其是散热水排（含锌、铜、锡）以及 BEAR (含铝) 从而促进腐蚀。另外，使用硫酸铜还会加速有碍观瞻的铜变色现象。

注意

为防止冷却剂漏液而导致短路，每次拆装冷却接头时请在液冷接头与壳体连接的螺纹处使用 PTFE 密封胶带进行密封，并在使用前进行密封确认。

注意

对于长期使用液冷的应用，请每周定期检查冷却系统密封性，谨防漏液。

警告

为防止漏液带来的短路风险，请勿随意松动或紧固冷却盖板螺丝或拆卸冷却盖板。一经拆卸，需更换液冷密封圈。

- Kodiak BEAR™ V1 (CB01)
  Kodiak BEAR V1（CB01）是我们专门为大型应用打造的最强产品线，提供非常强大的扭矩输出能力，适用于从人形机器人到工业机器人的各种应用，同时保持灵敏的扭矩感知和控制能力。

  CB01 的 CAD 模型、机械、电气和热管理特性对于现有的 Kodiak BEAR 驱动器客完全开放。一旦您的购买完成，请联系我们以获取这些文件。

2 使用 BEAR

2.1 电源与信号

a) **电源**

Koala 系列驱动器的电源端口为 XT30 公头，Panda 系列驱动器的电源端口为 XT60 公头，Kodiak 系列驱动器的电源端口为 XT90 公头。电源端口的极性遵循 XT30/XT60/XT90 连接器的常规设计：靠近平面的针脚为正极，靠近圆形/倒角侧的针脚为负极，如图. 5所示。

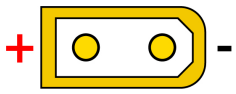


图 5: BEAR 电源接口与极性

> **警告**
> 切忌将电源极性反接。

不同 BEAR 的电源电压如表. 3所示。同时使用多个 BEAR 时，请仅将使用相同电压的 BEAR 并联。

> **警告**
> 为避免火灾所带来的危险，请在并联使用多个 BEAR 时估算每个 BEAR 的工作电流，并相应地选择合适的电源线，特别是预计会使用较高电流的情况。

产品系列	Koala	Panda	Kodiak
供电电压	9 ∼ 33.6V (3 ∼ 8S)	9 ∼ 50.4V (3 ∼ 12S)	9 ∼ 50.4V (3 ∼ 12S)
电源接口	XT30	XT60	XT90

表 3: BEAR 电源标准

b) **指示灯**

每一个 BEAR 都在其背面配有一个 LED 指示灯。该指示灯由绿色, 蓝色和红色三个不同颜色的 LED 灯珠组成，每个颜色代表着 BEAR 驱动器的不同状态：

绿色 指示灯在 BEAR 上电并完成初始化后保持常亮；The 蓝色指示灯在 BEAR 驱动器使能时亮起并在去使能后关闭；红色指示灯代表 BEAR 报错。

c) USB2BEAR **通讯接头**

我们推荐使用 Westwood Robotics 提供的 USB2BEAR™ 超高速 RS485-USB 通讯接头与 BEAR 进行通讯。

> **注意**
>
> 使用其他通用 RS485 接头装置时，风险自负。

USB2BEAR 接头为 USB2.0 设备，其 RS485 接口是 4 针 Molex Mini-SPOX 5268 公头（配合母头胶壳：4 针 Molex Mini-SPOX 5264，零件号：0050375043）它的接口引脚标注以及开关功能如图 6所示。

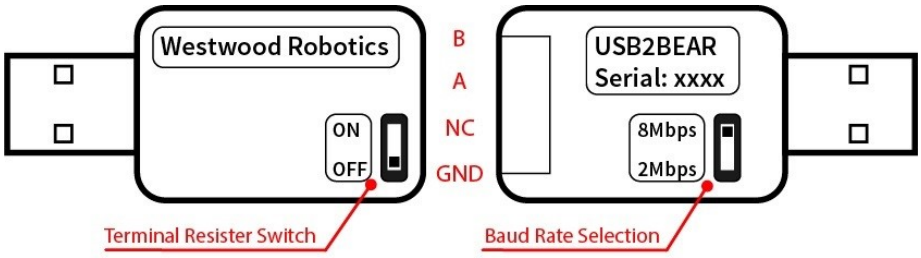


图 6: USB2BEAR 接口引脚标注以及开关

当使用较长信号线或较多 BEAR 驱动器导致信号阻抗较高从而通信嘈杂时，建议在 RS485 链的末端增加一个 120 欧姆的终端电阻同时将 USB2BEAR 的终端电阻开关置于“ON”，但在没有使用终端电阻时，请确保将该开关置于“OFF”。USB2BEAR 还可用于与 BEAR 以外的通用 RS485 设备通信，但请确保选择了匹配的通信波特率。与 BEAR 配对时，建议使用 8Mbps 进行高通信。

d) BEAR **通讯接口的链接**

当您的系统中有多个 BEAR 时，恰当的链接方式可以使您的线缆管理变得简单和整洁，而且还有助于提高系统的可靠性和鲁棒性。就像所有其他 RS485 设备一样，您可以简单地将 BEAR 进行菊链串接，如图 7 所示。当信号线相对较长或观察到异常信号噪声时，建议在信号链的末端添加一个 120 欧姆的终端电阻。

机器人系统中也可能需要有多个 BEAR 链，例如在双臂机器人或足式机器人的应用中。在这种情况下，传统解决方案要么使用一根长信号线将一条菊链的末端与另一条菊链的

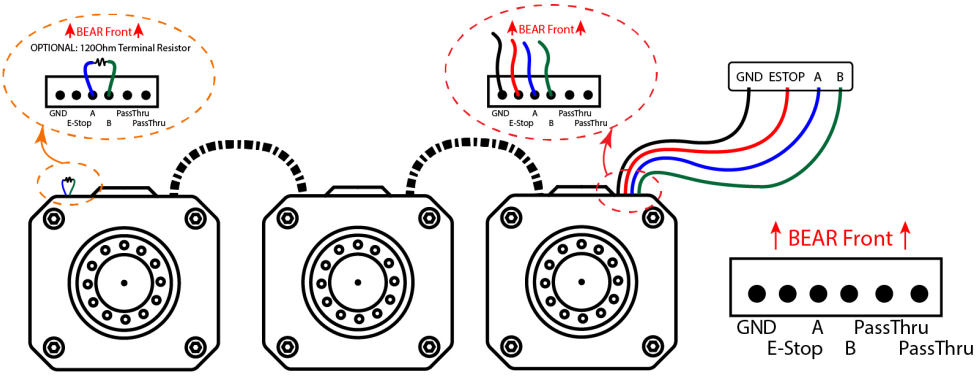


图 7: BEAR 的菊链串接

起点连接起来，要么使用多个 RS485 接头，每个菊链一个。前一种方案可能会导致信号线中的阻抗升高从而导致信号噪音大幅增加，而后者可能会加重控制复杂度并要求上位机控制器具备足够多的 USB 端口。

在这种情况下，可使用 BEAR 信号接口所特有的中继通道来大幅降低信号链的复杂度。图. 8展示了一个使用中继通道实现分叉链的简单示例。与之前相同，当信号线相对较长或观察到异常信号噪声时，建议在信号链的末端添加一个 120 欧姆的终端电阻。

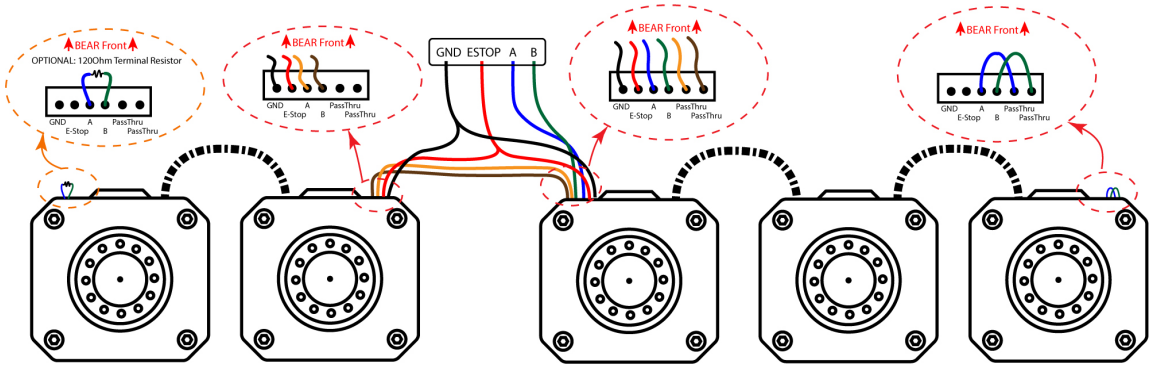


图 8: BEAR 的分叉链接

您可以将不同型号的 BEAR 链接在一起，但要注意确保所有 BEAR 都获得正确的电源电压；同一链中的所有 BEAR 应具有相同的波特率设置，并且不存在 ID 冲突。

e) **与** USB2BEAR **连接**

将 BEAR 的信号线与 USB2BEAR 连接的方式总共有 3 种，如图 9所示。在这三种连接方式中，GND、A、B 端子始终连接到 USB2BEAR 上的相应引脚，他们的唯一区别在于 ESTOP 信号端子的处理方式。最简化的连接方式是将 ESTOP 端子连接到信号 GND，但同时去除了急停保护的功能，因而不推荐使用；基本连接方式是在 ESTOP 和 GND 端子之间添加一个 E-STOP 急停开关，紧急情况下，通过断开 E-STOP 急停开关将 ESTOP 与 GND 断开，则可触发该信号链上所有 BEAR 的 E-STOP 保护机制。

我们推荐的配置是将 GND 和 ESTOP 端子连接到 Westwood Robotics Wireless ESTOP 模块的相应端子，从而使得用户能够在紧急状况下远程触发该信号链上的所有 BEAR 上的 E-STOP 保护机。在大功率或高度复杂的系统以及任何围绕人类工作的系统中这一点尤为重要。

有关急停保护的详细说明，请参阅 f)小节。

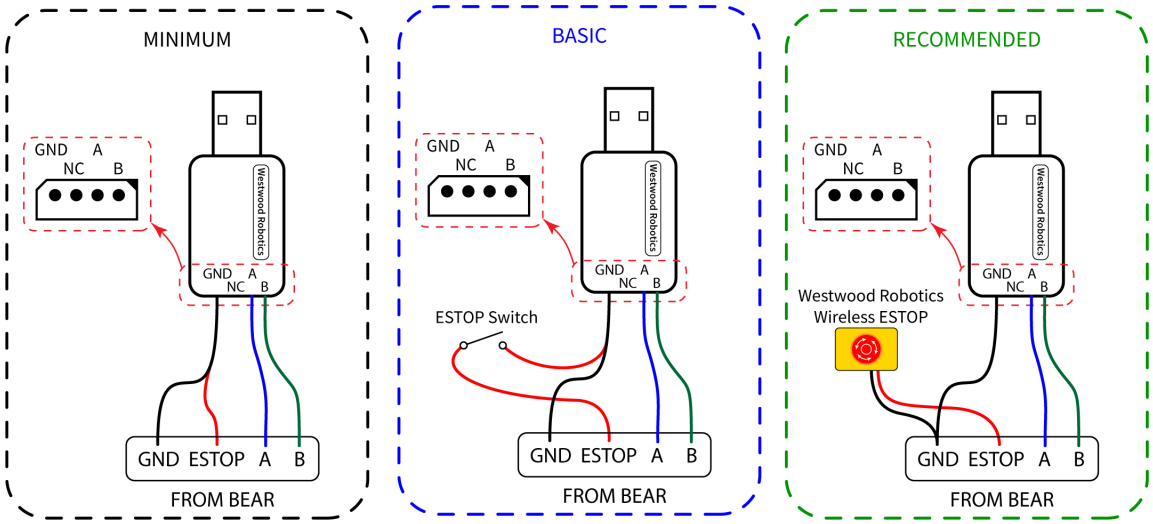


图 9: BEAR 与 USB2BEAR 的连接方法

注意

切忌让 ESTOP 引脚悬空，这将使 BEAR 保持在 ESTOP 状态并阻止其使能。

f) E-STOP

当 BEAR 上的 ESTOP 端子电平未被拉低至 GND 时，将触发 BEAR 模块的 E-STOP

保护机制。其错误代码的第 3 位将变为 HIGH，扭矩使能状态变为 3。如果在 BEAR 使能时触发 E-STOP 保护，则该 BEAR 将进入内部阻尼模式，防止任何潜在的进一步硬件损坏；如果在 BEAR 未使能时触发 E-STOP 保护，则 BEAR 将保持去使能状态，不会进入内部阻尼模式。

E-STOP 保护机制也可以通过向 BEAR 的使能状态寄存器写入 “3” 来触发，此时不必将 ESTOP 与 GND 之间的连接断开。

要解除 BEAR 的 E-STOP 保护，首先确保 ESTOP 电平被拉低至 GND，然后通过向其使能状态寄存器写入 “0” 来对 BEAR 去使能，即可解除 E-STOP 保护。这也会将 BEAR 的错误代码的第 3 位重置为 LOW。

有关扭矩使能状态寄存器的详细信息，请参阅2.2.2，有关错误代码的详细信息，请参阅2.2.3。

2.2 通讯

与 BEAR 的通信是通过使用 BEAR SDK 与 BEAR 的控制列表交互来实现的。控制列表是一种由多个寄存器组成的结构，用于存储状态或控制设备。用户可以通过读取控制列表中的特定寄存器来检查设备的当前状态，或者通过向某些寄存器写入特定数据来控制设备。

本节将对控制列表进行详细的解释说明。使用各种语言的 BEAR SDK 与控制列表进行交互的完整说明请参考第3章。

2.2.1 控制列表

控制列表中的所有寄存器分为两组：配置寄存器 (CONFIG) 和状态寄存器 (STAT)。当 BEAR 接收到 “save_config” 命令时，配置寄存器中的所有当前数值都将保存在闪存中。状态寄存器中的值和所有未保存的配置寄存器数值在断电时都将丢失，并在开机时复位至默认值或最后保存值。

警告

不要在驱动器使能时保存配置。驱动器在写入闪存时可能会暂时停止响应。

注意

建议仅在必要时保存配置。内部闪存仅可保证约一万次写入。

请参阅表. 4以获取完整的配置寄存器（CONFIG）和状态寄存器（STAT）列表。

配置寄存器							
名称	描述	访问权限	数据类型	默认值	最小值	最大值	
id	驱动器 ID	R/W	uint32		1	0	0xFC
mode	运行模式	R/W	uint32				
baudrate	通讯波特率	R/W	uint32	Mbps			
homing_offset	零位偏置	R/W	float32	rad			
p_gain_id	励磁电流控制环增益 P	R/W	float32		0.001	0	10
i_gain_id	励磁电流控制环增益 I	R/W	float32		0.0001	0	10
d_gain_id	励磁电流控制环增益 D	R/W	float32		0	0	10
p_gain_iq	扭矩电流控制环增益 P	R/W	float32		0.001	0	10
i_gain_iq	扭矩电流控制环增益 I	R/W	float32		0.0001	0	10
d_gain_iq	扭矩电流控制环增益 D	R/W	float32		0	0	10
p_gain_velocity	速度控制环增益 P	R/W	float32		0.2	0	1000
i_gain_velocity	速度控制环增益 I	R/W	float32		0.001	0	1000
d_gain_velocity	速度控制环增益 D	R/W	float32		0	0	1000
p_gain_position	位置控制环增益 P	R/W	float32		0.01	0	1000
i_gain_position	位置控制环增益 I	R/W	float32		2E-05	0	1000
d_gain_position	位置控制环增益 D	R/W	float32		0	0	1000
p_gain_direct_force	直接力控增益 P	R/W	float32		0	0	1000
i_gain_direct_force	直接力控增益 I	R/W	float32		0	0	1000
d_gain_direct_force	直接力控增益 D	R/W	float32		0	0	1000
limit_acc_max	最大角加速度限制	R/W	float32	rad/s^2	5	0	100000
limit_i_max	最大相间电流限制	R/W	float32	A	5	0	100
limit_velocity_max	最大转限制	R/W	float32	rad/s	100	0	10000
limit_position_min	位置下限	R/W	float32	rad	-8π	-8π	8π
limit_position_max	位置上限	R/W	float32	rad	8π	-8π	8π
min_voltage	最小输入电压	R/W	float32	V	6	6	60
max_voltage	最大输入电压	R/W	float32	V	40	6	60
watchdog_timeout	看门狗超时	R/W	uint32	μs	0	0	10000000
temp_limit_low	过温下限（限制输出）	R/W	float32	°C	80	0	125
temp_limit_high	过温上限（切断输出）	R/W	float32	°C	100	0	125
状态寄存器							
名称	描述	访问权限	数据类型	默认值	最小值	最大值	
torque_enable	输出使能	R/W	uint32		0	0	3
goal_id	目标励磁电流	R/W	float32	A			
goal_iq	目标力矩电流	R/W	float32	A			

goal_velocity	目标转	R/W	float32	rad/s			
goal_position	目标位置	R/W	float32	rad		-8π	8π
present_id	当前励磁电流	RO	float32	A			
present_iq	当前扭矩电流	RO	float32	A			
present_velocity	当前转速	RO	float32	rad/s			
present_position	当前位置	RO	float32	rad		-8π	8π
input_voltage	当前输入电压	RO	float32	V			
winding_temperature	线圈温度 °C	RO	float32				
powerstage_temperature	供电模块温度 °C	RO	float32				
ic_temperature	微控器温度 °C	RO	float32				
error_status	* 尚未启用	RO	float32				

*R/W: 可读可写 RO: 只读

表 4: Table of Registers

2.2.2 详细描述

配置寄存器

- id BEAR 的 ID：同一个信号链中的 BEAR 应被提前分配一个独特的 ID 来避免冲突。
- mode 运行模式：详细信息请参考 2.3小节。
- baudrate RS-485 通讯的波特率：除非特殊需要，建议保留默认的 8Mbps。
- homing_offset 零位偏置：当前位置 = 原始位置 + 零位偏置。当计算新的零位偏置时，请注意将已有零位偏置考虑进去。
- PID Gains PID 环路增益：详细信息请参考 2.4小节。
- limit_acc_max 最大角加速度限制（单位: rad/s^2）：在运行模式 2（位置模式）下限制驱动器的最大角加速度。
- limit_i_max 最大相间电流限制：扭矩电流 Iq 和励磁电流 Id 的最大限制（单位：A）：在所有模式下有效，因为输出扭矩与 Iq 成正比，此限制即最大输出扭矩限制。
- limit_velocity_max 最大角速度限制（单位：rad/s）：在模式 1（速度模式）和模式 2（位置模式）下有效。
- limit_position_min/max BEAR 的输出位置上/下限（单位：rad）：当 BEAR 的实际输出位置超出了设置的输出位置上/下限时，则触发内部阻尼模式并报错。去使能，并将 BEAR 输出轴转回限位范围内或重新调整限位范围即可清除错误代码。在模式 2（位置模式）和模式 3（直接力控模式）下有效。
- min/max_voltage 最小/大输入电压：当输入电压超出该范围则会报错。去使能，并将输入电压限制在所设置的范围内或调整电压限制设置即可清除错误代码。
- watchdog_timeout 看门狗超时（单位：μs）：当 BEAR 使能且通讯超时，BEAR 进入内部阻尼模式并报错以保障安全。**仅**在模式 0（力矩模式）下有效。看门狗超时前与 BEAR 进行一次通讯则复位看门狗。
- temp_limit_low 过温下限（单位：°C）：当 BEAR 中的任何关键模块温度超过此限制但还未达到过温上限，BEAR 将自动逐步线性降低最大相间电流限制，同时生成错误代码。

- temp_limit_high 过温上限（单位：°C）：当 BEAR 中的任何关键模块温度超过此限制，BEAR 将自动降低最大相间电流限制为 0，其现象等同于将 BEAR 去使能。

状态寄存器

- torque_enable BEAR 的输出使能状态与控制
  写入时：
  0 - 去使能并清除已修正的错误代码；
  1 - 输出使能；
  3 - 触发 EStop 保护机制，如果触发前 BEAR 为使能状态则进入内部安全阻尼模式。
  读取时：
  0 - BEAR 去使能；
  1 - BEAR 已使能；
  2 - BEAR 去使能且存在关键错误而无法使能；
  3 - BEAR 因为非关键错误而进入内部安全阻尼模式。
- goal_id 目标励磁电流 Id（单位：A）：正常情况下保持为 0。
- goal_iq 目标扭矩电流 Iq（单位：A）：当 BEAR 处于模式 0（力矩模式）和模式 3（直接力控模式）时可写入，Iq 与输出扭矩成正比。
- goal_velocity 目标输出转速（单位：rad/s）：仅当 BEAR 处于模式 1（速度模式）时可写入。
- goal_position 目标位置（单位：rad）：当 BEAR 处于模式 2（位置模式）和模式 3（直接力控模式）时可写入。
- present_id/iq/velocity/position BEAR 的当前电流、速度以及位置状态：只读。
- input_voltage 当前输入电压（单位：V）：只读。
- winding_temperature 线圈温度（单位：°C）。
- powerstage_temperature 供电模块温度（单位：°C）。
- ic_temperature 微控器温度（单位：°C）。

2.2.3 **错误代码**

BEAR 将于每次返回的数据中包含 8 位错误代码来返回其当前的错误状态。错误代码的最高位始终为 1，之后每一位的详细解释见表. 7。

位	类型	名称	备注
0	警告	通讯异常	
1	警告	过热	
2	错误	绝对位置异常	
3	错误	看门狗超时或 E-STOP	
4	错误	超限位	
5	错误	硬件错误	
6	错误	初始化错误	
7	1		始终为 1

表 5: 错误代码（小端字节序）

详细描述

- **通讯异常**收到损坏的数据包。此警告会自动重置，并且仅与相对应的通信贞相关联。
- **过热**该 BEAR 中的微控器、供电模块和绕组中至少一个组件的温度超过了写入到过温下限（temperature_limit_low）中的值。当过温下限（temperature_limit_low）值高于此 BEAR 模块中测得的最高温度时，此警告会自动复位。
- **绝对位置异常**绝对位置读数有误。
- **看门狗超时或** E-STOP 当处于模式 0（力矩模式）且驱动器使能时，看门狗超时会触发此错误。在看门狗超时之前与 BEAR 进行一次通信则复位看门狗。

  外部 ESTOP 信号也会触发此错误，包括物理电信号和写入 0x03 到力矩使能寄存器。
- **超限位** BEAR 实际位置超出位置限制。
- **硬件错误**输入电压超出限制或者场效应管驱动器异常。
- **初始化错误**闪存中的保存文件已损坏，需要对 BEAR 进行校准。

2.3 运行模式

BEAR 驱动器有四种运行模式（mode），分别是：

> **警告**
>
> 当在 BEAR 仍处于使能状态下切换运行模式时请格外当心。BEAR 会在完成模式切换后立即执行新运行模式下相应的 goal_xxx 设定。

0 - Torque （力矩模式）

对 BEAR 的扭矩电流进行控制，从而直接控制其输出扭矩。该模式下的控制通过将控制指令写入状态寄存器 goal_iq 实现，输入单位为：安培。输出扭矩 T 可使用扭矩电流 i_q 和 BEAR 的扭矩常数 K_T 由公式1推得:

$$T = i_q \times K_T \tag{1}$$

> **注意**
>
> 当输入的指令电流 goal_iq 高于设置的最高电流 limit_i_max 时，BEAR 模块自动忽略该指令。

> **注意**
>
> BEAR 的实际输出最高扭矩将受限于当前供电最大电流以及设置的最高扭矩电流 limit_i_max。

1 - Velocity （速度模式）

对 BEAR 的输出速度进行直接控制。该模式下的控制通过将控制指令写入状态寄存器 goal_velocity 实现，控制输入的单位为：角度/秒

> **注意**
>
> 当输入的指令速度 goal_velocity 高于设置的最高速度 limit_velocity_max 时，BEAR 模块自动忽略该指令。

> **注意**
>
> BEAR 的实际输出速度将受限于当前供电电压和负载状况下所能实现的最高转速；实际输出最高扭矩将受限于供电最大电流以及设置的最高扭矩电流 limit_i_max。

2 - Position （位置模式）

对 BEAR 的输出位置进行控制以实现较为精确的位置输出。该模式下的控制通过将控制指令写入状态寄存器 goal_position 实现，控制输入单位为：角度。

> **注意**
>
> 当输入的目标位置 goal_position 超出 limit_position_max 和 limit_position_min 所限定的范围时，BEAR 模块自动忽略该指令。

注意

BEAR 的实际输出速度将受限于当前供电电压、负载状况下所能实现的最高转速以及设置的最高速度 limit_velocity_max；实际输出最高扭矩将受限于当前供电最大电流以及设置的最高扭矩电流 limit_i_max。

3 - Direct Force （直接力控模式）

在这种模式下，BEAR 将跟踪设定的目标位置，同时目标速度、目标 Iq 也可兼顾。换句话说，用户可以使用此模式命令 BEAR 跟踪包含位置、速度和扭矩数据的轨迹。图. 10 展示了该模式的控制环路。

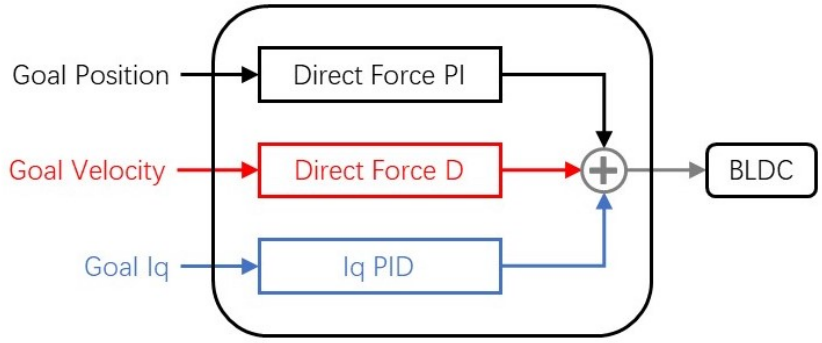


图 10: Direct Force Mode 示意图.

CAUTION

当输入的指令速度 goal_velocity 高于设置的最高速度 limit_velocity_max，或当输入的指令电流 goal_iq 高于设置的最高电流 limit_i_max 时，BEAR 模块自动忽略该指令。然而，当输入的目标位置 goal_position 超出了 limit_position_max 和 limit_position_min 所限定的范围时，BEAR 模块扔将执行该指令。

注意

BEAR 的实际输出速度将**仅**受限于当前供电电压和负载状况下所能实现的最高转速；实际输出最高扭矩将受限于当前供电最大电流以及设置的最高扭矩电流 limit_i_max。

警告

直接力控模式下 BEAR 的实际输出速度不受设置的最高速度 limit_velocity_max 的限制。

2.4 PID 调校

想要使 BEAR 发挥预期功能，良好的 PID 增益调整至关重要。取决于 BEAR 的操作模式，可能需要调校多组 PID 增益以使 BEAR 充分匹配其实际应用。

0 - **扭矩模式**

此模式仅涉及 iq 和 id 回路，因此只需要调整这两个回路的 PID 增益。在正常状况下，iq 和 id 回路的 PID 增益应该始终保持一致，因此当 BEAR 在此模式下运行时，实际上只有一组 PID 增益需要调整。

我们建议按照表 6中给出的数值作为初始设定。这组设定理论上应适用于大多数应用场景。

BEAR	P Gain	I Gain	D Gain
Koala V2	0.277	0.061	0
Koala Muscle Build v1	0.358	0.045	0
Panda v2	0.099	0.039	0
Panda v2 Plus	0.184	0.065	0
Kodiak v1	0.25	0.017	0

表 6: 电流环增益设定

配置了正确 id/iq PID 增益的 BEAR 可出色地追踪 id/iq 目标值。即使需要进一步调校，P 增益和 I 增益也不应显着偏离上述值；如果在一些特殊应用情况下，电流环中出现过多噪音，您可尝试成倍缩小上述设定值，并且 D 增益应始终保持为零 (0)。

1 - **速度模式**

在速度模式下运行时，速度 PID 回路的结果反馈到 iq 回路，因此速度 PID 增益应在正确的 id/iq PID 增益的基础之上进行调整。有关如何正确调整 id/iq PID 增益，请参阅上一节。

虽然实际设置可能因应用场景和所需的动态性能而异，但建议从以下设置开始尝试：

Velocity $\quad \mathrm{P} = 0.5 \sim 1, \mathrm{I} = 0, \mathrm{D} = 0$

根据实际应用，可能需要一个非常小（$0 \sim 0.001$）的 I 增益，但建议将 D 增益保持为零 (0)。

配置了正确的 id/iq PID 增益和合理调校的速度 PID 增益的 BEAR 可以出色地对目标速度进行跟踪。

2 - **位置模式**

为 BEAR 配置合适的 PID 设置使其能够在位置模式下动态而精准地运行可能稍具挑战，然而只要按照正确的顺序逐步调整，这便是一件简单而直截了当的任务。

在位置模式下，位置 PID 环的结果反馈至速度环，速度 PID 环的结果反馈至 iq 环，因此 id/iq 环、速度环和位置环的 PID 增益都需要正确调整。因此，在调整位置增益之前，速度环和 id/iq 环的增益应已经得到合理调校。有关如何正确调整速度环和 id/iq 环的增益，请参阅之前的小节。

位置环的实际 PID 设置在不同型号的 BEAR 及不同应用场景之间差异很大。在花时间调整位置 PID **之前**，请先保证获得正确的速度和 id/iq PID 增益。有些情况下，糟糕的速度环和/或 id/iq 增益可能才是 BEAR 没有输出足够扭矩或无法准确而动态地跟踪目标位置的根本原因，无论其位置增益如何。

3 - **直接力控模式**

当在直接力控模式下运行时，位置、速度和 iq 都可以作为输入信号而混合使用，如 2.3所述，并最终反馈入 Iq 控制环。因此，一组经过良好调校的 id/iq PID 增益是此模式的基础。

在此模式下，BEAR 将模拟一个 “弹簧-阻尼” 系统，Direct Force P 增益等效弹簧刚度，Direct Force D 增益等效阻尼大小。调校时可以从 $\mathrm{D} = 0.1\mathrm{P}$ 开始尝试，但任何 I 增益都可能会导致振荡，因而请慎重使用。

在此模式下，根据实际应用情况，力矩电流 Iq 和速度也可以作为发送到 BEAR 的目标命令的一部分。在这种情况下，速度指令将结合 Direct Force D 增益、电流指令 iq 将结合电流环 PID 增益融合进实际输出。

有关如何调整 id/iq PID 增益的信息，请参阅之前的小节。

3 软件开发工具包（SDK）

3.1 Python

本节将详细介绍 BEAR™ 驱动器的 Python 版驱动程序开发包，PyBEAR™。PyBEAR 兼容 Python3 语言环境，其具体功能和函数如下。

CAUTION

自 2023 年 1 月 1 日起, PyBEAR 不再支持 python2.

3.1.1 入门指南

请按以下步骤安装 PyBEAR 开发包：

0) **串口访问权限** 请确保您的账户有权访问本地计算机的串口。在 Linux 操作系统下，可在终端中执行以下命令来为您的账户添加串口访问权限：

```
sudo chown -R your_username /usr/local
sudo usermod -a -G dialout your_username
```

1) **解压缩** 在下载完成后，请指定一个本地路径并在该路径下解压 PyBEAR 压缩包。本手册的以下内容将以`/usr_path` 指代 PyBEAR 的解压路径。

2) **安装必要组建** PyBEAR 需要 NumPy 模块来进行科学运算，还需要 PySerial 模块来访问串口。如上述两个模块并未安装在本地，请在终端中运行以下命令来完成安装(Python3)：

```
pip3 install numpy pyserial
```

3) **安装** PyBEAR 将当前路径改为`/usr_path/PyBEAR` 并进行安装：

Python 3:

```
pip3 install .
```

PyBEAR 准备就绪。

3.1.2 与 BEAR 的通信

要使用 PyBEAR, 请首先将其导入:

```
# Import Manager from PyBEAR
from pybear import Manager
```

0) **连接** BEAR　与 BEAR 模块沟通之前，必须先通过创建串口对象建立串口连接。

例:

```
# Create a serial port object
# A typical port will be /dev/ttyUSB*
# The default (and fastest) baud rate is 8000000 bps
bear = Manager.BEAR(port='/dev/ttyUSB0', baudrate=8000000)
```

1) **数据读取**　BEAR 驱动器的配置寄存器和状态寄存器中的数据可使用 "get_" 函数读取。

该类函数格式为 "get_name_of_register(motor_id)"，您可以一次读取多个 BEAR 驱动器的寄存器，函数返回值为 <tuple> 组成的 <list>.

例:

```
# Read torque_enable status from BEAR 1, 2, 3
bear.get_torque_enable(1,2,3)

# Read p_gain_position from BEAR 1, 4
bear.get_p_gain_position(1,4)

# Read present_velocity from BEAR 2, 3
bear.get_present_velocity(2,3)
```

如果发生不良连接使得 BEAR 返回的数据包损坏或着甚至完全没有返回数据，PyBEAR 将自动重试读取，并生成如下警告：

```
[PyBEAR | WARNING] :: Status response timed out. Re-sending the same packet.
```

2) **数据写入**　您可以对所有的配置寄存器和大多数状态寄存器进行写入操作，来实现对 BEAR 模块的准确控制。您可以使用 "set_" 函数对支持写入的寄存器进行数据写入操

作，该类函数格式为 “set_name_of_register(*args)”，输入变量 “*args” 是 “(motor_id, value)” 格式的 <tuple>，或者当同时向多个 BEAR 模块的相同寄存器进行写入操作时，输入变量为多个 “(motor_id, value)” 格式的 <tuple> 组成的 <tuple>。数据写入函数没有返回值。

例:

```
# Enable the torque of BEAR 1, 3 and disable BEAR 2
bear.set_torque_enable((1,1),(2,0),(3,1))

# set p_gain_velocity on BEAR 1 to 0.02, BEAR 4 to 0.05
bear.set_p_gain_position((1,0.02),(4,0.05))

# set goal_position of BEAR 1 to 0, BEAR 3 to 10000
bear.set_goal_position((1,0),(3,2.5))
```

配置寄存器和状态寄存器

配置寄存器是非易失寄存器，它们在断电后数据不会丢失，但其数据一旦更改就需要进行保存操作；状态寄存器是易失性的，BEAR 每次重新上电时将被恢复为默认值，并用相应的 BEAR 实时状态填充其数据。因此每次对配置寄存器进行更新后，可以用 “save_config(motor_id)” 函数保存配置数据。

以更改 BEAR 模块的 ID 为例：

```
# Set the ID of motor (1) (default) to (4)
bear.set_id((1,4))
# Save the configuration using the new motor ID
bear.save_config(4)
```

注意

“save_config” 目前只支持对单个 ID 进行操作，以下指令无效：

```
bear.save_config(1,2,3)
```

> **警告**
>
> BEAR 电机出于使能状态时，请勿进行 “save_config” 操作。电机在对内置存储器进行写入操作时可能会停止相应。

配置寄存器包括：

- Motor ID
- Operation Mode
- Baud Rate
- Home offset
- Limits (position, velocity, current, voltage, temperature)
- PID gains

状态寄存器包括

- Torque enable/disable
- Goal position/velocity/current
- Present position/velocity/current/voltage/temperature

BEAR 寄存器的完整列表请参考表. 4和。

3) **批量通信（测试功能）** 使用 “set_bulk_regtype(*args)” 和 “get_bulk_regtype(*args)” 函数可以便捷地实现对一个或多个 BEAR 驱动器的多个寄存器进行读写操作。输入变量 “*args” 应为一个或多个 “(motor_id, ‘name_of_register’, value, ‘name_of_register’, value)” 格式的 <tuple> 组成的 <tuple>。“get_bulk_” 函数返回包含指令里所要求的返回值的 <list>

例：

```
# Writing to multiple registers
# Set BEAR 1 to Torque mode, and set it's 'p_gain_id' to 1,
# 'd_gain_id' to 0.3
bear.set_bulk_config((1, 'mode', 0,
```

```
                        'p_gain_id', 1,
                        'd_gain_id', 0.3))
# Set BEAR 1 'goal_position' to 1000, BEAR 2, 'goal_position' to 0
bear.set_bulk_status((1, 'goal_position', 1.5),
                     (2, 'goal_position', 0))

# Reading from multiple registers
# Read the 'p_gain_id' and 'd_gain_id' from BEAR 1
bear.get_bulk_config((1, 'p_gain_id', 'd_gain_id'))
# Read the 'present_iq' and 'present_velocity' from BEAR 1
# as well as the 'ic_temperature' and 'error_status' from BEAR 2
bear.get_bulk_status((1, 'present_iq', 'present_velocity'),
                     (2, 'ic_temperature', 'error_status'))
```

当仅对状态寄存器进行读写操作时，可分别使用 “bulk_read(motor_ids, registers)” 函数和 “bulk_write(motor_ids, registers, values)” 函数来同时对多个 BEAR 模块的一组状态寄存器进行批量读写操作。

例:

```
# Read present_iq, present position and present_velocity from BEAR 1, 2, 3
bear.bulk_read([1, 2, 3],
               ['present_iq', 'present position', 'present_velocity'])
# Write to goal_position to BEAR 1, 2, 3
bear.bulk_write([1, 2, 3],
                ['goal_position'], [2.15])
```

另一个批量通信函数 “bulk_read_write(motor_ids, read_registers, write_registers, values)” 是上述 “bulk_read” 和 “bulk_write” 函数的功能组合，它可以仅用一次通信将 “values” 中指明的数值分别写入 “motor_ids” 列出的 BEAR 模块上由 “write_registers” 所指明的状态寄存器中，同时向该组 BEAR 读取 “read_registers” 中指明的状态寄存器的数据。

前面的例子中的读写操作用 “bulk_read_write” 实现的例子如下:

```
# Write to goal_position of BEAR 1, 2, 3
# and read back their present_iq, present position and present_velocity
```

```
bear.bulk_read_write([1, 2, 3],
                     ['present_iq', 'present position', 'present_velocity'],
                     ['goal_position'], [2.15])
```

4) **错误代码** 所有的数据读取函数都会返回一个错误代码，且在函数返回值中由每个 BEAR 返回的读取值和错误代码组成一个 <tuple>。

例:

```
# Get present position of BEAR 1
bear.get_present_position(1)

-> return: [(0.0855, 128)]
# 0.0855 is the present position and 128 is the error code
```

错误代码 128 代表正常。详细的错误代码含义请参考表. 7。

bit	Type	Name	Description
0	Warning	Communication	
1	Warning	Overheat	
2	Error	Absolute Position	
3	Error	Watchdog Timeout	
4	Error	Joint Limit	
5	Error	Hardware Fault	
6	Error	Initialization Error	
7	1		Always 1

表 7: 错误代码

3.2 LabBEAR

本节将详细介绍 BEAR™ 驱动模块的 LabVIEW 驱动——LabBEAR。

3.2.1 入门指南

请按照以下步骤做好 LabBEAR 使用前的准备工作:

0) **检查** LabVIEW **版本** LabBEAR 要求 LabVIEW 2016 或更高版本，所以如果您本地的 LabVIEW 版本过低，请先进行升级。

1) **下载**下载 LabBEAR SDK 压缩包并解压到您选定的工作路径

2) **安装附属插件** LabBEAR 需要搭配 NI VISA 模块来使用您计算机上的 USB 接口与 BEAR 通信。请使用 NI Package Manager 安装 NI VISA 模块。

LabBEAR 准备就绪。

3.2.2 与 BEAR 的通信

0) **打开** VISA **接口**使用 VISA Configure Serial Port 以及 VISA Set I/O Buffer Size 和 VISA Flush I/O Buffer 来建立与 BEAR 的 USB 通信。

Find under: as 如图11所示，在 Instrument I/O → Serial 选项卡中找到 VISA Configure Serial Port subVI：

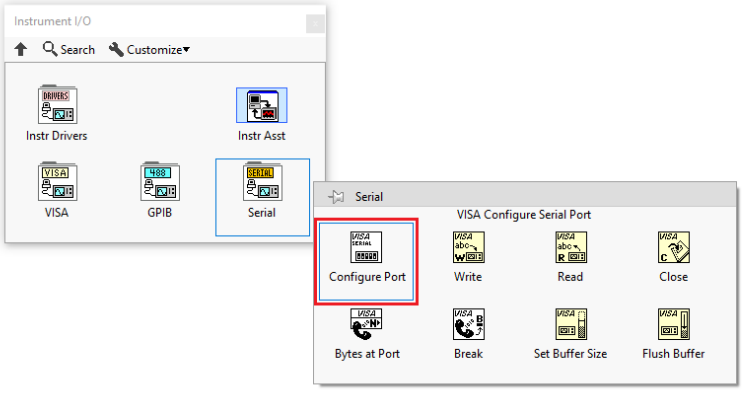


图 11: VISA Configure Serial Port

然后在 Instrument I/O → VISA → Advanced → Bus Specific 选项卡中找到 VISA Set I/O Buffer Size 和 VISA Flush I/O Buffer，如图12所示：

按图13链接上述三个 subVI，并为 VISA Configure Serial Port 创建 control 来设定 VISA resource name 和 baud rate，并将超时设置为 200ms、将 termination char 设置为 FALSE；

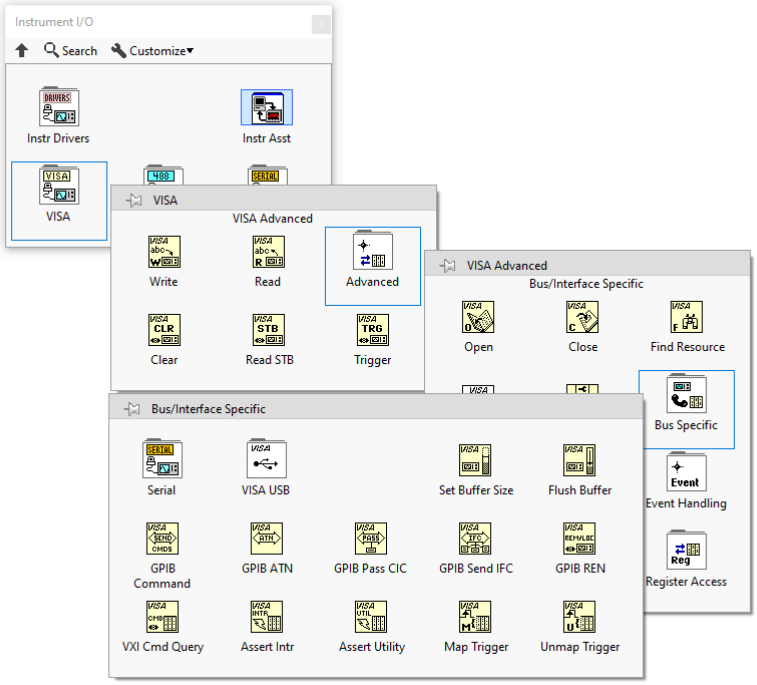


图 12: VISA Set I/O Buffer Size an 和 Flush I/O Buffer

将 VISA resource name 和 error 传递给 VISA Set I/O Buffer Size 并将 I/O Receive Buffer 设置为 10000 bytes；最后，将 VISA resource name 和 error 传递给 VISA Flush I/O Buffer。现在您已完成用于打开 VISA 接口的程序，从 VISA Flush I/O Buffer 输出的 VISA resource name 和 error 已可供后续程序使用。

请注意，如果 com port 和 baud rate 是预先设定好的，您可以在这两个位置使用常量而非 control。否则，请在人机界面（front panel）相应地设定 com port 和 baud rate，如图13所示。

1) **数据读取** BEAR 驱动器的配置寄存器和状态寄存器中的数据可分别使用 Read_Config 和 Read_Stat 这两个 subVIs 来读取, 如图14所示。

使用数据读取 subVI 时，将 VISA resource 和 Error 与对应通道连接，然后通过在对应的通道建立 control 或者 constant，来指定 BEAR 的 ID 以及目标寄存器的地址（名称）。您可以一次读取多个寄存器，数据读取 subVI 将返回目标寄存器中的数据以及目 BEAR 的错误代码。

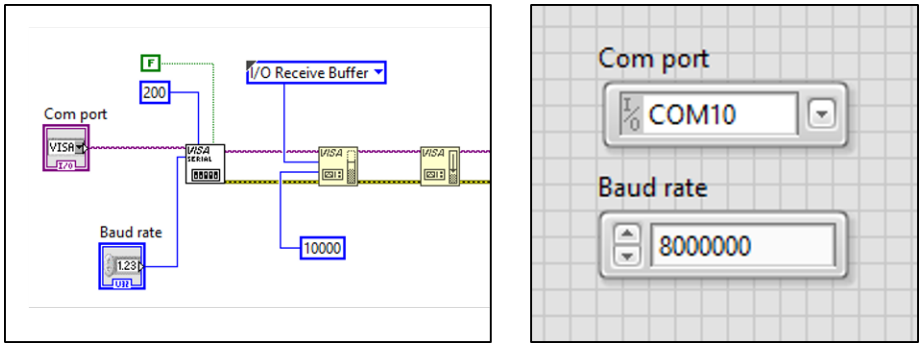


图 13: 打开一个与 BEAR 通信的 VISA 端口。

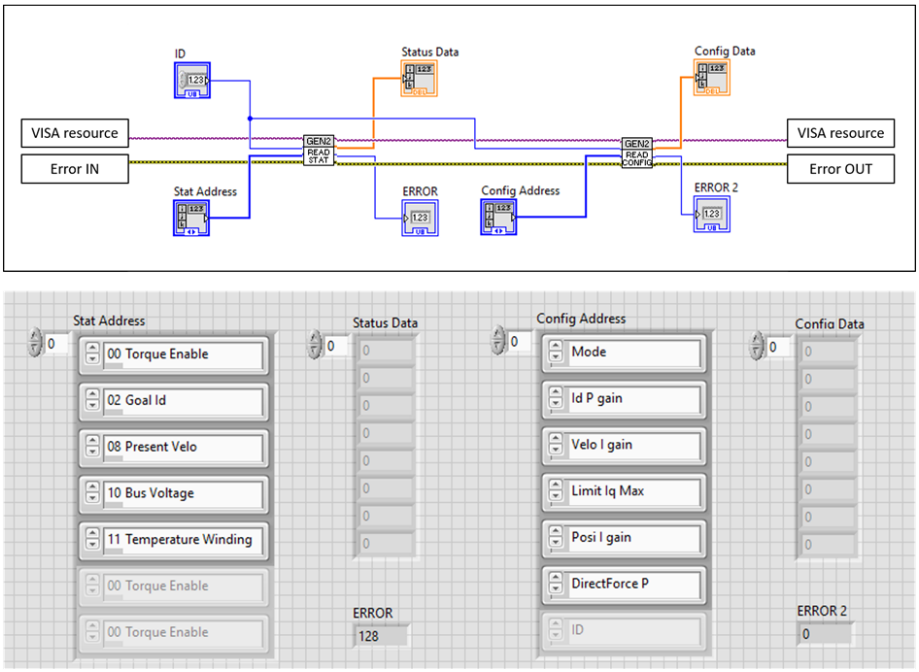


图 14: 数据读取

2) **数据写入**您可以分别使用 Write_Config 和 Write_Stat 这两个 subVI 来对支持的配置寄存器和状态寄存器进行数据写入，如图15所示。

使用数据读取写入 subVI 时，将 VISA resource 和 Error 与对应通道连接，然后通过在对应的通道建立 control 或者 constant，来指定 BEAR 的 ID 以及目标寄存器的地址（名称）。您可以一次对多个寄存器进行数据写入操作。

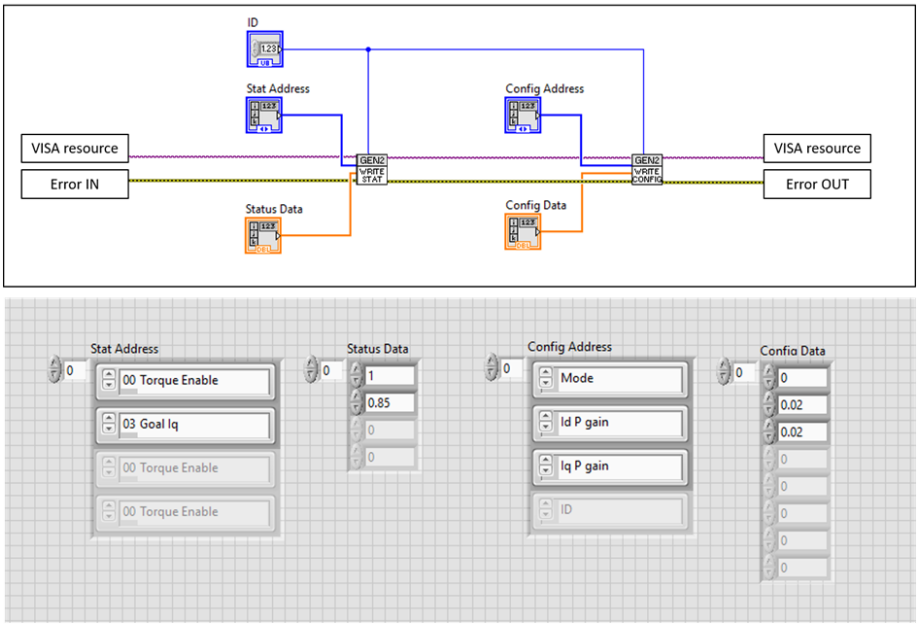


图 15: 数据写入

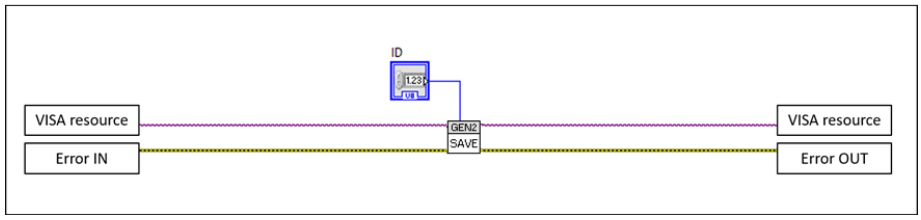


图 16: 保存配置寄存器

配置寄存器和状态寄存器

配置寄存器是非易失寄存器，它们在断电后数据不会丢失，但其数据一旦更改就需要进行保存操作；状态寄存器是易失性的，BEAR 每次重新上电时将被恢复为默认值，并用相应的 BEAR 实时状态填充其数据。因此每次对配置寄存器进行更新后，可以用 Save_Config subVI 保存配置数据，如图16所示。

警告

BEAR 电机出于使能状态时，请勿进行 “save_config” 操作。电机在对内置存储器进行写入操作时可能会停止相应。

配置寄存器包括：

- Motor ID
- Operation Mode
- Baud Rate
- Home offset
- Limits (position, velocity, current, voltage, temperature)
- PID gains

状态寄存器包括

- Torque enable/disable
- Goal position/velocity/current
- Present position/velocity/current/voltage/temperature

BEAR 寄存器的完整列表请参考表. 4。

3) **批量通信**您可以使用 Bulk_Stat subVI 来同时对单个或多个 BEAR 的单个或多个状态寄存器进行读写，使用方法如图17所示。

> **使**
> 用 Bulk_Stat subVI 的批量通信只适用于支持的状态寄存器。

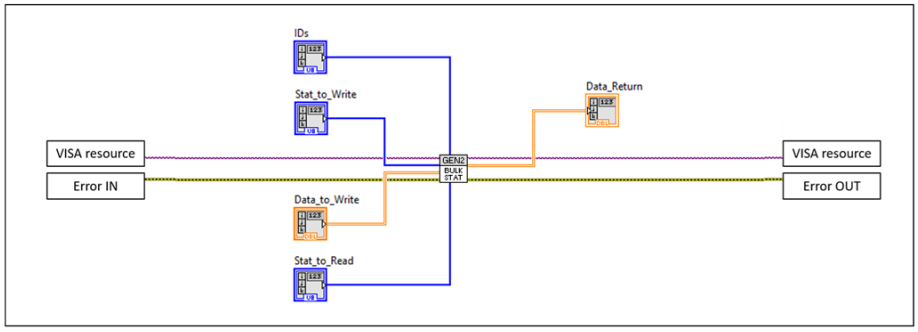


图 17: 状态寄存器的批量读写。

4) **例程**在 LabBEAR 程序包中还包含如下几个例程谨供参考：

- SIMPLE_Read_Stat 对一个 BEAR 模块的状态寄存器进行简单读取。
- SIMPLE_Read_Config 对一个 BEAR 模块的配置寄存器进行简单读取。
- SIMPLE_Write_Stat 对一个 BEAR 模块的状态寄存器进行简单写入。
- SIMPLE_Write_Config 对一个 BEAR 模块的配置寄存器进行简单写入。
- SIMPLE_Bulk_Stat 批量通信例程

5) **调校界面**: Data_Ctrl 当使用 LabVIEW 调校和调试您的 BEAR 时，SDK 包中的 Data_Ctrl VI 是一款非常好用的工具，其人机界面如图18所示。接下来将分区域介绍 Data_Ctrl。

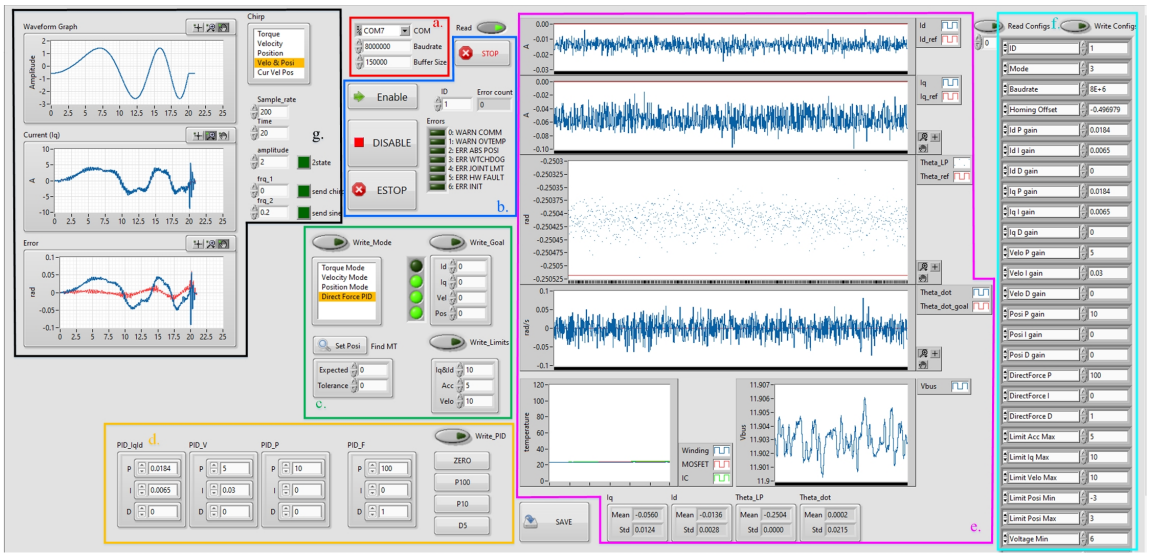


图 18: Data_Ctrl VI 的人机界面。

a. 通信接口设置 运行 VI 之前，请在这里根据您的 BEAR 进行相应设置。

b. ID、使能和错误代码 您可以串联多个 BEAR 模块并通过更改 ID 设置来依次对其进行调试。您可以在本 VI 运行中随时更改 ID 设置。您可以使用 Enable 和 Stop 按钮来使能/禁用当前 BEAR，并可以使用STOP按钮来终止本 VI。当前 BEAR 的错误代码将实时显示在 ERROR 显示框中，Error count 显示框中显示的是从本 VI 运行以来，BEAR 错误发生的次数。

c. 模式，限制和目标指令 使用本区域中的功能来对当前连接的 BEAR 进行模式、限制和目标指令的写入；您可以使用 SAVE 按钮来保存当前配置寄存器中的数据。

> **注意**
> 本 VI 可能会在执行配置寄存器数据保存操作时极其短暂地停滞一下。

d. PID 设置 使用本区域来对当前连接的 BEAR 模块的 PID 设置进行更改。

e. 实时状态 当前连接的 BEAR 模块的实时状态以图线的形式显示在本区域中，包括励磁电流 id、扭矩电流 iq、位置 Theta、速度 Theta_dot、BEAR 中各模块的温度以及当前供电电压 Vbus。

f. 读取配置 单击 Read Config 按钮来对当前连接的 BEAR 模块的所有配置寄存器进行一次读取，返回值将显示在显示框中。

g. 调频信号 使用本区域中的功能来对当前连接的 BEAR 模块发送调频、正弦或方波信号来辅助调校。使用本区域中的控制选项来定义您的波形，并选择目标扭矩 goal_torque、目标速度 goal_velocity 或目标位置 goal_position 中的一种信号模式，最后点亮对应的绿色方块开始发送波形信号。当针对某个相应频段对一个 BEAR 模块进行精细调校时，本功能非常有效。

4 文档版本历史

— Update Log: —

0.1.2 Complete Python section, Introduction now contains everything about KB02

0.1.3 Add LabBEAR

0.1.4 Add Warning section in Intro

0.1.5 Modified KT value and torque specs.

0.1.6 Add details on PB02; Add connection instruction.

0.2.0 A couple of updates:

a. Move connection, PID tuning and control table all into a new section: Using BEAR.

b. Add units to regs and correct reg value range.

c. Add more details to PID tuning.

d. Update file structure.

0.2.1 Update Ping function output: it outputs [([List of firmware and hardware versions], error), ...]

0.2.2 Update Specification

0.2.3 Officially support BULK_COMM in PyBEAR.

a. Add detailed description and examples of related functions.

b. Depreciate get/set_bulk_status/config() as these are in fact not bulk_comm and replace with get/set_status/config().

c. User can specify timeout and bulk_timeout when instancing pybear.

0.2.4 Update LabBEAR to 2.1: Update Introduction section to include the usage of ESTOP

0.2.5 Update PB02 Spec figure and signal port access instruction

0.2.6 Small Bug fix:

a. Fix typo in PyBEAR multi-communication.

b. Fix daisy chain fig missing Estop wire.

0.2.7 Minor improvements.

0.2.8 Update recommended current loop PID gains.

0.2.9 Fix a wrong statement about velocity in DF mode

0.3.1 Fix a wrong labeling about U2B pinout

0.3.2 Add recommended current loop PID gains for Kodiak BEAR V1(CB01) and Koala BEAR Muscle Build V1(KBMB01), and attach version history at the end of manual. Add Kodiak voltage range. Officially stopped support for python2, Section 3. Update BEAR spec table as well as introduction to include KBMB01 and CB01.

0.3.3 Modify Figure 6: USB2BEAR pin-out and switches for better understanding.