

GXN 系列运动控制器编程手册

补充说明及补充指令

R1.1

2022 年 3 月

© 2022 固高科技 版权所有

版权申明

固高科技股份有限公司

保留所有权力

固高科技股份有限公司（以下简称固高科技）保留在不事先通知的情况下，修改本手册中的产品和产品规格等文件的权力。

固高科技不承担由于使用本手册或本产品不当，所造成直接的、间接的、特殊的、附带的或相应产生的损失或责任。

固高科技具有本产品及其软件的专利权、版权和其它知识产权。未经授权，不得直接或者间接地复制、制造、加工、使用本产品及其相关部分。



运动中的机器有危险！使用者有责任在机器中设计有效的出错处理和安全保护机制，固高科技没有义务或责任对由此造成的附带的或相应产生的损失负责。

联系我们

固高科技股份有限公司

地址：深圳市高新技术产业园南区深港产学研
基地西座二楼 W211 室

电话：0755-26970817 26737236 26970824

传真：0755-26970821

电子邮件：googol@googoltech.com

网址：<http://www.googoltech.com.cn>

固高科技（海外）有限公司

地址：香港九龍觀塘偉業街 108 號絲寶國際大
廈 10 樓 1008-09 室

電話：+(852) 2358-1033

傳真：+(852) 2719-8399

電子郵件：sales@googoltech.com

info@googoltech.com

網址：<http://www.googoltech.com>

臺灣固高科技股份有限公司

地址：台中市西屯區福中二路 10 巷 22 號 2 樓

電話：+886-4-23588245

傳真：+886-4-23586495

電子郵件：twinfo@googoltech.com

文档版本

| 版本号 | 修订日期 |
|-----|-------------|
| 1.0 | 2020年05月09日 |
| 1.1 | 2021年07月15日 |
| | |
| | |
| | |
| | |

目录

| | |
|---|----|
| 版权申明 | 1 |
| 联系我们 | 1 |
| 文档版本 | 2 |
| 目录 | 3 |
| 一、 | 7 |
| 二、 指令返回值 | 7 |
| 三、 指令参数范围 | 8 |
| 四、 补充指令列表 | 10 |
| 五、 指令详细说明 | 15 |
| 1. 位置比较功能 | 15 |
| 指令 1 GTN_SetPosCompareFifoMode | 15 |
| 指令 2 GTN_GetPosCompareFifoMode | 15 |
| 指令 3 GTN_GetPosCompareLatchValue..... | 15 |
| 指令 4 GTN_PosComparePulse | 16 |
| 指令 5 GTN_SetPosComparePulseCount..... | 17 |
| 指令 6 GTN_SetPosComparePulseDuty..... | 17 |
| 指令 7 GTN_PosComparePulseEx..... | 17 |
| 指令 8 GTN_GetPosComparePulseStatus..... | 18 |
| 指令 9 GTN_SetHsoPulsePrm..... | 18 |
| 指令 10 GTN_GetHsoPulsePrm | 20 |
| 指令 11 GTN_BufPosComparePulse | 22 |
| 指令 12 GTN_BufPosComparePulseEx | 22 |
| 指令 13 GTN_SetPosComparePsoSyncPrm | 23 |
| 2. 激光补充指令 | 24 |
| 指令 14 GTN_GetLaserOnOffCount..... | 24 |
| 指令 15 GTN_ClearLaserOnOffCount | 24 |
| 3. 压力补偿 | 24 |
| 指令 16 GTN_SetAxisPressCompensate | 24 |
| 指令 17 GTN_GetAxisPressCompensate | 25 |
| 指令 18 GTN_SetAxisPressCompensateTable | 26 |
| 指令 19 GTN_SelectAxisPressCompensateTable..... | 26 |
| 4. Trigger 触发重新规划运动..... | 26 |
| 指令 20 GTN_SetTriggerProfilePrm..... | 26 |
| 指令 21 GTN_GetTriggerProfileStatus..... | 27 |
| 5. 限位开关 | 29 |
| 指令 22 GTN_LmtsOnEx | 29 |
| 指令 23 GTN_LmtsOffEx | 30 |
| 6. 高精度当量变换 | 30 |
| 指令 24 GTN_SetProfileScale | 30 |
| 指令 25 GTN_GetProfileScale..... | 30 |
| 指令 26 GTN_SetEncoderScale..... | 31 |
| 指令 27 GTN_GetEncoderScale | 31 |

| | | |
|-----|---|----|
| 7. | 摩擦力补偿 | 31 |
| | 指令 28 GTN_SetFriction | 31 |
| | 指令 29 GTN_GetFriction | 32 |
| | 指令 30 GTN_EnableFriction | 32 |
| | 指令 31 GTN_DisableFriction | 32 |
| 8. | 高速读 | 33 |
| | 指令 32 GTN_LoadReadHsConfig | 33 |
| | 指令 33 GTN_ReadHsOn | 34 |
| | 指令 34 GTN_SetAxisArriveMode | 34 |
| | 指令 35 GTN_GetStsEx | 34 |
| 9. | 功能复位选择设置 | 35 |
| | 指令 36 GTN_SetResetMode | 35 |
| 10. | 读取绝对式编码器值 | 35 |
| | 指令 37 GTN_RN_GetAbsEncPosEx | 35 |
| | 指令 38 GTN_RN_SetEncMultiLinesEx | 36 |
| 11. | 脉冲当量不一致的 PSO 功能 | 36 |
| | 指令 39 GTN_SetPosComparePsoPrmEx | 36 |
| | 指令 40 GTN_GetPosComparePsoPrmEx | 38 |
| 12. | 插补轮廓误差控制 | 39 |
| | 指令 41 GTN_SetCrdContourErrorControl | 39 |
| 13. | 同时设置/读取多轴参数相关指令 | 40 |
| | 指令 42 GTN_SetPosArray | 40 |
| | 指令 43 GTN_GetPosArray | 40 |
| | 指令 44 GTN_SetVelArray | 41 |
| | 指令 45 GTN_SetTrapPrmArray | 41 |
| | 指令 46 GTN_SetTriggerArray | 42 |
| | 指令 47 GTN_ClearTriggerStatusArray | 43 |
| 14. | 检查网络结构是否和配置文件中一致 | 43 |
| | 指令 48 GTN_CheckRingNetStructure | 43 |
| 15. | 驱动器功能 | 44 |
| | 注意事项: | 44 |
| | 指令 49 GTN_GetMotionMode | 45 |
| | 指令 50 GTN_SetMotionMode | 45 |
| | 指令 51 GTN_SetDac | 46 |
| | 指令 52 GTN_SetDrvPrfVel | 46 |
| | 指令 53 GTN_SetPrfTorque | 46 |
| | 指令 54 GTN_GetAtlTorque | 47 |
| | 指令 55 GTN_GetDriverSts | 47 |
| | 指令 56 GTN_SetServoPosLoopPid | 48 |
| | 指令 57 GTN_GetServoPosLoopPid | 48 |
| | 指令 58 GTN_RN_SetTorqueLimit | 49 |
| | 指令 59 GTN_RN_GetTorqueLimit | 49 |
| | 指令 60 GTN_RN_GetServoAlarmInfo | 49 |
| | 指令 61 GTN_ReadServoParamInfo | 50 |
| 16. | 回零功能 | 51 |
| | 指令 62 GTN_GoHome | 51 |

| | | |
|-----|--|----|
| 17. | Trigger 功能 | 53 |
| | 指令 63 GTN_SetTriggerEx | 53 |
| | 指令 64 GTN_GetTriggerStatusEx | 53 |
| | 指令 65 GTN_GetTriggerLatchValue | 54 |
| 18. | 缓冲区等待功能 | 55 |
| | 指令 66 GTN_BufWaitDi | 55 |
| | 指令 67 GTN_BufWaitLongVar | 56 |
| | 指令 68 GTN_GetBufWaitDiStatus | 57 |
| | 指令 69 GTN_GetBufWaitLongVarStatus | 58 |
| | 指令 70 GTN_ClearBufWaitStatus | 58 |
| | 指令 71 GTN_SetLongVar | 58 |
| | 指令 72 GTN_GetLongVar | 59 |
| 19. | 椭圆插补 | 59 |
| | 指令 73 GTN_Ellipse | 59 |
| | 指令 74 GTN_EllipseEx | 60 |
| 20. | 振镜功能 | 62 |
| | 指令 75 GTN_ScanBufLaserDelayLong | 62 |
| | 指令 76 GTN_SetScanAlarmAutoStopMode | 62 |
| | 指令 77 GTN_GetScanAlarmAutoStopMode | 62 |
| | 指令 78 GTN_GetScanErrorCode | 63 |
| 21. | 52 开卡模式读取物理站号和轴号 | 63 |
| | 指令 79 GTN_GetResPhyInfo | 63 |
| 22. | 运控模式配置功能 | 63 |
| | 指令 80 GTN_SetMcMode | 63 |
| | 指令 81 GTN_GetMcMode | 64 |
| 23. | 前瞻功能 | 65 |
| | 指令 82 GTN_SetRadiusRatioTableLa | 65 |
| 24. | 点位运动新增功能 | 65 |
| | 指令 83 GTN_GetTrapSts | 65 |
| | 指令 84 GTN_ClearTrapSts | 66 |
| | 指令 84 GTN_SetPosEx | 66 |
| | 指令 84 GTN_GetPosEx | 66 |
| 25. | 手轮导引功能 | 67 |
| | 指令 85 GTN_SetCrdMPGMode | 67 |
| | 指令 86 GTN_GetCrdMPGMode | 67 |
| | 指令 87 GTN_SetCrdMPGModeEx | 68 |
| | 指令 88 GTN_GetCrdMPGModeEx | 69 |
| 26. | Event-Task 新增功能 | 70 |
| | 指令 89 GTN_AddTask | 70 |
| | 指令 90 GTN_GetTaskSaveMcVarResult | 72 |
| 27. | 编码器相关配置功能 | 73 |
| | 指令 91 GTN_SetEncoderMapRelation | 73 |
| | 指令 92 GTN_GetEncoderMapRelation | 74 |
| | 指令 93 GTN_SetEncoderDeltaLimit | 74 |
| | 指令 94 GTN_GetEncoderDeltaLimit | 75 |
| 28. | GT 指令按照物理寻址 | 75 |

| | | |
|--------|--|----|
| 指令 95 | GTN_ReadPhysicalMap | 75 |
| 指令 96 | ConvertPhysical | 75 |
| 29. | 补偿功能 | 76 |
| 指令 97 | GTN_SetLeadScrewCrossComp | 76 |
| 指令 98 | GTN_EnableLeadScrewCrossComp | 76 |
| 指令 99 | GTN_SetTransformOrthogonal | 76 |
| 指令 100 | GTN_GetTransformOrthogonal | 77 |
| 指令 101 | GTN_GetTransformOrthogonalPosition | 78 |
| 30. | 插补单步执行功能 | 78 |
| 指令 102 | GTN_CrdStepMode | 78 |
| 指令 103 | GTN_CrdStartStep | 79 |
| 31. | 平滑功能 | 79 |
| 指令 104 | GTN_SetStopSmoothTime | 79 |
| 指令 105 | GTN_GetStopSmoothTime | 79 |
| 指令 106 | GTN_SetAxisMotionSmooth | 80 |
| 指令 107 | GTN_GetAxisMotionSmooth | 81 |
| 指令 108 | GTN_SetCrdJerkTime | 82 |
| 指令 109 | GTN_GetCrdJerkTime | 82 |
| 32. | 模块断线状态及安全模式设置 | 83 |
| 指令 110 | GTN_RN_GetStationOfflineCount | 83 |
| 指令 111 | GTN_RN_SetStationSafeModeOut | 83 |
| 指令 112 | GTN_RN_ClearStationSafeModeStatus | 83 |
| 指令 113 | GTN_RN_SetStationSafeModeControl | 84 |
| 指令 114 | GTN_RN_IlinkSetSafeModeOut | 84 |
| 指令 115 | GTN_RN_IlinkClearSafeModeStatus | 85 |
| 指令 116 | GTN_RN_IlinkSetSafeModeControl | 85 |
| 33. | 网络恢复指令 | 85 |
| 指令 117 | GTN_RN_Recover | 85 |
| 34. | 批量读取控制器状态 | 86 |
| 指令 118 | GTN_GetMcVarArray | 86 |
| 35. | 串行通讯指令（本地 485/232 通信指令） | 88 |
| 指令 119 | GT_RN_ComOpen | 88 |
| 指令 120 | GT_RN_ComClose | 88 |
| 指令 121 | GT_RN_ComRead | 88 |
| 指令 122 | GT_RN_ComWrite | 89 |
| 指令 123 | GT_RN_ComGetState | 89 |
| 指令 124 | GT_RN_ComSetSettings | 90 |
| 指令 125 | GT_RN_ComClearErr | 90 |
| 指令 126 | GT_RN_ComSetMode | 90 |
| 36. | 串行通讯指令（通用 485/232 通信指令） | 91 |
| 指令 127 | GTN_RN_SerialComOpen | 91 |
| 指令 128 | GTN_RN_SerialComClose | 91 |
| 指令 129 | GTN_RN_SerialComRead | 92 |
| 指令 130 | GTN_RN_SerialComWrite | 92 |
| 指令 131 | GTN_RN_SerialComGetState | 92 |
| 指令 132 | GTN_RN_SerialComSetSettings | 93 |

| | | |
|-------------|--------------------------------------|----|
| 指令 133 | GTN_RN_SerialComClearErr | 93 |
| 指令 134 | GTN_RN_SerialComSetMode | 94 |
| 指令 135 | GTN_RN_SerialComGetRecvFifoCnt | 94 |
| 六、 例程 | | 95 |

一、指令返回值

| 返回值 | 说明 | 可能的原因 | 解决措施 |
|-----|--------------|---|--|
| 0 | 指令执行正常 | / | / |
| -2 | 读取数据长度错误 | PCI 通信异常，通信区被破坏，数据丢失或包含了杂数据 | PCI 通信异常的原因很多，需要根据具体问题具体分析。 可以先尝试更换 PCI 插槽，清除控制卡 PCI 上金手指的灰尘，实在不行只能寄回工厂对硬件进行检测。 |
| -3 | 读取数据校验和错误 | 数据在 PCI 传输过程中受到干扰，电平发生改变，导致数据被更改，因此是无效数据 | |
| -4 | 写入数据块错误 | Windows API 函数向 PCI 写或读数据出错，例如 PCI 通信异常，或者没有创建 PCI 通信就向通信区读写数据 | |
| -5 | 读取数据块错误 | | |
| -6 | 打开/关闭设备错误 | 打开卡时，没有控制卡、或者控制卡数量超过最大设定值（4 张卡）、创建 PCI 通信区失败 关闭卡时，已经没有卡、或者关闭 PCI 通信区失败 | / |
| -7 | DSP 忙 | 调用 GT 指令后，DSP 仍然在处理，不再接收新指令 | / |
| -8 | 多线程资源忙 | GT 指令在线程里执行超时才返回，有可能是 PCI 通信异常，导致 GT 指令无法及时返回 | / |
| 1 | 错误调用指令 | 相关的指令没有调用，该指令的执行条件不满足 | / |
| 7 | 参数错误 | 输入指令的参数出错 | / |
| 8 | DSP 固件不支持该指令 | DSP 固件不支持该指令对应的功能 | / |

二、指令参数范围

GXN 产品指令参数范围

| 参数名称 | GTN-024-BB-CC | | GTN-016-BB-CC | |
|----------------------|------------------------------|--------------|---------------|--------------|
| 内核 | [1,32] | | [1,32] | |
| 内核序号 | (card ^[1] -1)*2+1 | (card-1)*2+1 | (card-1)*2+1 | (card-1)*2+1 |
| 轴 | [1,12] | [1,12] | [1,8] | [1,8] |
| 插补坐标系序号 | [1, 2] | [1, 2] | [1, 2] | [1, 2] |
| 插补缓存区序号 | [0, 1] | [0, 1] | [0, 1] | [0, 1] |
| 网络端子板模块序号 | [1,3] | [1,3] | [1,2] | [1,2] |
| 参数个数 | [1,8] | [1,8] | [1,8] | [1,8] |
| 伺服控制器 ^[2] | [1,12] | [1,12] | [1,8] | [1,8] |
| 非轴模拟量输出 | [1,6] | [1,6] | [1,6] | [1,6] |
| 非轴模拟量输入 | [1,24] | [1,24] | [1,16] | [1,16] |
| 通用输入 | [1,66] | [1,66] | [1,44] | [1,44] |
| 通用输出 | [1,30] | [1,30] | [1,20] | [1,20] |
| 位置比较输出 | [1,6] | [1,6] | [1,4] | [1,4] |
| 辅助编码器 | [1,6] | [1,6] | [1,4] | [1,4] |
| 手轮 | [1,3] | [1,3] | [1,2] | [1,2] |

[1]: card 指主卡个数, 例: 第 1 张卡对应 card=1, core 为 1, 2

[2]: 对于 GTN-016-G-BB 和 GTN-024-G-BB 这两款主卡类型, 伺服控制器资源个数为 0

(续 1) GXN 产品指令参数范围

| 参数名称 | GSN-024-AA-BB | | GSN-048-AA-BB | | GSN-008-LT |
|----------------------|---------------|--------------|---------------|--------------|--------------|
| 内核 | [1,32] | | [1,32] | | [1,32] |
| 内核序号 | (card-1)*2+1 | (card-1)*2+2 | (card-1)*2+1 | (card-1)*2+2 | (card-1)*2+1 |
| 轴 | [1,12] | [1,12] | [1,24] | [1,24] | [1,8] |
| 插补坐标系序号 | [1, 2] | [1, 2] | [1, 2] | [1, 2] | [1,2] |
| 插补缓存区序号 | [0, 1] | [0, 1] | [0, 1] | [0, 1] | [0,1] |
| 网络端子板模块序号 | [1,4] | [1,4] | [1,4] | [1,4] | [1,4] |
| 参数个数 | [1,8] | [1,8] | [1,8] | [1,8] | [1,8] |
| 伺服控制器 ^[1] | [1,12] | [1,12] | [1,24] | [1,24] | [1,8] |
| 非轴模拟量输出 | [1,6] | [1,6] | [1,12] | [1,12] | [1,6] |
| 非轴模拟量输入 | [1,24] | [1,24] | [1,48] | [1,48] | [1,24] |
| 通用输入 | [1,100] | [1,100] | [1,100] | [1,100] | [1,100] |
| 通用输出 | [1,40] | [1,40] | [1,40] | [1,40] | [1,40] |
| 位置比较输出 | [1,8] | [1,8] | [1,8] | [1,8] | [1,8] |
| 辅助编码器 | [1,9] | [1,9] | [1,18] | [1,18] | [1,9] |
| 手轮 | [1,3] | [1,3] | [1,3] | [1,3] | [1,3] |

[1]: 对于 GSN-024-G-00, GSN-048-G-00, 伺服控制器资源个数为 0

[2]: 网络端子板模块序号取值[1,4]是采用 GTN_OPen(5,1)开卡方式，向下兼容，如果采用 GTN_Open(5,2)开卡方式，那么取值范围为[1,单核最大轴数]，例如对于 GSN-024-AA-BB，则为[1,12]

(续 2) GXN 产品指令参数范围

| 参数名称 | GTC-NC610 |
|----------------------|----------------------|
| 内核 | [1,32] |
| 内核序号 | $(card^{[1]}-1)*2+1$ |
| 轴 | [1,24] |
| 插补坐标系序号 | [1, 2] |
| 插补缓存区序号 | [0, 1] |
| 网络端子板模块序号 | [1,64] |
| 参数个数 | [1,8] |
| 伺服控制器 ^[2] | [1,24] |
| 非轴模拟量输入 | [1,24] |
| 通用输入 | [1,100] |
| 通用输出 | [1,40] |
| 位置比较输出 | [1,8] |
| 辅助编码器 | [1,9] |
| 手轮 | [1,3] |

三、补充指令列表

| 指令 | 说明 |
|------------------------------------|--------------------------------------|
| 位置比较功能 | |
| GTN_SetPosCompareFifoMode | 使能读取位置锁存位置并设置存储锁存位置的 FIFO 模式。 |
| GTN_GetPosCompareFifoMode | 获取位置比较锁存比较位置的 FIFO 模式。 |
| GTN_GetPosCompareLatch Value | 读取位置比较锁存比较位置的值及 FIFO 状态。 |
| GTN_PosComparePulse | HSIO立即输出1个指定宽度的脉冲或者反转输出电平 |
| GTN_SetPosComparePulseCount | 设置HSIO立即输出的脉冲个数。 |
| GTN_SetPosComparePulseDuty | 设置HSIO输出的脉冲占空比。 |
| GTN_PosComparePulseEx | HSIO立即输出多个设定脉冲的形态或者反转输出电平。 |
| GTN_GetPosComparePulseStatus | 获取HSIO口立即输出一串脉冲的状态。 |
| GTN_SetHsoPulsePrm | 设置GNM002模块上，HSO输出脉冲参数。 |
| GTN_GetHsoPulsePrm | 读取GNM002模块上，HSO输出脉冲参数。 |
| GTN_BufPosComparePulse | 缓冲区中操作HSIO输出1个指定宽度的脉冲或者反转输出电平 |
| GTN_BufPosComparePulseEx | 前瞻缓冲区中操作HSIO输出1个指定宽度的脉冲或者反转输出电平 |
| GTN_SetPosComparePsoSyncPrm | PSO重频功能，设置重频模式参数 |
| 激光补充指令 | |
| GTN_GetLaserOnOffCount | 读取激光开关次数。 |
| GTN_ClearLaserOnOffCount | 清除记录激光开关次数。 |
| 压力补偿 | |
| GTN_SetAxisPressCompensate | 设置轴压力补偿（位置环，压力补偿到轴规划位置）。 |
| GTN_GetAxisPressCompensate | 获取压力补偿参数。 |
| GTN_SetAxisPressCompensateTable | 设置压力补偿。 |
| GTN_SelectAxisPressCompensateTable | 选择补偿表。 可以设置多个表格的数据，通过该指令选择哪张表格生效。 |
| Trigger 触发重新规划运动 | |
| GTN_SetTriggerProfilePrm | 设置运动遇到捕获时重新规划。 |
| GTN_GetTriggerProfileStatus | 获取捕获触发重新规划的状态。 |
| 限位开关 | |
| GTN_LmtsOnEx | 开启控制器限位 |
| GTN_LmtsOffEx | 关闭控制器限位 |
| 高精度当量变换 | |
| GTN_SetProfileScale | 设置规划器当量值（long 型）。 |
| GTN_GetProfileScale | 获取规划器当量值（long 型）。 |
| GTN_SetEncoderScale | 设置编码器当量值（long 型）。 |
| GTN_GetEncoderScale | 获取编码器当量值（long 型）。 |
| 摩擦力补偿 | |

补充指令与补充说明

| | |
|-------------------------------|---------------------------|
| GTN_SetFriction | 设置摩擦力补偿参数。 |
| GTN_GetFriction | 获取摩擦力补偿参数。 |
| GTN_EnableFriction | 使能摩擦力补偿。 |
| GTN_DisableFriction | 关闭摩擦力补偿。 |
| 高速读 | |
| GTN_LoadReadHsConfig | 配置高速读取指令 |
| GTN_ReadHsOn | 使能或关闭高速读取指令 |
| GTN_SetAxisArriveMode | 配置电机到位模式 |
| GTN_GetStsEx | 获取轴状态 |
| 功能复位选择设置 | |
| GTN_SetResetMode | 调用 GTN_Reset 时，对应功能是否被复位。 |
| 读取绝对式编码器值 | |
| GTN_RN_GetAbsEncPosEx | 读取绝对式编码器位置。 |
| GTN_RN_SetEncMultiLinesEx | 设置绝对式编码器多圈线数。 |
| 脉冲当量不一致的 PSO 功能 | |
| GTN_SetPosComparePsoPrmEx | 设置等间距位置比较输出参数。 |
| GTN_GetPosComparePsoPrmEx | 获取等间距位置比较输出参数。 |
| 插补轮廓误差控制 | |
| GTN_SetCrdContourErrorControl | 设置插补轮廓误差控制。 |
| 同时设置/读取多轴参数相关指令 | |
| GTN_SetPosArray | 设置目标位置。 |
| GTN_GetPosArray | 读取目标位置。 |
| GTN_SetVelArray | 读取目标速度。 |
| GTN_SetTrapPrmArray | 设置 Trap 运动参数。 |
| GTN_SetTriggerArray | 设置捕获参数。 |
| GTN_ClearTriggerStatusArray | 清除捕获状态。 |
| 检查网络结构是否和配置文件中一致 | |
| GTN_CheckRingNetStructure | 检查网络结构是否和配置文件中一致。 |
| 驱动器功能 | |
| GTN_GetMotionMode | 获取驱动器环路模式 |
| GTN_SetMotionMode | 设置驱动器环路模式 |
| GTN_SetDac | 设置驱动器周期同步速度模式时的目标速度 |
| GTN_SetDrvPrfVel | 设置驱动器周期同步速度模式时的目标速度 |
| GTN_SetPrfTorque | 设置驱动器周期同步力矩模式时的目标力矩 |
| GTN_GetAtlTorque | 读取驱动器周期同步力矩模式时的目标力矩 |
| GTN_GetDriverSts | 获取驱动器状态 |
| GTN_SetServoPosLoopPid | 设置驱动器位置环系数 |
| GTN_GetServoPosLoopPid | 读取驱动器位置环系数 |
| GTN_RN_SetTorqueLimit | 设置 GSHD 驱动器力矩限制 |
| GTN_RN_GetTorqueLimit | 获取 GSHD 驱动器力矩限制 |

补充指令与补充说明

| | |
|------------------------------|---|
| GTN_RN_GetServoAlarmInfo | 获取驱动器报警信息 |
| GTN_ReadServoParamInfo | 读取 GSHD 驱动器参数 |
| 回零功能 | |
| GTN_GoHome | 增加驱动器回零模式 |
| Trigger 新增功能 | |
| GTN_SetTriggerEx | 增加设置Trigger捕获沿的方式，sence = 2，或者sence = 3，并启动捕获 |
| GTN_GetTriggerStatusEx | 读取Trigger捕获状态 |
| 缓冲区等待功能 | |
| GTN_BufWaitDi | 设置缓冲区等待外部输入信号。 |
| GTN_BufWaitLongVar | 设置缓存区等待long型变量。 |
| GTN_GetBufWaitDiStatus | 读取缓存区等待外部输入信号的状态。 |
| GTN_GetBufWaitLongVarStatus | 读取缓存区等待变量的状态。 |
| GTN_ClearBufWaitStatus | 清除缓存区等待的状态。 |
| GTN_SetLongVar | 设置内部long型变量的值。 |
| GTN_GetLongVar | 读取内部long型变量的值。 |
| 椭圆插补功能 | |
| GTN_Ellipse | 椭圆插补指令（标准版本不支持）。 |
| GTN_EllipseEx | 椭圆插补前瞻指令（标准版本不支持）。 |
| 振镜功能 | |
| GTN_ScanBufLaserDelayLong | 设置振镜缓存区激光延时（32位接口）。 |
| GTN_SetScanAlarmAutoStopMode | 设置振镜报警自动停止模式(报警信息通过振镜通讯协议从振镜反馈) |
| GTN_GetScanAlarmAutoStopMode | 读取振镜报警自动停止模式(报警信息通过振镜通讯协议从振镜反馈) |
| GTN_GetScanErrorCode | 读取振镜报警信息(报警信息通过振镜通讯协议从振镜反馈) |
| 52 开卡模式读取物理站号和轴号 | |
| GTN_GetResPhyInfo | 输入轴号，获取该轴所在的物理站号和轴号 |
| 运控模式配置功能 | |
| GTN_SetMcMode | 配置运控模式。 |
| GTN_GetMcMode | 读取运控模式。 |
| 前瞻功能 | |
| GTN_SetRadiusRatioTableLa | 设置前瞻曲率参数表。 |
| 点位运动新增功能 | |
| GTN_GetTrapSts | 读取点位运动到位标志。 |
| GTN_ClearTrapSts | 清除点位运动到位标志。 |
| 手轮导引功能 | |
| GTN_SetCrdMPGMode | 设置手轮导引功能参数。 |
| GTN_GetCrdMPGMode | 读取手轮导引功能参数。 |
| GTN_SetCrdMPGModeEx | 设置手轮导引功能参数(增强版，仅R688支持)。 |

补充指令与补充说明

| | |
|------------------------------------|-----------------------------|
| GTN_GetCrdMPGModeEx | 读取手轮导引功能参数(增强版, 仅R688支持)。 |
| Event-Task 新增功能 | |
| GTN_AddTask | 设置事件触发的任务。(新增保存运控变量任务) |
| GTN_GetTaskSaveMcVarResult | 读取保存运控变量任务已保存的值。 |
| 编码器相关配置功能 | |
| GTN_SetEncoderMapRelation | 设置编码器之间的映射关系 |
| GTN_GetEncoderMapRelation | 读取编码器之间的映射关系 |
| GTN_SetEncoderDeltaLimit | 设置编码器增量限制 |
| GTN_GetEncoderDeltaLimit | 读取编码器增量限制 |
| 补偿功能 | |
| GTN_SetLeadScrewCrossComp | 加载交叉误差补偿表 |
| GTN_EnableLeadScrewCrossComp | 开启或关闭交叉误差补偿功能 |
| GTN_SetTransformOrthogonal | 设置平面坐标系非正交转换功能参 |
| GTN_GetTransformOrthogonal | 获取平面坐标系非正交转换功能参 |
| GTN_GetTransformOrthogonalPosition | 获取平面坐标系非正交转换后的各轴位 |
| 平滑功能指令 | |
| GTN_SetStopSmoothTime | 设置异常停止的平滑时间 |
| GTN_GetStopSmoothTime | 获取异常停止的平滑时间 |
| GTN_SetAxisMotionSmooth | 设置轴的平滑时间和平滑系数。 |
| GTN_GetAxisMotionSmooth | 读取轴的平滑时间和平滑系数。 |
| GTN_SetCrdJerkTime | 设置插补运动的平滑时间和平滑系数。 |
| GTN_GetCrdJerkTime | 读取插补运动的平滑时间和平滑系数。 |
| 模块断线状态及安全模式设置 | |
| GTN_RN_GetStationOfflineCount | 读取从站网络连接状态 |
| GTN_RN_SetStationSafeModeOut | 设置从站物理资源的安全模式 |
| GTN_RN_ClearStationSafeModeStatus | 手动清除从站安全模式状态 |
| GTN_RN_SetStationSafeModeControl | 从站安全模式总开关及安全模式清除类型 |
| GTN_RN_IlinkSetSafeModeOut | 设置扩展模块物理资源安全模式 |
| GTN_RN_IlinkClearSafeModeStatus | 扩展模块手动清除从站安全模式状态 |
| GTN_RN_IlinkSetSafeModeControl | 设置扩展模块物理资源安全模式的总开关及安全模式清除类型 |
| 网络恢复指令 | |
| GTN_RN_Recover | 网络恢复指令 |
| 串行通讯指令 (本地 485/232 通信指令) | |
| GT_RN_ComOpen | 打开FPGA转串口模块。 |
| GT_RN_ComClose | 关闭FPGA转串口模块。 |
| GT_RN_ComRead | 从串口读取数据。 |
| GT_RN_ComWrite | 从串口发送数据。 |
| GT_RN_ComGetState | 获取串口状态。 |
| GT_RN_ComSetSettings | 设置串口参数。 |
| GT_RN_ComClearErr | 清除串口状态。 |

补充指令与补充说明

| | |
|--------------------------------|----------------|
| GT_RN_ComSetMode | 设置串口电平模式。 |
| 串行通讯指令（通用 485/232 通信指令） | |
| GTN_RN_SerialComOpen | 打开FPGA转串口模块。 |
| GTN_RN_SerialComClose | 关闭FPGA转串口模块。 |
| GTN_RN_SerialComRead | 从串口读取数据。 |
| GTN_RN_SerialComWrite | 从串口发送数据。 |
| GTN_RN_SerialComGetState | 从串口发送数据。 |
| GTN_RN_SerialComSetSettings | 设置串口参数。 |
| GTN_RN_SerialComClearErr | 设置串口参数。 |
| GTN_RN_SerialComSetMode | 设置串口电平模式。 |
| GTN_RN_SerialComGetRecvFifoCnt | 获取串口接收缓冲区数据数量。 |

四、指令详细说明

1. 位置比较功能

指令 1 GTN_SetPosCompareFifoMode

| | |
|-------|--|
| 指令原型 | short GTN_SetPosComapreFifoMode(short core,short index,short mode) |
| 指令说明 | 使能读取位置锁存位置并设置存储锁存位置的 FIFO 模式。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| index | 位置比较的通道号。 |
| mode | FIFO 模式，0-静态模式，缓冲区满了就不再接收新数据，1-循环模式，缓冲区满后从头开始存储。 FIFO 大小为 2048。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 2 GTN_GetPosCompareFifoMode

| | |
|--------|---|
| 指令原型 | short GTN_GetPosCompareFifoMode(short core,short index,short *pMode); |
| 指令说明 | 获取位置比较锁存比较位置的 FIFO 模式。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| index | 位置比较的通道号。 |
| *pMode | FIFO 模式。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 3 GTN_GetPosCompareLatchValue

| | |
|--------|---|
| 指令原型 | short GTN_GetPosCompareLatchValue(short core,short index,long count,long *pDataX,long *pDataY,long *pCount,TLatchValueInfo *pInfo); |
| 指令说明 | 读取位置比较锁存比较位置的值及 FIFO 状态。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 7 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| index | 位置比较的通道号。 |
| count | 希望读取多少个比较锁存位置。 |
| pDataX | 存储读取比较锁存位置 X 方向值的数组，数组长度应该大于或等于 count 值。 |

| | |
|--------|--|
| pDataY | 存储读取比较锁存位置 Y 方向值的数组，数组长度应该大于或等于 count 值。 |
| pCount | 实际读取的个数，从 pDataX 和 pDataY 读取比较锁存位置以此个数为准。 |
| pInfo | 存储比较锁存位置的 FIFO 的状态。 typedef struct { short fifoFull; //0-缓冲区未滿，-缓冲区满了 short pad1[3]; double pad2[2]; }TLatchValueInfo; |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 4 GTN_PosComparePulse

| | | | |
|------------------|---|------|---|
| 指令原型 | short GTN_PosComparePulse (short core,short index,short outputMode,short level,unsigned short outputPulseWidth); | | |
| 指令说明 | HSIO立即输出1个设定宽度的脉冲或者反转输出电平。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 5 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| outputMode | 输出模式，0-脉冲，1-电平。 | | |
| level | 输出模式为脉冲，则该参数无效，输出脉冲为电平则该参数表示电平高低，0-低电平，1-高电平。 | | |
| outputPulseWidth | 输出模式为脉冲有效，表示输出脉冲的宽度，单位 us。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | <p>一、设置第一路硬件输出口输出一个脉冲，该输出口控制权为第一套位置比较</p> <pre>rtn = GTN_SetTerminalPermitEx(1,1,MC_HSO,0x2,1);//设置站 1 硬件上的第一路输出口控制权为第一套位置比较输出口,标红 0x2 表示第一套位置比较。</pre> <pre>rtn = GTN_PosComparePulse(1,1,0,0,100); //第一套位置比较立即输出 100us 脉宽的脉冲，输出口为站 1 硬件上的第一路输出口，标红 1 表示第一套位置比较。</pre> <p>二、设置第二路硬件输出口输出一个脉冲，该输出口控制权为第一套位置比较</p> <pre>rtn = GTN_SetTerminalPermitEx(1,1,MC_HSO,0x2,2,1);//设置站 1 硬件上的第二路输出口控制权为第一套位置比较输出口，标红 0x2 表示第一套位置比较。</pre> <pre>rtn = GTN_PosComparePulse(1,2,0,0,100); //第一套位置比较立即输出 100us 脉宽的脉冲，输出口为站 1 硬件上的第二路输出口，标红 1 表示第一套位置比较。</pre> <p>三、设置第一路硬件输出口输出一个脉冲，该输出口控制权为第二套位置比较</p> <pre>rtn = GTN_SetTerminalPermitEx(1,1,MC_HSO,0x4,1,1);//设置站 1 硬件上的第一路输出口控制权为第二套位置比较输出口，标红 0x4 表示第二套位置比较。</pre> <pre>rtn = GTN_PosComparePulse(1,2,0,0,100); //第二套位置比较立即输出 100us 脉宽的脉冲，输出口为站 1 硬件上的第一路输出口，标红 2 表示第二套位置比较。</pre> | | |

四、设置第二路硬件输出口输出一个脉冲，该输出口控制权为第二套位置比较

rtn = GTN_SetTerminalPermitEx(1,1,MC_HSO,0x4,2,1); //设置站 1 硬件上的第二路输出口控制权为第二套位置比较输出口，标红 0x4 表示第二套位置比较。

rtn = GTN_PosComparePulse(1,2,0,0,100); //第二套位置比较立即输出 100us 脉宽的脉冲，输出口为站 1 硬件上的第二路输出口，标红 2 表示第二套位置比较。

指令 5 GTN_SetPosComparePulseCount

| | | | |
|-------|--|------|---|
| 指令原型 | short GTN_SetPosComparePulseCount (short core,short index,long count); | | |
| 指令说明 | 设置HSIO立即输出的脉冲个数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| count | 到达位置比较点或者设置强制输出脉冲时的脉冲输出个数，内部默认该值为 1，取值范围[1,32767]。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

指令 6 GTN_SetPosComparePulseDuty

| | | | |
|-------|---|------|---|
| 指令原型 | short GTN_SetPosComparePulseDuty(short core,short index,double duty); | | |
| 指令说明 | 设置HSIO输出的脉冲的占空比。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| duty | 占空比，取值范围：0~100，单位：% 该参数配合位置比较输出初始化过程中设置的脉宽使用，如脉宽设置为 200 微秒，占空比设置为 30，则输出的脉冲，高电平时间为 60 微秒，低电平时间为 140 微秒 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

指令 7 GTN_PosComparePulseEx

| | | | |
|----------------|---|------|---|
| 指令原型 | short GTN_PosComparePulseEx (short core,short index, TPosComparePulse *pPosComparePulse); | | |
| 指令说明 | HSIO立即输出多个设定脉冲的形态或者反转输出电平。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| pPosComparePul | typedef struct | | |

| | |
|-------|--|
| se | <pre>{ short outputMode; short level; short reserve1[2]; double highLevelTime; double lowLevelTime; double reserve2[4]; }TPosComparePulse;</pre> <p>outputMode: 输出模式, 0-脉冲, 1-电平。 level: 电平模式下, 输出电平, 取值范围[0,1], 脉冲模式下该值无效。 highLevelTime: 脉冲模式下, 脉冲的高电平时间, 单位: us, 取值范围[1,32767], 电平模式该值下无效。 lowLevelTime: 脉冲模式下, 脉冲的低电平时间, 单位: us, 取值范围[1,32767], 电平模式该值下无效。 reserve1、reserve2: 保留参数。</p> <p>【注: 当未调用 GTN_SetPosComparePulseCount 设置脉冲输出个数或者调用 GTN_SetPosComparePulseCount 设置脉冲输出个数设置为 1, 调用该值令时会输出一个脉冲, 其脉冲宽度即为 highLevelTime】</p> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | 参考 GTN_PosComparePulse |

指令 8 GTN_GetPosComparePulseStatus

| | | | |
|------------------------|--|------|---|
| 指令原型 | short GTN_GetPosComparePulseStatus(short core,short index,TPosComparePulseStatus *pPosComparePulseStatus) | | |
| 指令说明 | 获取HSIO口立即输出一串脉冲的状态。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| index | 位置比较索引, 正整数, 取值范围[1,8] | | |
| pPosComparePulseStatus | <pre>typedef struct { long count; short reserve1[2]; double reserve2[4]; }TPosComparePulseStatus;</pre> <p>count: HSIO 口已经输出脉冲的个数。 reserve1、reserve2: 保留参数。</p> | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

指令 9 GTN_SetHsoPulsePrm

| | |
|------|--|
| 指令原型 | short GTN_SetHsoPulsePrm (short core,short station,short hsoIndex,THsoPulsePrm |
|------|--|

| | |
|----------|---|
| 指令说明 | *pPrm,short hsoCount=1); |
| 指令类型 | 设置GNM002模块上，HSO输出脉冲参数。 |
| 指令参数 | 立即指令。 |
| core | 该指令共有 5 个参数，参数的详细信息如下。 |
| station | 内核，正整数，取值范围请参考 指令参数范围 中的“内核”一栏。 |
| hsoIndex | 物理站号。 |
| pPrm | Hso 起始通道号，正整数，取值范围：[1,10]。 |
| hsoCount | Hso 通道输出脉冲参数结构体 数组 |
| 指令返回值 | <pre>typedef struct { short mode; short timeScale; short pad1[2]; double pulseWidth; double pad2[3]; }THsoPulsePrm;</pre> <p>mode: hso 输出模式, 0-不输出, 1-按照位置比较指令设置参数输出, 2-按照该条指令设置值输出。 timeScale: 时间精度, 0-1us, 1-0.1us。 pad1[2]: 保留值。 pulseWidth: 输出脉冲宽度, 单位: us。 pad2[3]: 保留值。</p> <p>重点说明: 当 mode = 2, timeScale = 0, pulseWidth = 5.45 时, 硬件输出脉宽为 5us 的脉冲; 当 mode = 2, timeScale = 1, pulseWidth = 5.45 时, 硬件输出脉宽为 5.4us 的脉冲。 timeScale 表示脉冲输出时的精度, 而不是设置硬件接口的精度。</p> |
| 指令示例 | Hso 通道个数，默认为 1。 |
| 指令示例 | 请参照指令返回值列表。 |
| 指令示例 | <p>a) 设置 GNM001 模块上 12 路 HSO 都输出，通道 1 输出脉宽为 2.5us 的脉冲，通道 2 输出脉宽为 5.0us 的脉冲，通道 3 输出脉宽为 7.5us 的脉冲，...，通道 12 输出脉宽为 30.0us 的脉冲。</p> <pre>THsoPulsePrm pulsePrm[12]; rtn = GTN_GetHsoPulsePrm(core,station,1,pulsePrm,12); if(CMD_SUCCESS != rtn) { return; } for(i=0; i<12; ++i) { pulsePrm[i].mode = 2; pulsePrm[i].timeScale = 1; pulsePrm[i].pulseWidth = 2.5*(i+1); }</pre> |

```

rtn = GTN_SetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}
b) 设置 GNM001 上第一路 HSO 输出脉宽为 2.5us 的脉冲，其他通道不输出。
THsoPulsePrm pulsePrm[12];
rtn = GTN_GetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}

for(i=0; i<12; ++i)
{
    pulsePrm[i].mode = 0;
}

pulsePrm[0].mode = 2;
pulsePrm[0].timeScale = 1;
pulsePrm[0].pulseWidth = 2.5;

rtn = GTN_SetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}
    
```

指令 10 GTN_GetHsoPulsePrm

| | | | |
|----------|--|------|---|
| 指令原型 | short GTN_GetHsoPulsePrm (short core,short station,short hsoIndex,THsoPulsePrm *pPrm,short hsoCount=1); | | |
| 指令说明 | 读取GNM002模块上，HSO输出脉冲参数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 5 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| station | 物理站号。 | | |
| hsoIndex | Hso 起始通道号，正整数，取值范围：[1,10]。 | | |
| pPrm | Hso 通道输出脉冲参数结构体数组 <pre> typedef struct { short mode; short timeScale; short pad1[2]; double pulseWidth; double pad2[3]; }THsoPulsePrm; </pre> | | |

hsoCount
指令返回值

mode: hso 输出模式, 0-不输出, 1-按照位置比较指令设置参数输出, 2-按照该条指令设置值输出。

timeScale: 时间精度, 0-1us, 1-0.1us。

pad1[2]: 保留值。

pulseWidth: 输出脉冲宽度, 单位: us。

pad2[3]: 保留值。

Hso 通道个数, 默认为 1。

请参照指令返回值列表。

c) 设置 GNM001 模块上 12 路 HSO 都输出, 通道 1 输出脉宽为 2.5us 的脉冲, 通道 2 输出脉宽为 5.0us 的脉冲, 通道 3 输出脉宽为 7.5us 的脉冲, ..., 通道 12 输出脉宽为 30.0us 的脉冲。

```
THsoPulsePrm pulsePrm[12];
rtn = GTN_GetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}

for(i=0; i<12; ++i)
{
    pulsePrm[i].mode = 2;
    pulsePrm[i].timeScale = 1;
    pulsePrm[i].pulseWidth = 2.5*(i+1);
}
```

指令示例

```
rtn = GTN_SetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}
```

d) 设置 GNM001 上第一路 HSO 输出脉宽为 2.5us 的脉冲, 其他通道不输出。

```
THsoPulsePrm pulsePrm[12];
rtn = GTN_GetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}

for(i=0; i<12; ++i)
{
    pulsePrm[i].mode = 0;
}

pulsePrm[0].mode = 2;
pulsePrm[0].timeScale = 1;
pulsePrm[0].pulseWidth = 2.5;
```

```

rtn = GTN_SetHsoPulsePrm(core,station,1,pulsePrm,12);
if( CMD_SUCCESS != rtn )
{
    return;
}
    
```

指令 11 GTN_BufPosComparePulse

| | | | |
|------------------|---|------|---|
| 指令原型 | short GTN_BufPosComparePulse(short core, short crd, short index, short outputMode, short level, unsigned short outputPulseWidth, short fifo); | | |
| 指令说明 | 缓冲区中操作HSIO输出1个设定宽度的脉冲或者反转输出电平。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 5 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系索引，取值范围[1,2]。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| outputMode | 输出模式，0-脉冲，1-电平。 | | |
| level | 输出模式为脉冲，则该参数无效，输出脉冲为电平则该参数表示电平高低，0-低电平，1-高电平。 | | |
| outputPulseWidth | 输出模式为脉冲有效，表示输出脉冲的宽度，单位 us。 | | |
| fifo | 插补缓冲区 FIFO，取值范围[0,1]。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

指令 12 GTN_BufPosComparePulseEx

| | | | |
|------------------|---|------|---|
| 指令原型 | short GTN_BufPosComparePulseEx(short core, short crd, short index, short outputMode, short level, unsigned short outputPulseWidth, short fifo); | | |
| 指令说明 | 前瞻缓冲区操作HSIO输出1个设定宽度的脉冲或者反转输出电平。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 5 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系索引，取值范围[1,2]。 | | |
| index | 位置比较索引，正整数，取值范围[1,8] | | |
| outputMode | 输出模式，0-脉冲，1-电平。 | | |
| level | 输出模式为脉冲，则该参数无效，输出脉冲为电平则该参数表示电平高低，0-低电平，1-高电平。 | | |
| outputPulseWidth | 输出模式为脉冲有效，表示输出脉冲的宽度，单位 us。 | | |
| fifo | 插补缓冲区 FIFO，取值范围[0,1]。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

指令 13 GTN_SetPosComparePsoSyncPrm

| | |
|-----------------|---|
| 指令原型 | short GTN_SetPosComparePsoSyncPrm (short core, short posCompareIndex, short psoSyncMode, double frq=100); |
| 指令说明 | PSO重频功能，设置重频模式参数， 只有GNM-403-06模块支持重频模式 |
| 指令类型 | 立即指令。 章节页码 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 |
| posCompareIndex | 位置比较索引，正整数，取值范围[1,8] |
| psoSyncMode | <p>设置重频模式</p> <p>外部重频模式： POS_COMPARE_MODE_PSO_SYNC_EXTERNAL_SIGNAL (1)</p> <p>内部重频模式： POS_COMPARE_MODE_PSO_SYNC_INTERNAL_SIGNAL (2)</p> <p>外部重频模式下，需要接入外部重频信号 GNM-403-06 模块（和 GNM-403 模块外观相同）使用 MPG 接口的第 10 路和第 14 路引脚接入外部重频信号 第 10 路引脚 MPGA- 接入外部重频信号负向 第 14 路引脚 MPGA+ 接入外部重频信号正向</p> <p>(5) 手轮接口</p>  |
| frq | 该参数仅在内部重频模式下有效，表示重频频率，单位：kHz，默认是 100kHz |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | 例程 5-1：重频模式例程 |

图 3-12 模块 MPG 接口引脚号说明

模块手轮 MPG 接口有 1 路辅助编码器输入（接受 A 相和 B 相差分输入（5V 电平），7 路数字量 IO 输入（24V）。接口定义如下表：

表 3-13 MPG 接口定义

| 引脚 | 信号 | 说明 | 引脚 | 信号 | 说明 |
|----|----------|--------------------------|----|--------------|--------------------------|
| 1 | OGND | 24V 电源地 | 9 | MPGB- | 编码器输入 B 负向 |
| 2 | MPGI2 | 数字量输入 | 10 | MPGA- | 编码器输入 A 负向 |
| 3 | MPGI0 | 数字量输入 | 11 | MPGI6 | 数字量输入 |
| 4 | MPGB+ | 编码器输入 B 正向 | 12 | MPGI5 | 数字量输入 |
| 5 | OGND/GND | 外部/内部电源地 ^(注1) | 13 | MPGI4 | 数字量输入 |
| 6 | OVCC | 24V 电源 | 14 | MPGA+ | 编码器输入 A 正向 |
| 7 | MPGI3 | 数字量输入 | 15 | OVCC5V/VCC | 5V/5V 电源 ^(注1) |
| 8 | MPGI1 | 数字量输入 | | | |

2. 激光补充指令

指令 14 GTN_GetLaserOnOffCount

| | |
|-------------|--|
| 指令原型 | short GTN_GetLaserOnOffCount(short core,short channel,TLaserOnOffCount *pLaserCount) |
| 指令说明 | 读取激光开关次数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| channel | 激光通道号。 |
| pLaserCount | typedef struct { unsigned long onCount; //激光开的次数 unsigned long offCount; //激光关的次数 unsigned long onCountInFpga; //暂无用 unsigned long offCountInFpga; //暂无用 unsigned long pad[4]; //保留 }TLaserOnOffCount; 读取激光开关次数的结构体指针。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_ClearLaserOnOffCount |
| 指令示例 | |

指令 15 GTN_ClearLaserOnOffCount

| | |
|---------|--|
| 指令原型 | short GTN_ClearLaserOnOffCount(short core,short channel) |
| 指令说明 | 清除记录激光开关次数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| channel | 激光通道号。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_GetLaserOnOffCount |
| 指令示例 | |

3. 压力补偿

指令 16 GTN_SetAxisPressCompensate

| | |
|------|--|
| 指令原型 | short GTN_SetAxisPressCompensate(short core,short axis,TAxisPressCompensate *pPressComp) |
| 指令说明 | 设置轴压力补偿（位置环，压力补偿到轴规划位置）。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |

| | |
|------------|---|
| axis | 要叠加到的规划轴。 |
| pPressComp | <pre>typedef struct { double kp; double ki; double kd; double integralLimit; //积分极限 double derivativeLimit; //微分极限 double limit; //总极限 }TAxisPressPid; typedef struct { short enable; //是否使能, -不使能, -使能 short type; //输入类型, 模拟量 (MC_ADC, MC_AU_ADC) //或者网络模块 short index; //输入的索引, 如果是ADC就是ADC的索引号, 从开始 short mode; //0-线性模式, -查表模式, 需要设置表格 double target; //目标值, 单位: 伏特或牛顿 double threshold; //阈值, 当前值超过阈值才开始补偿, 单位: 伏特或牛顿 double deadZone; //死区, 当前值和目标值的差在死区内不补偿, 防止振荡, //单位: 伏特或牛顿 double factor; //力和位移的转换系数 TAxisPressPid pid; //调节补偿的PID }TCombineAdc;</pre> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 17 GTN_GetAxisPressCompensate

| | |
|------------|---|
| 指令原型 | short GTN_GetAxisPressCompensate(short core,short axis, TAxisPressCompensate *pPressComp) |
| 指令说明 | 获取压力补偿参数。 |
| 指令类型 | 立即指令, 调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 |
| core | 控制器核号 |
| axis | 规划轴。 |
| pPressComp | 获取的参数值。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 18 GTN_SetAxisPressCompensateTable

| | |
|------------|---|
| 指令原型 | short GTN_SetAxisPressCompensateTable(short core, short axis, short index,long count, double *pPressData, Endouble *pPosData) |
| 指令说明 | 设置压力补偿。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| axis | 规划轴。 |
| index | 表格索引，从 1 开始，目前支持 2 个。 |
| count | 传入的数据量，目前最大 2048。 |
| pPressData | 压力数据，需要单调递增。 |
| pPosData | 位置数据，单位：脉冲。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 19 GTN_SelectAxisPressCompensateTable

| | |
|-------|---|
| 指令原型 | short GTN_SelectAxisPressCompensateTable(short core,short axis,short index) |
| 指令说明 | 选择补偿表。GTN_SetAxisPressCompensateTable 可以设置多个表格的数据，通过该指令选择哪张表格生效。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| axis | 规划轴。 |
| index | 表格索引，从 1 开始，目前支持 2 个。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

4. Trigger 触发重新规划运动

指令 20 GTN_SetTriggerProfilePrm

| | |
|---------|---|
| 指令原型 | short GTN_SetTriggerProfilePrm(short core,short profile,TTriggerProfilePrm *pPrm) |
| 指令说明 | 设置运动遇到捕获时重新规划。重新规划以捕获位置+设定的运动偏移量作为终点规划位置。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 控制器核号 |
| profile | 规划器的编号。 |
| pPrm | 规划的参数 typedef struct { short mode; //运动类型 |

补充指令与补充说明

| | |
|-------|---|
| | <pre> short enable; //是否使能, -不使能, -使能 short trigger; //trigger索引, 取值从开始 short pad1; //保留 long distance; //触发时偏移量, 非负数, 单位: 脉冲 long posLimit; //重新规划后, 触发位置+偏移量不能超过该值, 可正可负, 单位: 脉冲 double vel; //目标速度, 重新规划的目标速度(绝对值), 非负数, 单位: 脉冲/ms double acc; //需要提速时的加速度, 单位: 脉冲/ms double dec; //运动到触发偏移量的减速度, 单位: 脉冲/ms double percent; //减速段S型曲线时间百分比, 例如表示% double reserve[4]; }TTriggerProfilePrm; </pre> |
| 指令返回值 | mode的取值: <pre> #define PROFILE_MODE_TRAP (0) //点位运动 #define PROFILE_MODE_PVT (6) //PVT 运动 </pre> |
| 相关指令 | 请参照指令返回值列表。 |
| 指令示例 | |

指令 21 GTN_GetTriggerProfileStatus

| | |
|---------|---|
| 指令原型 | short GTN_GetTriggerProfileStatus(short core, short profile, TTriggerProfileStatus *pSts) |
| 指令说明 | 获取捕获触发重新规划的状态。 |
| 指令类型 | 立即指令, 调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 |
| core | 控制器核号 |
| profile | 规划器的编号。 |
| pSts | 状态值 |
| | <pre> typedef struct { short mode; short enable; short execute; //是否执行中, 0-未执行, -执行 short status; //执行过程中的状态, 正常为, 异常则返回错误码 long endPos; //终点位置(捕获+偏移量) long reserve[7]; }TTriggerProfileStatus; </pre> |
| | status的异常错误码: <pre> //设置的目标速度小于当前运动速度 #define TRIGGER_PROFILE_STATUS_ERROR_END_POS (1) //捕获到的位置或者设置的位置异常 #define TRIGGER_PROFILE_STATUS_ERROR_VEL (2) </pre> |

指令返回值
相关指令

请参照指令返回值列表。

```
short rtn;
short core=1;
short profile=1;
TTriggerEx triggerPrm;
bool ok;
bool okFinal = true;
short index=1;

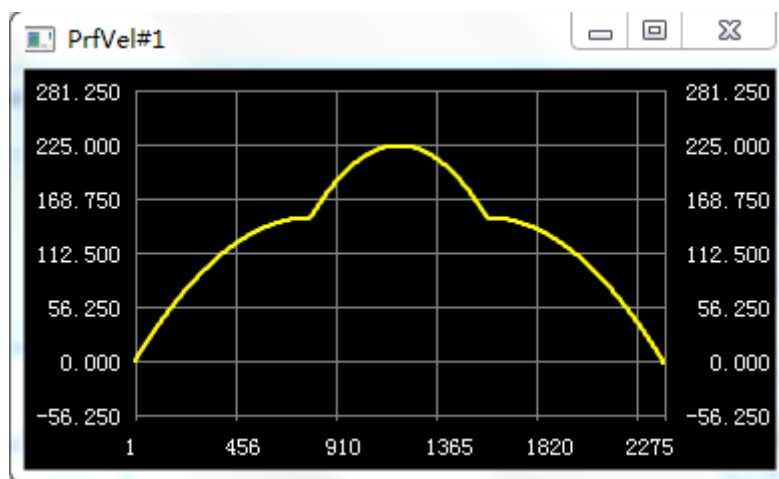
triggerPrm.latchType = MC_ENCODER;
triggerPrm.latchIndex = 1;
triggerPrm.probeType = 1;
triggerPrm.probeIndex = 1;
triggerPrm.sense = 1;
triggerPrm.offset = 0;
triggerPrm.loop = 100;
triggerPrm.windowOnly = 1;           //设置监测窗
triggerPrm.firstPosition = 150000;
triggerPrm.lastPosition = 250000;
rtn = GTN_SetTriggerEx(core,1,&triggerPrm); //设置Trigger
```

指令示例

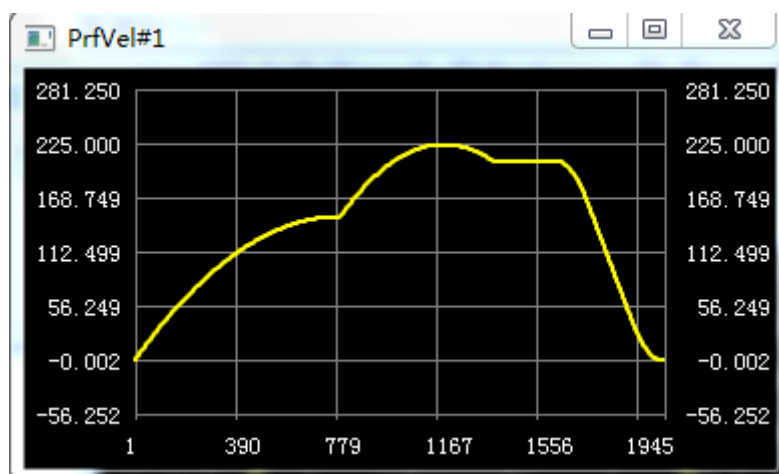
```
double timeArray[]={0,400,800,1200};
double posArray[]={0,80000,240000,320000};
double velArray[]={0,300,300,0};
rtn = GTN_PrFpvt(core,profile);
rtn = GTN_PvtTable(core,profile,4,&timeArray[0],&posArray[0],&velArray[0]);
rtn = GTN_PvtTableSelect(core,profile,profile);
rtn = GTN_PvtStart(core,1<<(profile-1));
```

```
TTriggerProfilePrm triggerPrfPrm;
triggerPrfPrm.type = 6;
triggerPrfPrm.enable = 1;
triggerPrfPrm.trigger = 1;
triggerPrfPrm.distance = 100000;
triggerPrfPrm.posLimit = 300000;
triggerPrfPrm.vel = 418;
triggerPrfPrm.acc = 2;
triggerPrfPrm.dec = 2;
triggerPrfPrm.percent = 0;
rtn = GTN_SetTriggerProfilePrm(core,profile,&triggerPrfPrm);
```

下图是使能捕获触发重新规划功能前的 PVT 运动：



下图是规划位置在 200000 左右时触发捕获后重新规划的 PVT 运动：



5. 限位开关

指令 22 GTN_LmtsOnEx

| | |
|-----------|---|
| 指令原型 | short GTN_LmtsOnEx (short core,short axis,short limitType=-1,short limitMode=-1); |
| 指令说明 | 开启控制器限位 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 内核，正整数，GEN 8,16,32 版本取值范围 1，GEN 64 版本取值范围 1-2 |
| axis | 轴号，正整数 |
| limitType | MC_LIMIT_POSITIVE (0) : 正限位 |
| | MC_LIMIT_NEGATIVE (1) : 负限位 |
| | -1 : 正限位和负限位 |
| limitMode | LIMIT_MODE_EXTERMAL (0) : 硬限位 |
| | LIMIT_MODE_SOFT (1) : 软限位 |
| | -1 : 硬限位和软限位 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 23 GTN_LmtsOffEx

| | |
|-----------|--|
| 指令原型 | short GTN_LmtsOffEx (short core,short axis,short limitType=-1,short limitMode=-1); |
| 指令说明 | 关闭控制器限位 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 内核，正整数，GEN 8,16,32 版本取值范围 1，GEN 64 版本取值范围 1-2 |
| axis | 轴号，正整数 |
| limitType | MC_LIMIT_POSITIVE : 正限位 |
| | MC_LIMIT_NEGATIVE : 负限位 -1 : 正限位和负限位 |
| limitMode | LIMIT_MODE_EXTERMAL : 硬限位 |
| | LIMIT_MODE_SOFT : 软限位 -1 : 硬限位和软限位 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

6. 高精度当量变换

指令 24 GTN_SetProfileScale

| | |
|-------|--|
| 指令原型 | short GTN_SetProfileScale (short core,short axis,long alpha,long beta) |
| 指令说明 | 设置规划器当量值 (long 型)。该指令可以与 GT_ProfileScale 同时使用。最终的当量值为两条指令设置的当量值的乘积。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号。 |
| axis | 控制器轴号。 |
| alpha | 规划器当量的 alpha 值。 |
| beta | 规划器当量的 beta 值。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 25 GTN_GetProfileScale

| | |
|--------|---|
| 指令原型 | short GTN_GetProfileScale(short core,short axis,long *pAlpha,long *pBeta) |
| 指令说明 | 获取规划器当量值 (long 型)。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号。 |
| axis | 控制器轴号。 |
| pAlpha | 规划器当量的 alpha 值。 |
| pBeta | 规划器当量的 beta 值。 |

| | |
|-------|--|
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | <pre> rtn = GT_SetProfileScale(1,10,1); //执行点位运动，目标位置 10000 则执行结果为：规划位置=10000，编码器位置=1000 如果也设置编码器当量 rtn = GT_SetEncoderScale(1,1,10); 则执行结果为：规划位置=10000，编码器位置=10000 </pre> |

指令 26 GTN_SetEncoderScale

| | |
|---------|---|
| 指令原型 | short GTN_SetEncoderScale (short core,short encoder,long alpha,long beta) |
| 指令说明 | 设置编码器当量值（long 型）。该指令可以与 GT_ProfileScale 同时使用。最终的当量值为两条指令设置的当量值的乘积。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号。 |
| encoder | 编码器号，取值范围[1,8]。 |
| alpha | 编码器当量的 alpha 值。 |
| beta | 编码器当量的 beta 值。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 27 GTN_GetEncoderScale

| | |
|---------|--|
| 指令原型 | short GTN_GetEncoderScale(short core,short encoder,long *pAlpha,long *pBeta) |
| 指令说明 | 获取编码器当量值（long 型）。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号。 |
| encoder | 编码器号。 |
| pAlpha | 编码器当量的 alpha 值。 |
| pBeta | 编码器当量的 beta 值。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

7. 摩擦力补偿

指令 28 GTN_SetFriction

| | |
|------|---|
| 指令原型 | short GTN_SetFriction(short core,short axis,short gain,double compTime) |
| 指令说明 | 设置摩擦力补偿参数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 核号。 |

| | |
|----------|--------------------------------------|
| axis | 控制器轴号。 |
| gain | 补偿 DAC 值，-32768 对应-10V，32767 对应+10V。 |
| compTime | 补偿时间，单位 ms。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 29 GTN_GetFriction

| | |
|-----------|---|
| 指令原型 | short GTN_GetFriction(short core,short axis,short *pGain,double *pCompTime) |
| 指令说明 | 获取摩擦力补偿参数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 核号。 |
| axis | 控制器轴号。 |
| pGain | 获取的补偿值。 |
| pCompTime | 获取的补偿时间。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 30 GTN_EnableFriction

| | |
|-------|---|
| 指令原型 | short GTN_EnableFriction(short core,short axis) |
| 指令说明 | 使能摩擦力补偿。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 |
| core | 核号。 |
| axis | 控制器轴号。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

指令 31 GTN_DisableFriction

| | |
|-------|--|
| 指令原型 | short GTN_DisableFriction(short core,short axis) |
| 指令说明 | 关闭摩擦力补偿。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 |
| core | 核号。 |
| axis | 控制器轴号。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | |

8. 高速读

指令 32 GTN_LoadReadHsConfig

| | | |
|------------|--|---|
| 指令原型 | short GTN_LoadReadHsConfig (short core,char *pFile) | |
| 指令说明 | 配置高速读取指令 | |
| 指令类型 | 立即指令，调用后立即生效。 | |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 | |
| core | 核号，正整数，取值范围[1,2] | |
| pFile | INI 格式的配置文件，最多支持加载 64 个变量，配置文件中若大于 64 个变量只有前 64 个变量会生效。同时返回错误值 1 目前支持的指令包括： | |
| | 指令 | INI 文件 section 的 type |
| | GTN_GetSts | WATCH_VAR_AXIS_STATUS |
| | GTN_GetPrfPos | 6003 |
| | GTN_GetEncPos | WATCH_VAR_ENC_POS |
| | GTN_GetDiEx | WATCH_VAR_BANK_GPI |
| | GTN_GetDiBit | WATCH_VAR_BANK_HOME_DI WATCH_VAR_BANK_ALARM_DI WATCH_VAR_BANK_POSITIVE_LIMIT_DI WATCH_VAR_BANK_NEGATIVE_LIMIT_DI WATCH_VAR_BANK_ARRIVE_DI |
| | GTN_GetStsEx | WATCH_VAR_AXIS_STATUS_EX |
| type 宏对应的值 | 20100 | |
| | 6003 | |
| | 30000 | |
| | 31060 | |
| | 31061 | |
| | 31062 | |
| | 31063 | |
| | 31064 | |
| | 31065 | |
| | | |
| | 20101 | |
| 指令返回值 | 请参照指令返回值列表。 | |
| 相关指令 | GTN_ReadHsOn | |
| 指令示例 | INI 文件格式示例 1: | |
| | <pre>[var1] type=6003 index=1 [var2] type=6003 index=2 [var3] type=WATCH_VAR_AXIS_STATUS index=1</pre> | |
| | <p>上述配置了轴 1 和轴 2，2 个规划位置加速，那么通过下面指令都可以获得加速</p> <pre>rtn = GTN_GetPrfPos(1,1,prfPosBak,2); //连续获取的数量都要在配置文件配置的加速项范围内</pre> <p>或者</p> <pre>rtn = GTN_GetPrfPos(1,1,&EncPos1); rtn = GTN_GetPrfPos(1,2,&EncPos2);</pre> | |

指令 33 GTN_ReadHsOn

| | |
|----------|--|
| 指令原型 | short GTN_ReadHsOn(short core,short enable,short mode,double interval) |
| 指令说明 | 使能或关闭高速读取指令 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| enable | 是否使能，0-不使能，1-使能 |
| mode | 2-寄存器模式 |
| interval | 刷新周期，0-按照每个规划周期刷新，N-N 个规划周期刷新一次 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 34 GTN_SetAxisArriveMode

| | |
|-------|---|
| 指令原型 | short GTN_SetAxisArriveMode(short core,short axis,TAxisArrivePrm *pPrm) |
| 指令说明 | 配置电机到位模式 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| axis | 轴号 |
| pPrm | <pre>typedefstruct { short mode; //模式，0-默认模式，1-仅判断一次进入误差带，2-规划到位，3-脉冲计算器到位 short pad0; long band; //控制器判断到位误差带 long time; //控制器判断到位时间 long pad1[2]; }TAxisArrivePrm;</pre> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

指令 35 GTN_GetStsEx

| | |
|------|---|
| 指令原型 | short GTN_GetStsEx(short core,short axis,long *pSts,short count=1,unsigned long *pClock=NULL) |
| 指令说明 | 获取轴状态 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| axis | 轴号 |

| | |
|-------|--|
| pSts | 按位标示轴状态，除了bit12为INP信号外，其他bit位与GTN_GetSts一致。INP信号为驱动器的电机到位信号。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | |

9. 功能复位选择设置

指令 36 GTN_SetResetMode

| | |
|-----------|---|
| 指令原型 | GT_API GTN_SetResetMode(short core,short type,TResetModePrmUnion *pResetPrm) |
| 指令说明 | 调用 GTN_Reset 时，对应功能是否被复位。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| tyoe | 复位功能类型 RESET_TYPE_NONE // type 为该类型时，全部功能复位到初始状态 RESET_TYPE_IO // type 为该类型时，IO 功能复位到根据设置参数进行复位 |
| pResetPrm | 复位功能参数 typedef struct { short mode; // mode为1表示保持，为0表示复位到初始状态 }TResetIo; typedef union { TResetIo resetIo; double data[7]; }TResetModePrmUnion; |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_Reset |
| 指令示例 | <pre> TResetModePrmUnion resetPrm; short resetType=RESET_TYPE_IO; resetPrm.resetIo.mode = 1; // mode为1，表示调用GTN_Reset时IO保持复位前的状态 rtn = GTN_SetResetMode(1,resetType,&resetPrm); rtn = GTN_Reset(1); </pre> |

10. 读取绝对式编码器值

指令 37 GTN_RN_GetAbsEncPosEx

| | |
|---------|--|
| 指令原型 | GT_API GTN_RN_GetAbsEncPosEx(short core,short encoder, long long *pEncPos) |
| 指令说明 | 读取绝对式编码器位置。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| encoder | 第几路编码器 |

| | |
|---------|---|
| pEncPos | 读取的绝对式编码器位置 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_RN_SetEncMultiLinesEx // 读取位置前需要先通过该指令设置编码器多圈线数 |
| 指令示例 | <pre>short coreTemp=1,encoderTemp=1; unsigned long long multiLines = 65536; // 多圈线数为16线, 2^16 rtn = GTN_RN_SetEncMultiLinesEx(coreTemp,encoderTemp,multiLines);// 设置多圈线数 long long absEncPosEx; rtn = GTN_RN_GetAbsEncPosEx(coreTemp,encoderTemp,&absEncPosEx); // 读取位置</pre> |
| 备注 | 如果该路编码器不是绝对式的编码器, 读取到的位置不会变化, |

指令 38 GTN_RN_SetEncMultiLinesEx

| | |
|------------|---|
| 指令原型 | GT_API GTN_RN_SetEncMultiLinesEx(short core, short encoder,unsigned long long multiLines) |
| 指令说明 | 设置绝对式编码器多圈线数。 |
| 指令类型 | 立即指令, 调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 |
| core | 核号, 正整数, 取值范围[1,2] |
| encoder | 第几路编码器 |
| multiLines | 绝对式编码器多圈线数, 如16线, 则填65536 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_RN_GetAbsEncPosEx |
| 指令示例 | <pre>short coreTemp=1,encoderTemp=1; unsigned long long multiLines = 65536; // 多圈线数为16线, 2^16 rtn = GTN_RN_SetEncMultiLinesEx(coreTemp,encoderTemp,multiLines);// 设置多圈线数 long long absEncPosEx; rtn = GTN_RN_GetAbsEncPosEx(coreTemp,encoderTemp,&absEncPosEx); // 读取位置</pre> |
| 备注 | |

11. 脉冲当量不一致的 PSO 功能

指令 39 GTN_SetPosComparePsoPrmEx

| | |
|-------|---|
| 指令原型 | GT_API GTN_SetPosComparePsoPrmEx(short core,short index,TPosComparePsoPrmEx *pPrm); |
| 指令说明 | 设置等间距位置比较输出参数。 |
| 指令类型 | 立即指令, 调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 |
| core | 核号, 正整数, 取值范围[1,2] |
| Index | 位置比较索引, 正整数, 取值范围[1,8]。 |
| pPrm | 等间距位置比较输出参数。 <pre>typedef struct { unsigned long count;// 保留, 使用时设置大于0的数 unsigned short hso; // 保留</pre> |

指令返回值
相关指令

```

unsigned short gpo;// 保留
long startPosX;      // 保留
long startPosY;      // 保留
long time;           // 保留

short pad1;          // 保留
short multiNumber;  // PSO输出间距有效个数，目前只能为 1
long scale[6];       // X/Y/Z/A/B/C..轴脉冲当量，单位：Pulse/毫米。维数由指令
// GTN_SetPosCompareMode中的参数dimension决定，目前只能是二维.
double syncPosArray[256];//PSO输出间距，X、Y的合成距离，单位：微米
} TPosComparePsoPrmEx;
    
```

请参照指令返回值列表。

GTN_GetPosComparePsoPrmEx

指令示例

```

short index = 1;
short station = 1;
short pPermit[16];
short permit;
short core = 1;

rtn = GTN_PosCompareStop(core,index);
//设置控制权
rtn = GTN_GetTerminalPermitEx(core,station,MC_HSO,&permit,index,1);
permit = 0x2;
rtn = GTN_SetTerminalPermitEx(core,station,MC_HSO,&permit,index,1);
//设置位置比较输出模式
TPosCompareMode prm;
rtn = GTN_PosCompareClear(core,index);           //清空位置比较输出数据
//设置位置比较输出参数
TPosCompareMode mode;
rtn = GTN_GetPosCompareMode(core,index,&mode);
mode.mode = 2;
mode.dimension = 2;           //2维 模式
mode.sourceMode = 1;         //输出比较源，0:编码器，1: 脉冲计数器
mode.sourceX = 1;           //X轴比较源[1,12]
mode.sourceY = 2;           //Y轴比较源[1,12]
mode.outputMode = 1;         //0: 输出脉冲， : 1,输出电平
mode.outputCounter = 1;     //保留，需要大于0.
mode.outputPulseWidth = 100; //输出脉冲宽度，单位：.1ms,电平模式下该参数无效
mode.errorBand = 10;
rtn = GTN_SetPosCompareMode(core,index,&mode);

//设置等间距输出相关参数
TPosComparePsoPrmEx psoPrm;
rtn = GTN_GetPosComparePsoPrmEx(core,index,&psoPrm);
psoPrm.count = 1;           //输出个数，暂时保留，必须大于0
psoPrm.scale[0] = 10000; //X轴脉冲当量，单位：Pulse/毫秒
    
```

```

psoPrm.scale[1] = 20000;//Y轴脉冲当量，单位：Pulse/毫秒
psoPrm.scale[2] = 1; //保留，必须大于0
psoPrm.scale[3] = 1; //保留，必须大于0
psoPrm.scale[4] = 1; //保留，必须大于0
psoPrm.scale[5] = 1; //保留，必须大于0
psoPrm.multiNumber = 1; //PSO间距个数，目前只能为1.
psoPrm.syncPosArray[0] = 1;//PSO输出间距，X、Y的合成距离，单位：微米
rtn = GTN_SetPosComparePsoPrmEx(core,index,&psoPrm); //启动位置比较输出

//启动位置比较输出
rtn = GTN_PosCompareStart(core,index);
    
```

指令 40 GTN_GetPosComparePsoPrmEx

| | |
|-------|--|
| 指令原型 | GT_API GTN_GetPosComparePsoPrmEx(short core,short index,TPosComparePsoPrmEx *pPrm); |
| 指令说明 | 获取等间距位置比较输出参数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 核号，正整数，取值范围[1,2] |
| Index | 位置比较索引，正整数，取值范围[1,8]。 |
| pPrm | 等间距位置比较输出参数。 typedef struct { unsigned long count;// 保留，使用时设置大于0的数 unsigned short hso; // 保留 unsigned short gpo; // 保留 long startPosX; // 保留 long startPosY; // 保留 long time; // 保留 short pad1; // 保留 short multiNumber; // PSO间距个数，目前只能为1 long scale[6]; // X/Y/Z/A/B/C..轴脉冲当量，单位：Pulse/毫秒。目前只能是二维。 double syncPosArray[256];//PSO输出间距，X、Y的合成距离，单位：微米 } TPosComparePsoPrmEx; |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_SetPosComparePsoPrmEx |
| 指令示例 | <pre> short index = 1; short station = 1; short pPermit[16]; short permit; short core = 1; </pre> |

```

rtn = GTN_PosCompareStop(core,index);
//设置控制权
rtn = GTN_GetTerminalPermitEx(core,station,MC_HSO,&permit,index,1);
permit = 0x2;
rtn = GTN_SetTerminalPermitEx(core,station,MC_HSO,&permit,index,1);
//设置位置比较输出模式
TPosCompareMode prm;
rtn = GTN_PosCompareClear(core,index); //清空位置比较输出数据
//设置位置比较输出参数
TPosCompareMode mode;
rtn = GTN_GetPosCompareMode(core,index,&mode);
mode.mode = 2;
mode.dimension = 2; //2维 模式
mode.sourceMode = 1; //输出比较源, 0:编码器, 1: 脉冲计数器
mode.sourceX = 1; //X轴比较源[1,12]
mode.sourceY = 2; //Y轴比较源[1,12]
mode.outputMode = 1; //0: 输出脉冲, : 1,输出电平
mode.outputCounter = 1; //保留, 需要大于0.
mode.outputPulseWidth = 100; //输出脉冲宽度, 单位: .1ms,电平模式下该参数无效
mode.errorBand = 10;
rtn = GTN_SetPosCompareMode(core,index,&mode);

//设置等间距输出相关参数
TPosComparePsoPrmEx psoPrm;
rtn = GTN_GetPosComparePsoPrmEx(core,index,&psoPrm);
psoPrm.count = 1; //输出个数, 暂时保留, 必须大于0
psoPrm.scale[0] = 10000; //X轴脉冲当量, 单位: Pulse/毫秒
psoPrm.scale[1] = 20000; //Y轴脉冲当量, 单位: Pulse/毫秒
psoPrm.scale[2] = 1; //保留, 必须大于0
psoPrm.scale[3] = 1; //保留, 必须大于0
psoPrm.scale[4] = 1; //保留, 必须大于0
psoPrm.scale[5] = 1; //保留, 必须大于0
psoPrm.multiNumber = 1; //PSO间距个数, 目前只能为1.
psoPrm.syncPosArray[0] = 1; //PSO输出间距, X、Y的合成距离, 单位: 微米
rtn = GTN_SetPosComparePsoPrmEx(core,index,&psoPrm); //启动位置比较输出

//启动位置比较输出
rtn = GTN_PosCompareStart(core,index);
    
```

12. 插补轮廓误差控制

指令 41 GTN_SetCrdContourErrorControl

| | |
|------|--|
| 指令原型 | GT_API GTN_SetCrdContourErrorControl(short core,short crd,short enable,double percent) |
| 指令说明 | 设置插补轮廓误差控制。 |
| 指令类型 | 立即指令, 调用后立即生效。 |
| 指令参数 | 该指令共有 4 个参数, 参数的详细信息如下。 |

| | |
|---------|---|
| core | 核号，正整数，取值范围[1,2] |
| Crd | 坐标系索引号 |
| enable | 0：关闭轮廓误差控制功能 1：开启轮廓误差控制 |
| percent | 控制系数：推荐值80 值越大控制越效果越明显 |
| 指令返回值 | 请参照指令返回值列表。 |
| 指令示例 | 调用GTN_SetCrdPrm设置坐标参数之后，再调用该指令设置开启或关闭轮廓误差控制 |

13. 同时设置/读取多轴参数相关指令

指令 42 GTN_SetPosArray

| | |
|---------|--|
| 指令原型 | GT_API GTN_SetPosArray(short core, short profile, double *pPos,short count) |
| 指令说明 | 设置目标位置。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏 |
| profile | 规划轴号，正整数，取值范围请参照指令参数范围中的“轴”一栏 |
| pPos | 设置目标位置，单位：pulse。取值范围：[-1073741824, 1073741823]。 |
| count | 设置的轴数 |
| 指令返回值 | 若返回值为 1：请检查当前轴是否为 Trap 模式。 其他返回值：请参照指令返回值列表。 |
| 相关指令 | GTN_GetPosArray |
| 指令示例 | <pre> short i=0,axisCount=5; // Trap模式下同时设置5个轴的目标位置 double targetPos[5]; for (i=0;i<5;i++) { targetPos[i] = 1000; } rtn = GTN_SetPosArray(1,axis,targetPos,axisCount); </pre> |

指令 43 GTN_GetPosArray

| | |
|---------|---|
| 指令原型 | GT_API GTN_GetPosArray(short core, short profile, double *pPos,short count) |
| 指令说明 | 读取目标位置。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏 |
| profile | 规划轴号。正整数，取值范围请参照指令参数范围中的“轴”一栏 |
| pPos | 读取目标位置，单位：pulse。 |
| count | 读取轴数 |
| 指令返回值 | 若返回值为 1：请检查当前轴是否为 Trap 模式/MovePos 模式。 其他返回值：请参照指令返回值列表。 |
| 相关指令 | GTN_SetPosArray |
| 指令示例 | // 同时读取5个轴的规划位置 |

```
short axisCount=5;
double targetPosArray[5];
axis = 1;
rtn = GTN_GetPosArray(1,axis,targetPosArray,axisCount);
```

指令 44 GTN_SetVelArray

| | |
|---------|---|
| 指令原型 | GT_API GTN_SetVelArray(short core,short profile,double *pVel,short count) |
| 指令说明 | 读取目标速度。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 |
| profile | 规划轴号。正整数，取值范围请参照 指令参数范围 中的“轴”一栏 |
| pVel | 目标速度，单位：pulse/ms。 |
| count | 设置的轴数 |
| 指令返回值 | 若返回值为 1：请检查当前轴是否为 Trap 模式/Jog 模式。 其他返回值：请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | <pre>// 同时读取6个轴的规划速度 short axisCount=6; double velArray[6]; short i,axis=1; for(i=0;i<axisCount;i++) { velArray[i] = 4; } rtn =GTN_SetVelArray(1,axis,&velArray[0],axisCount);</pre> |

指令 45 GTN_SetTrapPrmArray

| | |
|---------|---|
| 指令原型 | GT_API GTN_SetTrapPrmArray(short core,short profile,TTrapPrm *pPrm,short count) |
| 指令说明 | 设置 Trap 运动参数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 |
| profile | 规划轴号。正整数，取值范围请参照 指令参数范围 中的“轴”一栏 |
| pPrm | 点位运动参数，结构体说明参考 GTN_SetTrapPrm。 |
| count | 设置的轴数 |
| 指令返回值 | 若返回值为 1：请检查当前轴是否为 Trap 模式。 其他返回值：请参照指令返回值列表。 |
| 相关指令 | GTN_SetTrapPrm |
| 指令示例 | <pre>// 同时读取6个轴的Trap参数 short axisCount=6; TrapPrm trapArray[6];</pre> |

```

short axis=1;
short axisTemp;
for(axisTemp=0;axisTemp<axisCount;axisTemp++)
{
    rtn =GTN_GetTrapPrm(core,axisTemp+1,&trapArray[axisTemp]);
    trapArray[axisTemp].acc=0.1;
    trapArray[axisTemp].dec=0.1;
    trapArray[axisTemp].velStart=0;
    trapArray[axisTemp].smoothTime=50;
}
rtn =GTN_SetTrapPrmArray(1,axis,&trapArray[0],axisCount);
    
```

指令 46 GTN_SetTriggerArray

| | |
|----------|---|
| 指令原型 | GT_API GTN_SetTriggerArray(short core,short trigger,TTrigger *pTrigger,short count) |
| 指令说明 | 设置捕获参数。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏 |
| trigger | Trigger 序号，正整数，取值范围请参照指令参数范围中的“轴”一栏 |
| pTrigger | 捕获参数。结构体说明参考 GTN_SetTrigger |
| count | 设置的轴数 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_SetTrigger |
| 指令示例 | <pre> // 同时读取3路捕获参数 short coreTemp; TTrigger trigger[5]; short count=3; short i=1,rtn; coreTemp = 1; rtn = GTN_ClearTriggerStatusArray(coreTemp,i,count); for(i=0;i<count;i++) { rtn = GTN_GetTrigger(coreTemp,i+1,&trigger[i]); trigger[i].encoder = 1; // 捕获编码器 1. trigger[i].probeType = CAPTURE_INDEX; // 捕获类型 trigger[i].probeIndex = 1; // 捕获类型对应的资源索引。 trigger[i].sense = 0; // 捕获源触发沿 trigger[i].firstPosition = 0; // 捕获窗宽范围内的编码器位置才有效，捕获未在范围内则重新启动捕获。 trigger[i].lastPosition = 0; trigger[i].windowOnly = 0; // 捕获窗使能，1表示使能，使能后firstPosition和lastPosition才有效 trigger[i].loop = 1; // 捕获次数，0表示无限次 trigger[i].offset = 0; // 捕获偏移位置 } </pre> |

```
rtn = GTN_SetTriggerArray(coreTemp,1,&trigger[0],count);
```

指令 47 GTN_ClearTriggerStatusArray

| | |
|---------|--|
| 指令原型 | GT_API GTN_ClearTriggerStatusArray(short core,short trigger,short count) |
| 指令说明 | 清除捕获状态。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏 |
| trigger | Trigger 序号，正整数，取值范围请参照指令参数范围中的“轴”一栏 |
| count | Trigger 数量 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | <pre>// 同时读取3路捕获参数 short coreTemp; TTrigger trigger[5]; short count=3; short i=1,rtn; coreTemp = 1; rtn = GTN_ClearTriggerStatusArray(coreTemp,i,count); for(i=0;i<count;i++) { rtn = GTN_GetTrigger(coreTemp,i+1,&trigger[i]); trigger[i].encoder = 1; // 捕获编码器 1. trigger[i].probeType = CAPTURE_INDEX; // 捕获类型 trigger[i].probeIndex = 1; // 捕获类型对应的资源索引。 trigger[i].sense = 0; // 捕获源触发沿 trigger[i].firstPosition = 0; // 捕获窗宽范围内的编码器位置才有效，捕获未在范围内则重新启动捕获。 trigger[i].lastPosition = 0; trigger[i].windowOnly = 0; // 捕获窗使能，1表示使能，使能后firstPosition和lastPosition才有效 trigger[i].loop = 1; // 捕获次数，0表示无限次 trigger[i].offset = 0; // 捕获偏移位置 } rtn = GTN_SetTriggerArray(coreTemp,1,&trigger[0],count);</pre> |

14. 检查网络结构是否和配置文件中一致

指令 48 GTN_CheckRingNetStructure

| | |
|------|--|
| 指令原型 | GT_API GTN_CheckRingNetStructure(short core,char *pFile,unsigned short *pStatus) |
|------|--|

| | |
|---------|--|
| 指令说明 | 检查网络结构是否和配置文件中一致。 |
| 指令类型 | 立即指令，调用后立即生效。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下： |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 |
| pFile | 网络配置文件 |
| pStatus | 0，表当前网络结构和配置文件中的网络结构不一致，7 表示不一致。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 指令示例 | <pre>unsigned short connectSts; short rtn; rtn = GTN_CheckRingNetStructure(1,"core1NetCfg.cfg",&connectSts);</pre> |

15. 驱动器功能

驱动器功能指令只适用于固高科技 GSHD 伺服驱动器

注意事项：

- a) [GTN_SetMotionMode](#) 与 [GTN_CtrlMode](#) 指令必须在轴下使能状态调用时才会生效。
- b) 目前共有 4 种设置可供配套使用，
 - 1) [MotionMode](#) = 5 , [CtrlMode](#) = 1。即驱动器为周期同步位置模式，控制器为脉冲模式。这种情况即为一般使用情况，控制器发送位置脉冲信息，驱动器接收目标位置指令，按照正常情况使用即可。
 - 2) [MotionMode](#) = 6 , [CtrlMode](#) = 0。即驱动器为周期同步速度模式，控制器为闭环模式。这种情况下，控制器必须设置 PID 参数，控制器计算出的控制量发送给驱动器作为驱动器的目标速度，使电机按照设置的速度运动。需要注意的是，一般情况下 [kp](#) 值不宜过大，过大时有飞车危险，需视现场使用情况调节 [kp](#) 参数。具体使用可以参考例程 1。
 - 3) [MotionMode](#) = 6 , [CtrlMode](#) = 1。即驱动器为周期同步速度模式，控制器为脉冲模式。这种情况下，仅能通过 [GTN_SetDac](#) 或 [GTN_SetDrvPrfVel](#) 两条指令设置驱动器目标速度来控制电机运动。这种情况下，控制器规划位置不发生变化。具体使用可以参考例程 2、3。当调用 [GTN_SetDac](#) 或 [GTN_SetDrvPrfVel](#) 后，目标轴会以目标速度一直运动，当且仅当再次调用 [GTN_SetDac](#) 或 [GTN_SetDrvPrfVel](#) 指令将目标速度设为 0 时才能停止运动，此时 [GTN_Stop](#) 指令无效，在有撞机危险时请谨慎调用。
 - 3) [MotionMode](#) = 7 , [CtrlMode](#) = 0 或 1。即驱动器为周期同步力矩模式。这种情况下，仅能通过 [GTN_SetPrfTorque](#) 指令设置驱动器目标力矩来控制电机运动。这种情况下，控制器规划位置不发生变化。具体使用可以参考例程 4。当调用 [GTN_SetPrfTorque](#) 并设置力矩后，目标轴会以一直运动，当且仅当再次调用 [GTN_SetPrfTorque](#) 指令将目标力矩设为 0 时才能停止运动，此时 [GTN_Stop](#) 指令无效，在有撞机危险时请谨慎调用。
- c) [GTN_SetDac](#) 与 [GTN_SetDrvPrfVel](#) 主要区别在于其参数类型与目标速度的转换公式不一样。[GTN_SetDac](#) 参数为 16 位的 [short](#) 型，当参数为 [-32768,32767] 时，其对应为正负最

大转速。[GTN_SetDrvPrfVel](#)参数为32位的long型，参数为[-16777216, 16777215]，其对应的为正负最大转速，调用参数与目标速度的转换公式在指令详细说明列表中。

d) [GTN_SetPrfTorque](#)内设置的输入参数需要进行计算，计算公式如下：

$$\text{prfTorque} = 1000 * \frac{\text{力矩(N/m)}}{\text{力矩系数}} / \text{额定电流}$$

其中力矩系数和额定电流均需要查看驱动器参数，按照驱动器实际参数设置。

指令 49 GTN_GetMotionMode

| | | | |
|------------|--|------|---|
| 指令原型 | short GTN_GetMotionMode(short core, short axis, short *motionMode) | | |
| 指令说明 | 读取驱动器环路模式 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| motionMode | 驱动器环路模式 | | |
| | motionMode为5时：驱动器环路模式为周期同步位置模式。 motionMode为6时：驱动器环路模式为周期同步速度模式。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_SetMotionMode | | |
| 指令示例 | 例程 5-2：设置驱动器为周期同步速度模式 | | |

指令 50 GTN_SetMotionMode

| | | | |
|------------|--|------|---|
| 指令原型 | short GTN_SetMotionMode(short core, short axis, short motionMode) | | |
| 指令说明 | 设置驱动器环路模式 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| motionMode | 驱动器环路模式 | | |
| | motionMode为5时：驱动器环路模式为周期同步位置模式。 motionMode为6时：驱动器环路模式为周期同步速度模式。 | | |
| count | 读取的轴数，默认为1，正整数 | | |
| pClock | 读取控制器时钟，默认值为：NULL，即不用读取控制器时钟。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_GetMotionMode | | |
| 指令示例 | 例程 5-2：设置驱动器为周期同步速度模式 | | |

指令 51 GTN_SetDac

| | | | |
|--------|--|------|---|
| 指令原型 | short GTN_SetDac(short core, short dac, short *pValue, short count=1) | | |
| 指令说明 | 设置驱动器处于周期同步速度模式时的目标速度 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| dac | dac起始轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| pValue | 输出电压。整数，取值范围[-32767,32767] 驱动器处于周期同步速度模式时可通过公式转换为驱动器的目标速度 转换公式如下： 目标速度 = pValue * 驱动器最大速度 / 2¹⁵ | | |
| count | 设置的通道数。默认为 1，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | 例程 5-3：设置驱动器为周期同步速度模式，并通过 setdac 控制运动。 | | |

指令 52 GTN_SetDrvPrfVel

| | | | |
|-------|--|------|---|
| 指令原型 | short GTN_SetDrvPrfVel(short core, short axis, short *pValue, short count=1) | | |
| 指令说明 | 设置驱动器处于周期同步速度模式时的目标速度 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| pSts | 32位状态字： bit0：驱动器伺服准备状态，0：未完成，1：完成 bit1：驱动器伺服状态，0：使能未开启，1：使能开启 | | |
| count | 读取的轴数，默认为1，正整数 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 例程 5-4：设置驱动器为周期同步速度模式，并通过 SetDrvPrfVel 控制运动。 | | |

指令 53 GTN_SetPrfTorque

| | | | |
|------|---|------|---|
| 指令原型 | short GTN_SetPrfTorque(short core, short axis, short prfTorque) | | |
| 指令说明 | 设置驱动器周期同步模式力矩参数 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |

| | |
|-----------|--|
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 |
| | 驱动器电流环输入，输入参数需参照如下公式： |
| prfTorque | $\text{prfTorque} = 1000 * \frac{\text{力矩(N/m)}}{\text{力矩系数}} / \text{额定电流}$ |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 例程 5-5：设置驱动器为周期同步力矩模式，并通过 GTN_SetPrfTorque 控制运动。 |

指令 54 GTN_GetAtlTorque

| | | | |
|-----------|--|------|---|
| 指令原型 | short GTN_GetAtlTorque (short core, short axis, short AtlTorque) | | |
| 指令说明 | 读取驱动器周期同步力矩模式下的力矩 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| | 驱动器电流环输入，输入参数需参照如下公式： | | |
| AtlTorque | $\text{prfTorque} = 1000 * \frac{\text{力矩(N/m)}}{\text{力矩系数}} / \text{额定电流}$ | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 例程 5-5：设置驱动器为周期同步力矩模式，并通过 GTN_SetPrfTorque 控制运动。 | | |

指令 55 GTN_GetDriverSts

| | | | |
|--------|---|------|---|
| 指令原型 | short GTN_GetDriverSts (short core,short axis,long *pSts,short count,unsigned long *pClock) | | |
| 指令说明 | 获取驱动器状态。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| axis | 控制轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |
| | 32位状态字： | | |
| pSts | bit0：驱动器伺服准备状态，0：未完成，1：完成 bit1：驱动器伺服状态，0：使能未开启，1：使能开启 | | |
| count | 读取的轴数，默认为1，正整数 | | |
| pClock | 读取控制器时钟，默认值为：NULL，即不用读取控制器时钟。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |

指令示例 无

指令 56 GTN_SetServoPosLoopPid

| | | | |
|------------------|---|------|---|
| 指令原型 | short GTN_SetServoPosLoopPid(short core,short axis,TServoPosLoopPid *pServoPosLoopPid) | | |
| 指令说明 | 设置驱动器位置环系数 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| axis | 逻辑轴号。 | | |
| pServoPosLoopPid | PID参数： <pre>typedef struct { double value; double reverse[4]; }TServoPosLoopPidMode0; typedef union { TServoPosLoopPidMode0 servoPosLoopPidMode0; }TServoPosLoopPidUnion; typedef struct { short mode; TServoPosLoopPidUnion servoPosLoopPidPrm; }TServoPosLoopPid;</pre> mode: PID参数的模式，目前仅支持0号模式 servoPosLoopPidPrm: 对应PID参数的模式，设置pid的参数： mode = 0时，pid对应参数为servoPosLoopPidMode0.value，单位0.1HZ，参数范围[0,3276.7]，设置为负值时指令不报错，但会限制在0，通过Get指令读回来的pid为0。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_GetServoPosLoopPid | | |
| 指令示例 | | | |

指令 57 GTN_GetServoPosLoopPid

| | | | |
|------|--|------|---|
| 指令原型 | short GTN_GetServoPosLoopPid(short core,short axis,TServoPosLoopPid *pServoPosLoopPid) | | |
| 指令说明 | 读取驱动器位置环系数 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| axis | 逻辑轴号。 | | |

| | | | |
|------------------|--|--|--|
| pServoPosLoopPid | PID参数：详细说明请参考 GTN_SetServoPosLoopPid | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_SetServoPosLoopPid | | |
| 指令示例 | | | |

指令 58 GTN_RN_SetTorqueLimit

| | | | |
|--------------|--|------|---|
| 指令原型 | short GTN_RN_SetTorqueLimit(short core, short axis, TTorqueLimit *pTorqueLimit) | | |
| 指令说明 | 设置GSHD驱动器力矩限制 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| pTorqueLimit | <pre>typedef struct { unsigned short torqueMax; // 最大力矩，单位：千分号，1000表示1000‰ unsigned short torquePostive; // 正向力矩，单位：千分号，1000表示1000‰ unsigned short torqueNegative; // 反向力矩，单位：千分号，1000表示1000‰ short reserve1; double reserve2[4]; }TTorqueLimit;</pre> | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_RN_GetTorqueLimit | | |
| 指令示例 | | | |

指令 59 GTN_RN_GetTorqueLimit

| | | | |
|--------------|--|------|---|
| 指令原型 | short GTN_RN_GetTorqueLimit(short core, short axis, TTorqueLimit *pTorqueLimit) | | |
| 指令说明 | 获取GSHD驱动器力矩限制 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| pTorqueLimit | <pre>typedef struct { unsigned short torqueMax; // 最大力矩，单位：千分号，1000表示1000‰ unsigned short torquePostive; // 正向力矩，单位：千分号，1000表示1000‰ unsigned short torqueNegative; // 反向力矩，单位：千分号，1000表示1000‰ short reserve1; double reserve2[4]; }TTorqueLimit;</pre> | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | GTN_RN_SetTorqueLimit | | |
| 指令示例 | | | |

指令 60 GTN_RN_GetServoAlarmInfo

| | |
|------|--|
| 指令原型 | short GTN_RN_GetServoAlarmInfo(short core,short axis,unsigned long *pAlarmCode); |
|------|--|

| | | | | | | | | | |
|------------|---|------------|------------------|-------------|-------|------------------|--------|--------------|-----------|
| 指令说明 | 获取驱动器报警信息 | | | | | | | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 | | | | | | |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | | | | | | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | | | | | | | |
| axis | 轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | | | | | | | |
| pAlarmCode | 获取的伺服驱动器报警代码，具体含义如下： 报警代码一共 32 位，每一位表示一个报警信息，0 表示正常，1 表示该位有报警信息。 | | | | | | | | |
| | Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 报警信息 | IO 故障 | 过温 | 过载 | 编码器故障 | RST 输入 缺相 | 欠压 | 过压 | 过流 |
| | Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | 报警信息 | 电机过温 | 电流跟踪 误差超限 | 瞬时过 流 | 方向错误 | 保留 | 过速 | 功率模块 过流 | 寄存器故 障 |
| | Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 报警信息 | 速度跟随 误差超限 | 总线通信 异常 | 电机抱 闸电源 故障 | 安全继电器 故障 | 风扇故障 | 电机抱 闸故障 | STO 报警 | 位置跟随 误差超限 | |
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
| 报警信息 | 保留 | 保留 | 功率模 块过载 | 同步误差超 限 | 霍尔错误 | 寻相步 进距离 错误 | 回零失败 | 寻相失败 | |
| 指令返回值 | | | | | | | | | |
| 相关指令 | GTN_RN_SetTorqueLimit | | | | | | | | |
| 指令示例 | | | | | | | | | |

指令 61 GTN_ReadServoParamInfo

| | | | |
|------|---|------|---|
| 指令原型 | short GTN_ReadServoParamInfo(short core, short axis, TServoParamReader *pTServoParamReader, unsigned char* pData) | | |
| 指令说明 | 读取GSHD驱动器参数 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| axis | 轴号，正整数，取值范围请参照 指令参数范围 中的“轴”一栏 | | |

| | | | | | | | |
|--------------------------------|--|------------|--------|---------------|--------|--------|------|
| pTServoParamReader | GSHD 参数结构体，具体含义如下： typedef struct { unsigned long objectIndex; // 一级指令字 unsigned short subIndex; // 二级指令字 unsigned short reserve; // 保留位 short byteSize; // 字节尺寸 short memType; // 对象存储位置 0-RAM 1-FLASH }TServoParamReader; | | | | | | |
| | 指令字说明： | | | | | | |
| | 参数名称 | 一级指令字 | 二级指令字 | 字节尺寸 | 对象存储位置 | 对象属性 | 单位 |
| | 额定电流 | 0x00290000 | 0x0001 | 2 | RAM | 只读 | 0.1A |
| | 峰值电流 | 0x00290000 | 0x0002 | 2 | RAM | 只读 | 0.1A |
| 速度环时间常数 | 0x00002099 | 0x0000 | 2 | RAM/ FLASH | 读写 | 0.01ms | |
| 寻相结束标志 | 0x00290000 | 0x0022 | 2 | RAM | 只读 | | |
| 额定转速 | 0x00290000 | 0x0003 | 2 | RAM | 只读 | 1rpm | |
| pData 指令返回值 相关指令 指令示例 | 如果需要读取额定电流，则一级指令字需要填 0x00290000，二级指令字需要填 0x0001，字节尺寸填 2，对象存储位置填 0，保留位 reserve 填 0，然后 pData 返回的值为额定电流。 | | | | | | |
| | 读回 GSHD 驱动器的参数值 如果是读取寻相结束标志，1：表示寻相完成，0：表示寻相未完成。 | | | | | | |
| | | | | | | | |
| | | | | | | | |

16. 回零功能

指令 62 GTN_GoHome

| | | | |
|------|---|------|---|
| 指令原型 | short GTN_GoHome(short core,shortaxis,THomePrm *pHomePrm) | | |
| 指令说明 | 启动 Smart Home 实现各种方式回原点 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |

| | |
|----------|--|
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏 |
| | 需要进行回原点的轴号，正整数，取值范围请参照指令参数范围中的“轴”一栏 |
| axis | 设置 Smart Home 回原点的参数，该参数为一结构体，详细参数定义及说明请参照结构体 THomePrm: |
| | <pre> typedef struct { short mode; // 回原点模式，参考下面的回原点模式宏定义 short moveDir; // 设置启动搜索原点时的运动方向：非正数-负方向，正整数-正方向 short indexDir; // 设置搜索 Index 的运动方向：非正数-负方向，正整数-正方向 short edge; // 设置捕获沿：-下降沿，非值-上升沿 short triggerIndex; // 用于设置触发器：取值-1 和[1,8]，-1 表示使用的触发器和轴号对应，其它值表示使用其它轴的触发器，触发器用于实现高速硬件捕获，默认设置为-1 即可 short pad1[3]; // 保留（不需要设置） double velHigh; // 搜索 Home 速度（单位：pulse/ms） double velLow; // 搜索 Index 速度（单位：pulse/ms） double acc; // 回原点运动的加速度（单位：pulse/ms^2） double dec; // 回原点运动的减速度（单位：pulse/ms^2） short smoothTime; // 回原点运动的平滑时间：取值[0,1]，具体含义与 GTS 系列控制器点位运动相似 short pad2[3]; // 保留（不需要设置） long homeOffset; // 最终停止的位置相对于原点的偏移量 long searchHomeDistance; // Home 最大搜索距离，//0 表示不限制搜索距离，默认为或-805306368 long searchIndexDistance; // Index 最大搜索距离，//0 表示不限制搜索距离，默认为或-805306368 long escapeStep; // 采用“限位回原点”方式时，反方向离开限位的脱离步长 long pad3[2]; // 保留（不需要设置） } THomePrm; </pre> |
| pHomePrm | <p>回原点模式宏定义：</p> <p>HOME_MODE_LIMIT (10): 限位回原点</p> <p>HOME_MODE_LIMIT_HOME (11): 限位+Home回原点</p> <p>HOME_MODE_LIMIT_INDEX (12): 限位+Index回原点</p> <p>HOME_MODE_LIMIT_HOME_INDEX (13): 限位+Home+Index回原点</p> <p>HOME_MODE_HOME (20): Home回原点</p> <p>HOME_MODE_HOME_INDEX (22): Home+Index回原点</p> <p>HOME_MODE_INDEX (30): Index回原点</p> <p>HOME_MODE_DRIVER_HOME (42): 驱动器回零，使用条件：(1):(5,2)开卡;(2):GSHD驱动器。注意：使用该模式回零时，不需要设置回零运动参数</p> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无。 |
| 指令示例 | 无 |

17. Trigger 功能

指令 63 GTN_SetTriggerEx

| | | | |
|------------|---|------|---|
| 指令原型 | short GTN_SetTriggerEx (short core, short i, TTriggerEx *pTriggerEx) | | |
| 指令说明 | 设置trigger。增加设置Trigger捕获沿的方式，sence = 2，或者sence = 3，并启动捕获 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| i | Trigger序号 | | |
| pTriggerEx | <pre>typedef struct TriggerEx { short latchType; //锁存类型：MC_ENCODER short latchIndex; //编码器序号 short probeType; //捕获类型 // CAPTURE_HOME Home 捕获 // CAPTURE_INDEX index 捕获 // CAPTURE_PROBE DI 捕获 short probeIndex; // 对应捕获类型的序号 short sense; // 捕获沿：下降沿；上升沿； // 2 表示检测第一个下降沿后，锁存捕获值，之后再检测上升沿，锁存上升沿的捕获 // 值，最后通过 GTN_GetTriggerStatusEx 读回来的位置，为最后一次上升沿的捕获值和第 // 一次下降沿的捕获值之差。 // 3 表示检测第一个上升沿后，锁存捕获值，之后再检测下降沿，锁存下降沿的捕获 // 值，最后通过 GTN_GetTriggerStatusEx 读回来的位置，为最后一次上升沿的捕获值和第 // 一次下降沿的捕获值之差。 long offset; // 捕获位置偏置值，当 index 信号触发以后， // 偏置 offset 脉冲以后再锁存编码器 unsigned long loop; // 捕获循环次数：表示无限循环 short windowOnly; // 捕获窗口使能 long firstPosition; // 触发捕获位置的起点 long lastPosition; // 触发捕获位置的终点 } TTriggerEx;</pre> | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 64 GTN_GetTriggerStatusEx

| | | | |
|------|---|------|---|
| 指令原型 | short GTN_GetTriggerStatusEx (short core, short i, TTriggerStatusEx *pTriggerStatusEx, short count=1) | | |
| 指令说明 | 读取trigger状态。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏 | | |
| i | Trigger序号 | | |

| | |
|------------------|---|
| pTriggerStatusEx | <pre>typedef struct TriggerStatusEx { short execute; //捕获运行状态, : 使能捕获, : 未使能捕获 short done; //捕获状态, : 捕获完成, : 捕获中 long position; //捕获位置或刀径值 unsigned long clock; // 捕获触发事的时钟 unsigned long loopCount; // 捕获触发次数 } TTriggerStatusEx;</pre> |
| | count |
| 重点说明 | <p>1、在启动运动之前调用 GTN_SetTriggerEx 指令启动刀径检测。在运动结束以后调用 GTN_GetTriggerStatusEx 指令从 position 中读取刀径检测值。</p> <p>2、使用刀径检测功能时, sense 必须为 2 或 3, probeType 必须为 CAPTURE_HOME 或 CAPTURE_PROBE, loop 必须为 0。</p> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 65 GTN_GetTriggerLatchValue

| | | | |
|--------|--|------|---|
| 指令原型 | short GTN_GetTriggerLatchValue (short core, short index, long count, long *pValue, long *pCount, TLatchValueInfo *pInfo) | | |
| 指令说明 | 获取连续捕获锁存的多组数据。 | | |
| 指令类型 | 立即指令, 调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照指令参数范围中的“内核”一栏 | | |
| index | Trigger 序号, 正整数, 取值范围请参照指令参数范围中的“轴”一栏 | | |
| count | 需要获取的已锁存的编码器位置值的个数。 | | |
| pValue | 用于读取编码器位置值的数组指针, 数组的大小至少为 count。 | | |
| pCount | 实际读取到的编码器位置锁存值。 | | |
| | <p>控制器提供大小为 256 的空间保存编码器锁存值, 如果捕获到的次数大于 256, 则只保存前 256 个锁存值。</p> <p>如果只捕获到 50 次, 读取时 count 输入为 100, 则 pCount 返回真实读取到的个数 50。</p> | | |
| pInfo | 锁存值相关信息结构体 | | |
| | <pre>typedef struct { short fifoFull; // 锁存值空间已满 short pad1[2]; // 保留值 double pad2[2]; // 保留值 } TLatchValueInfo;</pre> | | |
| 重点说明 | | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

18. 缓冲区等待功能

指令 66 GTN_BufWaitDi

| | | | |
|--------------|--|------|---|
| 指令原型 | short GTN_BufWaitDi(short core, short crd, short diType, unsigned short diIndex, unsigned short level, unsigned short continueTime, unsigned long overTime, short flagMode, long segNum=0, short fifo=0) | | |
| 指令说明 | 设置缓冲区等待外部输入信号。 | | |
| 指令类型 | 缓存区指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 10 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| diType | <p>信号类型</p> <p>MC_LIMIT_POSITIVE: 正限位。</p> <p>MC_LIMIT_NEGATIVE: 负限位。</p> <p>MC_ALARM: 驱动报警。</p> <p>MC_HOME: 原点开关。</p> <p>MC_GPI: 通用输入。</p> <p>MC_ARRIVE: 电机到位信号。</p> <p>MC_MPG: 手轮MPG轴选和倍率信号（5V电平输入）。</p> | | |
| diIndex | 信号序号，取值范围请参照 指令参数范围 中的“通用输入”一栏。 | | |
| level | 信号有效电平。 | | |
| continueTime | 信号保持时间。 | | |
| overTime | 信号超时时间，单位：控制器的周期，如果控制器的周期为250us，则overTime的单位为250us。 | | |
| flagMode | 模式，0：立即模式，1：标识模式。 | | |
| segNum | 用户段号。 | | |
| fifo | 插补缓存区号。默认值为0，正整数，取值范围请参照 指令参数范围 中的“插补缓存区序号”。 | | |
| 重点说明 | <p>一、信号超时时间overTime</p> <p>0: 一直等待，直到检测到信号才继续运行后面的指令，等待时坐标系运行状态、坐标轴规划运行状态置起，使用指令GTN_GetBufWaitDiStatus读取的enable为1、count为计时的周期次数。</p> <p>>0: 超时后继续运行后面的指令。</p> <p>二、标识模式flagMode</p> <p>0: 立即模式，执行到这条指令时，停止运动、检测外部输入信号；如果没使用前瞻功能，则需把上一段运动指令的终点速度设置为0，否则将导致速度冲击。</p> <p>1: 标识模式</p> <p>a. 执行到这条指令时仅置起检测标识，在执行下一运动段时才进行检测。</p> <p>b. 如果检测到信号，则继续运行；如果没检测到信号，则减速停止。</p> <p>c. 在减速停止时，如果本段剩余距离足够停止下来，则停止在本段终点，否则停止到后面运动段内。</p> <p>d. 如下情况能够停止在本段终点：</p> <p>1. 本段剩余距离大于减速至零的距离；</p> | | |

| | |
|-------|--|
| | <p>2. 本段速度由零开始加速，加速过程中计算剩余距离是否足够减速停止，否则进入减速规划。</p> <p>e. 如下情况需停止到后面运动段内：</p> <p> 在本段之前，已加速到了一定速度，但本段的全部行程较短，且不足以由该速度减速至零。</p> <p>三、如果在等待过程中，用户暂停了插补运动，当再继续开始插补运动时，将会继续上一次的等待。</p> <p>四、如果在等待过程中，用户暂停了插补运动，使用指令GTN_GetBufWaitDiStatus读取的enable仍然为1，以用于标识是否有未结束的等待过程，该标识可以使用指令GTN_ClearBufWaitStatus清除。</p> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 67 GTN_BufWaitLongVar

| | | | |
|----------|--|------|---|
| 指令原型 | short GTN_BufWaitLongVar(short core, short crd, short index, long value, long overTime, short flagMode, long segNum=0, short fifo=0) | | |
| 指令说明 | 设置缓存区等待long型变量。 | | |
| 指令类型 | 缓存区指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| index | 控制器内部变量索引号，取值范围：[1,16] | | |
| value | 等待的变量目标值。 | | |
| overTime | 信号超时时间，单位：控制器的周期，如果控制器的周期为250us，则overTime的单位为250us。 | | |
| flagMode | 模式，0：立即模式，1：标识模式。 | | |
| segNum | 用户段号。 | | |
| fifo | 插补缓存区号。默认值为0，正整数，取值范围请参照《GXN系列运动控制器编程手册之基本功能》中的“插补缓存区序号”。 | | |
| 重点说明 | <p>一、信号超时时间overTime</p> <p>0：一直等待，直到检测到内部变量为目标值才继续运行后面的指令，等待时坐标系运行状态、坐标轴规划运行状态置起，使用指令GTN_GetBufWaitLongVarStatus读取的enable为1、status为停止状态。</p> <p>>0：超时后继续运行后面的指令。</p> <p>二、标识模式flagMode</p> <p>0：立即模式，执行到这条指令时，停止运动、检测内部变量是否为目标值；如果没使用前瞻功能，则需把上一段运动指令的终点速度设置为0，否则将导致速度冲击。</p> <p>1：标识模式</p> <p>a. 执行到这条指令时仅置起检测标识，在执行下一运动段时才进行检测。</p> <p>b. 如果内部变量为目标值，则继续运行；如果内部变量不是目标值，则减速停止。</p> <p>c. 在减速停止时，如果本段剩余距离足够停止下来，则停止在本段终点，否则停止到后面运动段内。</p> <p>d. 如下情况能够停止在本段终点：</p> | | |

| | |
|-------|--|
| | <p>1. 本段剩余距离大于减速至零的距离；</p> <p>2. 本段速度由零开始加速，加速过程中计算剩余距离是否足够减速停止，否则进入减速规划。</p> <p>e. 如下情况需停止到后面运动段内：</p> <p> 在本段之前，已加速到了一定速度，但本段的全部行程较短，且不足以由该速度降速至零。</p> <p>三、如果在等待过程中，用户暂停了插补运动，当再继续开始插补运动时，将会继续上一次的等待。</p> <p>四、如果在等待过程中，用户暂停了插补运动，使用指令GTN_GetBufWaitLongVarStatus读取的enable仍然为1，以用于标识是否有未结束的等待过程，该标识可以使用指令GTN_ClearBufWaitStatus清除。</p> |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 68 GTN_GetBufWaitDiStatus

| | | | |
|---------------|--|------|---|
| 指令原型 | short GTN_GetBufWaitDiStatus(short core, short crd, short *pDiType, unsigned short *pDiIndex, unsigned short *pLevel, unsigned short *pContinueTime, unsigned long *pOverTime, short *pFlagMode, long *pSegNum=0, short *pEnable, short *pCount, short fifo=0) | | |
| 指令说明 | 读取缓存区等待外部输入信号的状态。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 12 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| pDiType | <p>信号类型</p> <p>MC_LIMIT_POSITIVE: 正限位。</p> <p>MC_LIMIT_NEGATIVE: 负限位。</p> <p>MC_ALARM: 驱动报警。</p> <p>MC_HOME: 原点开关。</p> <p>MC_GPI: 通用输入。</p> <p>MC_ARRIVE: 电机到位信号。</p> <p>MC_MPG: 手轮MPG轴选和倍率信号（5V电平输入）。</p> | | |
| pDiIndex | 信号序号，取值范围请参照 指令参数范围 中的“通用输入”一栏。 | | |
| pLevel | 信号有效电平。 | | |
| pContinueTime | 信号保持时间。 | | |
| pOverTime | 信号超时时间。 | | |
| pFlagMode | 模式，0: 立即模式，1: 标识模式。 | | |
| pSegNum | 用户段号。 | | |
| pEnable | 是否在执行等待标识。 | | |
| pCount | 等待计时周期。 | | |
| fifo | 插补缓存区号。默认值为0，正整数，取值范围请参照 指令参数范围 中的“插补缓存区序号”。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |

| | |
|------|---|
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 69 GTN_GetBufWaitLongVarStatus

| | | | |
|-----------|---|------|---|
| 指令原型 | short GTN_GetBufWaitLongVarStatus(short core, short crd, short *pIndex, long *pValue, short *pFlagMode, long *pSegNum=0, short *pEnable, long *pStatus, short fifo=0) | | |
| 指令说明 | 读取缓存区等待变量的状态。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 9 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| pIndex | 控制器内部变量索引号，取值范围：[1,16] | | |
| pValue | 等待的变量目标值。 | | |
| pFlagMode | 模式，0：立即模式，1：标识模式。 | | |
| pSegNum | 用户段号。 | | |
| pEnable | 是否在执行等待标识。 | | |
| pStatus | 等待变量运行状态，0：默认状态，1：表示超时，2：表示在等待中。 | | |
| fifo | 插补缓存区号。默认值为0，正整数，取值范围请参照 指令参数范围 中的“插补缓存区序号”。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 70 GTN_ClearBufWaitStatus

| | | | |
|-------|---|------|---|
| 指令原型 | short GTN_ClearBufWaitStatus(short core, short crd, short fifo=0) | | |
| 指令说明 | 清除缓存区等待的状态。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| fifo | 插补缓存区号。默认值为0，正整数，取值范围请参照 指令参数范围 中的“插补缓存区序号”。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 71 GTN_SetLongVar

| | | | |
|------|---|------|---|
| 指令原型 | short GTN_SetLongVar(short core, short index, long value) | | |
| 指令说明 | 设置内部long型变量的值。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |

| | |
|-------|------------------------|
| index | 控制器内部变量索引号，取值范围：[1,16] |
| value | 变量目标值。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 72 GTN_GetLongVar

| | | | |
|--------|---|------|---|
| 指令原型 | short GTN_GetLongVar(short core, short index, long *pValue) | | |
| 指令说明 | 读取内部long型变量的值。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 2 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 控制器内部变量索引号，取值范围：[1,16] | | |
| pValue | 变量值。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

19. 椭圆插补

指令 73 GTN_Ellipse

| | | | |
|----------|--|------|---|
| 指令原型 | short GTN_Ellipse(short core, short crd, TEllipseParameter *pEllipse, double synVel, double synAcc, double velEnd, long segNum, short fifo) | | |
| 指令说明 | 椭圆插补指令（标准版本不支持）。 | | |
| 指令类型 | 缓存区指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| crd | 坐标系号，正整数，取值范围：[1,2]。 | | |
| pEllipse | 椭圆插补参数 <pre>typedef struct { double endPoint[INTERPOLATIONN_AXIS_MAX]; short plane; short dir; short overrideSelect; short mode; TEllipseParameterUnion data; }TEllipseParameter</pre> endPoint: 椭圆插补终点位置 INTERPOLATIONN_AXIS_MAX(8): 插补坐标系最大维数 plane: 椭圆平面选择 INTERPOLATIONN_CIRCLE_PLAT_XY(0): XY平面椭圆 INTERPOLATIONN_CIRCLE_PLAT_XY(1): YZ平面椭圆 INTERPOLATIONN_CIRCLE_PLAT_XY(2): ZX平面椭圆 | | |

| | |
|--------|--|
| | <p>dir: 椭圆方向 INTERPOLATIONN_CIRCLE_DIR_CW(0): 顺时针 INTERPOLATIONN_CIRCLE_DIR_CCW(1): 逆时针 overrideSelect: 倍率选择, 0: 第一组倍率, 1: 第二组倍率 mode: 椭圆描述方式 ELLIPSE_MODE_AUX_POINT(0): 辅助点描述模式 data: 椭圆描述参数</p> <p>椭圆描述参数联合体 typedef union { TEllipseAuxPoint ausPoint; double reserve[60]; }TEllipseParameterUnion; ausPoint: 椭圆辅助点描述模式参数</p> <p>typedefstruct { double pos[ELLIPSE_AUX_POINT_COUNT][INTERPOLATIONN_AXIS_MAX]; } TELLIPSE_AuxPoint; pos: 椭圆上辅助点位置坐标 ELLIPSE_AUX_POINT_COUNT(5): 椭圆辅助点描述所需辅助点个数 INTERPOLATIONN_AXIS_MAX(8): 插补坐标系最大维数</p> |
| synVel | 插补段的目标合成速度。取值范围: (0, 32767), 单位: pulse/ms |
| synAcc | 插补段的合成加速度。取值范围: (0, 32767), 单位: pulse/ms^2。 |
| velEnd | 插补段的终点速度。取值范围: [0, 32767), 单位: pulse/ms。该值只有在没有使用前瞻预处理功能时才有意义, 否则该值无效。默认值为: 0。 |
| segNum | 插补段段号标识。 |
| fifo | 插补缓存区号。取值范围: [0, 1], 默认值为: 0。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 74 GTN_EllipseEx

| | | | |
|----------|--|------|---|
| 指令原型 | short GTN_EllipseEx(short core, short crd, TELLIPSE_PARAMETER *pEllipse, double synVel, double synAcc, double velEnd, long segNum, short fifo) | | |
| 指令说明 | 椭圆插补前瞻指令 (标准版本不支持)。 | | |
| 指令类型 | 缓存区指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照指令参数范围中的“内核”一栏。 | | |
| crd | 坐标系号, 正整数, 取值范围: [1,2]。 | | |
| pEllipse | 椭圆插补参数 typedef struct | | |

```

{
    double endPoint[INTERPOLATIONN_AXIS_MAX];
    short plane;
    short dir;
    short overrideSelect;
    short mode;
    TEllipseParameterUnion data;;
}TEllipseParameter
endPoint: 椭圆插补终点位置
INTERPOLATIONN_AXIS_MAX(8): 插补坐标系最大维数
plane: 椭圆平面选择
INTERPOLATIONN_CIRCLE_PLAT_XY(0): XY平面椭圆
INTERPOLATIONN_CIRCLE_PLAT_XY(1): YZ平面椭圆
INTERPOLATIONN_CIRCLE_PLAT_XY(2): ZX平面椭圆
dir: 椭圆方向
INTERPOLATIONN_CIRCLE_DIR_CW(0): 顺时针
INTERPOLATIONN_CIRCLE_DIR_CCW(1): 逆时针
overrideSelect: 倍率选择, 0: 第一组倍率, 1: 第二组倍率
mode: 椭圆描述方式
ELLIPSE_MODE_AUX_POINT(0): 辅助点描述模式
data: 椭圆描述参数

椭圆描述参数联合体
typedef union
{
    TEllipseAuxPoint ausPoint;
    double reserve[60];
}TEllipseParameterUnion;
ausPoint: 椭圆辅助点描述模式参数

typedef struct
{
    double pos[ELLIPSE_AUX_POINT_COUNT][ INTERPOLATIONN_AXIS_MAX];
} TEllipseAuxPoint;
pos: 椭圆上辅助点位置坐标
ELLIPSE_AUX_POINT_COUNT(5): 椭圆辅助点描述所需辅助点个数
INTERPOLATIONN_AXIS_MAX(8): 插补坐标系最大维数
    
```

| | |
|--------|--|
| synVel | 插补段的目标合成速度。取值范围: (0, 32767), 单位: pulse/ms |
| synAcc | 插补段的合成加速度。取值范围: (0, 32767), 单位: pulse/ms^2。 |
| velEnd | 插补段的终点速度。取值范围: [0, 32767), 单位: pulse/ms。该值只有在没有使用前瞻预处理功能时才有意义, 否则该值无效。默认值为: 0。 |
| segNum | 插补段段号标识。 |
| fifo | 插补缓存区号。取值范围: [0, 1], 默认值为: 0。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |

| | |
|------|---|
| 指令示例 | 无 |
|------|---|

20. 振镜功能

指令 75 GTN_ScanBufLaserDelayLong

| | | | |
|---------------|--|------|---|
| 指令原型 | short GTN_ScanBufLaserDelayLong(short core, long laserOnDelay, long laserOffDelay, short scan) | | |
| 指令说明 | 设置振镜缓存区激光延时（32位接口）。 | | |
| 指令类型 | 缓存区指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| laserOnDelay | 振镜缓存区激光开延时，单位：us。 | | |
| laserOffDelay | 振镜缓存区激光开延时，单位：us。 | | |
| scan | 振镜坐标系号。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 76 GTN_SetScanAlarmAutoStopMode

| | | | |
|-------|---|------|---|
| 指令原型 | short GTN_SetScanAlarmAutoStopMode(short core,short mode,short crd=1) | | |
| 指令说明 | 设置振镜报警自动停止模式(报警信息通过振镜通讯协议从振镜反馈) | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| mode | 振镜报警是否会停止振镜运动 0: 振镜报警时，不停止振镜运动； 1: 振镜报警时，停止振镜运动。 | | |
| crd | 振镜坐标系号。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 77 GTN_GetScanAlarmAutoStopMode

| | | | |
|-------|---|------|---|
| 指令原型 | short GTN_GetScanAlarmAutoStopMode(short core,short *pMode,short crd=1) | | |
| 指令说明 | 读取振镜报警自动停止模式(报警信息通过振镜通讯协议从振镜反馈) | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| pMode | 当前的振镜报警自动停止模式 0: 振镜报警时，不停止振镜运动； 1: 振镜报警时，停止振镜运动。 | | |
| crd | 振镜坐标系号。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |

| | |
|------|---|
| 相关指令 | 无 |
| 指令示例 | 无 |

指令 78 GTN_GetScanErrorCode

| | | | |
|------------|--|------|---|
| 指令原型 | short GTN_GetScanErrorCode(short core,unsigned long *pErrorCode,short crd=1) | | |
| 指令说明 | 读取振镜报警信息(报警信息通过振镜通讯协议从振镜反馈) | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| pErrorCode | 读取到的振镜报警信息 | | |
| crd | 振镜坐标系号。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

21. 52 开卡模式读取物理站号和轴号

指令 79 GTN_GetResPhyInfo

| | | | |
|-----------------|--|------|---|
| 指令原型 | short GTN_GetResPhyInfo(short core,short type,short index,TResPhyInfo *pStationPhyInfo) | | |
| 指令说明 | 输入轴号，获取该轴所在的物理站号和轴号。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| type | 资源类型： MC_AXIS（目前仅支持该资源） | | |
| pStationPhyInfo | <pre>typedef struct { short index; //一级索引，站号 short id; //二级索引，该资源在站上的物理索引 short reverse1[2];//保留 double reverse2[4];//保留 }TResPhyInfo;</pre> | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 指令示例 | | | |

22. 运控模式配置功能

指令 80 GTN_SetMcMode

| | | | |
|------|--|------|---|
| 指令原型 | short GTN_SetMcMode(short core, short mode, short value) | | |
| 指令说明 | 配置运控模式。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |

| | | | |
|--------------|----------------------------------|--|---|
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| | mode | 需要配置的运控模式。 MC_MODE_POSITION_OVERFLOW_ENABLE: 位置值溢出模式 MC_MODE_ENCODER_LOOPBACK: 编码器环回模式 MC_MODE_SPORT_MODE: 通讯数据包模式 | |
| value | | MC_MODE_UNIT_MODE: 设置MoveContinuous的速度、加速度、加加速度的单位 MC_MODE_AXIS_FOLLOW_ERROR_SOURCE: 设置跟随误差数值的来源 MC_MODE_AXIS_FOLLOW_ALARM_SOURCE: 设置跟随误差报警的数据来源 MC_MODE_SMOOTH_STOP_MODE: 调用GTN_Stop, 停止模式为平滑停止时, 停止减速度使用哪个减速度进行停止 | |
| | 配置模式对应值。 | | |
| | mode参数数值 | Value数值 | 说明 |
| | MC_MODE_POSITION_OVERFLOW_ENABLE | OVERFLOW_ENABLE | 溢出处理 |
| | | OVERFLOW_DISABLE (默认) | 自然溢出 |
| | MC_MODE_ENCODER_LOOPBACK | LOOPBACK_ENABLE | 编码器源为规划位置环回 |
| | | LOOPBACK_DISABLE (默认) | 编码器源为实际编码器, 默认状态 |
| | MC_MODE_SPORT_MODE | SPORT_2 | 数据包格式为SPORT2.0 |
| | | SPORT_1 (默认) | 数据包格式为SPORT1.0 |
| | MC_MODE_UNIT_MODE | UNIT_MODE_PULSE | MoveContinuous的速度、加速度、加加速度的单位为pulse/ms |
| | | UNIT_MODE_UM (默认) | MoveContinuous的速度、加速度、加加速度的单位mm/s |
| | MC_MODE_AXIS_FOLLOW_ERROR_SOURCE | CONTROLLER_FOLLOW_ERROR (默认) | 控制器计算的跟随误差 (具有网络延时) |
| | | DRIVER_FOLLOW_ERROR | 驱动器计算的跟随误差 |
| | MC_MODE_AXIS_FOLLOW_ALARM_SOURCE | CONTROLLER_FOLLOW_ERROR (默认) | 根据控制器计算的跟随误差设置跟随误差报警的状态 |
| | | DRIVER_FOLLOW_ERROR | 驱动器计算的跟随误差设置跟随误差报警的状态 |
| | MC_MODE_SMOOTH_STOP_MODE | STOP_DEC_RECOVER (默认) | 使用各自运动模式中的减速度进行停止 |
| | | STOP_DEC_CHANGE | GTN_SetStopDec 中设置的减速度和运动模式中的减速度进行比较, 使用较大的减速度进行停止, 目前只支持Trap、Jog |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | 无 | | |

指令 81 GTN_GetMcMode

| | | | |
|------|--|------|---|
| 指令原型 | short GTN_GetMcMode(short core, short mode, short *pValue) | | |
| 指令说明 | 读取运控模式。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |

| | |
|--------|---|
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 |
| mode | 需要配置的运控模式。 请参照GTN_SetMcMode指令的参数2 |
| pValue | 配置模式对应值。 请参照GTN_SetMcMode指令的参数3 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | 无 |

23. 前瞻功能

指令 82 GTN_SetRadiusRatioTableLa

| | | | |
|---------|---|------|---|
| 指令原型 | short GTN_SetRadiusRatioTableLa(short core, short crd, short count,double *pRadius,double *pRatio) | | |
| 指令说明 | 设置前瞻曲率参数表。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 5 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| count | 设置的曲率参数表大小，正整数，取值范围：[1,16]。 | | |
| pRadius | 半径表，数组指针，数组大小为count。 | | |
| pRatio | 曲率参数表，数组指针，数组大小为count。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | <pre> double radius[5],ratio[5]; radius[0] = 0.1; radius[1] = 0.5; radius[2] = 1.0; radius[3] = 5.0; radius[4] = 10.0; ratio[0] = 0.001; //半径大小为 (0, 0.1]mm的圆，曲率参数未0.001 ratio[1] = 0.002; //半径大小为 (0.1, 0.5]mm的圆，曲率参数未0.002 ratio[2] = 0.01; //半径大小为 (0.5, 1.0]mm的圆，曲率参数未0.01 ratio[3] = 0.05; //半径大小为 (1.0, 5.0]mm的圆，曲率参数未0.05 ratio[4] = 0.1; //半径大小为 (5.0, 10.0]mm的圆，曲率参数未0.1 rtn = GTN_SetRadiusRatioTableLa(core,crd,5,radius,ratio); </pre> | | |

24. 点位运动新增功能

指令 83 GTN_GetTrapSts

| | |
|------|---|
| 指令原型 | short GTN_GetTrapSts(short core, short profile, short *pPrfSts) |
|------|---|

| | | | |
|---------|--|------|---|
| 指令说明 | 读取点位运动到位标志。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| profile | 规划轴号，正整数。取值范围请参照 指令参数范围 中的“轴”一栏。 | | |
| pPrfSts | 点位运动到位状态，0：未到位，1：到位。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | | | |

指令 84 GTN_ClearTrapSts

| | | | |
|---------|---|------|---|
| 指令原型 | short GTN_ClearTrapSts(short core, short profile) | | |
| 指令说明 | 清除点位运动到位标志。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 2 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| profile | 规划轴号，正整数。取值范围请参照 指令参数范围 中的“轴”一栏。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | | | |

指令 85 GTN_SetPosEx

| | | | |
|---------|---|------|---|
| 指令原型 | short GTN_SetPosEx(short core, short profile, double pos) | | |
| 指令说明 | 设置点位运动/Jog运动的目标位置 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| profile | 规划轴号，正整数。取值范围请参照 指令参数范围 中的“轴”一栏。 | | |
| pos | 目标位置，单位：pulse | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | | | |

指令 86 GTN_GetPosEx

| | | | |
|---------|---|------|---|
| 指令原型 | short GTN_GetPosEx(short core, short profile, double *pPos) | | |
| 指令说明 | 读取设置的点位运动/Jog运动的目标位置 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| profile | 规划轴号，正整数。取值范围请参照 指令参数范围 中的“轴”一栏。 | | |
| pPos | 目标位置，单位：pulse | | |
| 指令返回值 | 请参照指令返回值列表。 | | |

| | |
|------|---|
| 相关指令 | 无 |
| 指令示例 | |

25. 手轮导引功能

指令 87 GTN_SetCrdMPGMode

| | | | |
|------------|---|------|---|
| 指令原型 | short GTN_SetCrdMPGMode(short core, short crd, short enable, short master, long masterEven, long slaveEven, short filterTime, short mode) | | |
| 指令说明 | 设置手轮导引功能参数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| enable | 手轮导引功能使能，0：不使能，1：使能。 | | |
| master | 手轮主轴号，正整数。 | | |
| masterEven | 手轮主轴比例。 | | |
| slaveEven | 手轮从轴比例。 | | |
| filterTime | 手轮主轴滤波时间，正整数，单位：ms。 | | |
| mode | 手轮导引模式： CRD_MPG_MODE_BIDIRECTION（0）：正负向都可以固定缓冲区模式 CRD_MPG_MODE_POS（1）：MPG正方向转动 正向插补 CRD_MPG_MODE_NEG（-1）：MPG负方向转动 正向插补 CRD_MPG_MODE_WINDOW（2）：正负向都可以滚动缓冲区模式 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | | | |

指令 88 GTN_GetCrdMPGMode

| | | | |
|-------------|---|------|---|
| 指令原型 | short GTN_GetCrdMPGMode(short core, short crd, short *pEnable, short *pMaster, long *pMasterEven, long *pSlaveEven, short *pFilterTime, short *pMode) | | |
| 指令说明 | 读取手轮导引功能参数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号，正整数，取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| pEnable | 手轮导引功能使能，0：不使能，1：使能。 | | |
| pMaster | 手轮主轴号，正整数。 | | |
| pMasterEven | 手轮主轴比例。 | | |
| pSlaveEven | 手轮从轴比例。 | | |
| pFilterTime | 手轮主轴滤波时间，正整数，单位：ms。 | | |
| pMode | 手轮导引模式： CRD_MPG_MODE_BIDIRECTION（0）：正负向都可以固定缓冲区模式 CRD_MPG_MODE_POS（1）：MPG正方向转动 正向插补 | | |

| | |
|-----------------------|---|
| 指令返回值 相关指令 指令示例 | CRD_MPG_MODE_NEG (-1) : MPG负方向转动 正向插补 |
| | CRD_MPG_MODE_WINDOW (2) : 正负向都可以滚动缓冲区模式 |
| | 请参照指令返回值列表。 |
| | 无 |

指令 89 GTN_SetCrdMPGModeEx

| | | | |
|---------|--|------|---|
| 指令原型 | short GTN_SetCrdMPGMode(short core, short crd, TCrdMpgPrm *pMpgPrm) | | |
| 指令说明 | 设置手轮导引功能参数(增强版, 仅R688支持)。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号, 正整数, 取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| pMpgPrm | <p>手轮导引功能参数结构体。</p> <pre>typedef struct { short enable; short master; short filterTime; short mode; short fixedMpgVelCount; short pad1[3]; long masterEven; long slaveEven; long pad2[2]; double fixedMpgVel[MAX_CRDMPG_FIXEDMPGVEL]; double ratioUpdateTime; double pad3[3]; }TCrdMpgPrm;</pre> <p>enable: 手轮导引功能使能, 0: 不使能, 1: 使能。 master: 手轮主轴号, 正整数。 filterTime: 手轮主轴滤波时间, 正整数, 单位: ms。 mode: 手轮导引模式 CRD_MPG_MODE_BIDIRECTION (0) : 正负向都可以固定缓冲区模式 CRD_MPG_MODE_POS (1) : MPG正方向转动 正向插补 CRD_MPG_MODE_NEG (-1) : MPG负方向转动 正向插补 CRD_MPG_MODE_WINDOW (2) : 正负向都可以滚动缓冲区模式 fixedMpgVelCount: 手轮导引固定速度档位个数, 正整数, 取值范围: [0,8]。当 fixedMpgVelCount=0时, 不使用固定档位功能。 masterEven: 手轮主轴比例。 slaveEven: 手轮从轴比例。 fixedMpgVel[MAX_CRDMPG_FIXEDMPGVEL]; : 手轮导引档位 Ratio 值, 当 fixedMpgVelCount>0时, 对应的速度档位ratio必须大于零, 且为递增关系。 ratioUpdateTime: 手轮导引速度倍率刷新时间, 取值范围: [0, 1000], 单位: ms。如果</p> | | |

| | |
|-------|--|
| | ratioUpdateTime=0, 则默认200ms刷新一次, 如果ratioUpdateTime>0, 根据设置的时间刷新倍率。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | |

指令 90 GTN_GetCrdMPGModeEx

| | | | |
|----------|--|------|---|
| 指令原型 | short GTN_GetCrdMPGMode(short core, short crd, short *pFifoEnd, TCrdMpgPrm *pMpgPrm) | | |
| 指令说明 | 读取手轮导引功能参数(增强版, 仅R688支持)。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系号, 正整数, 取值范围请参照 指令参数范围 中的“插补坐标系序号”一栏。 | | |
| pFifoEnd | 插补缓冲区数据结束标志, 0: 未结束, 1: 已结束。 | | |
| pMpgPrm | <p>手轮导引功能参数结构体。</p> <pre>typedef struct { short enable; short master; short filterTime; short mode; short fixedMpgVelCount; short pad1[3]; long masterEven; long slaveEven; long pad2[2]; double fixedMpgVel[MAX_CRDMPG_FIXEDMPGVEL]; double ratioUpdateTime; double pad3[3]; }TCrdMpgPrm;</pre> <p>enable: 手轮导引功能使能, 0: 不使能, 1: 使能。 master: 手轮主轴号, 正整数。 filterTime: 手轮主轴滤波时间, 正整数, 单位: ms。 mode: 手轮导引模式 CRD_MPG_MODE_BIDIRECTION (0): 正负向都可以固定缓冲区模式 CRD_MPG_MODE_POS (1): MPG正方向转动 正向插补 CRD_MPG_MODE_NEG (-1): MPG负方向转动 正向插补 CRD_MPG_MODE_WINDOW (2): 正负向都可以滚动缓冲区模式 fixedMpgVelCount: 手轮导引固定速度档位个数, 正整数, 取值范围: [0,8]。当 fixedMpgVelCount=0时, 不使用固定档位功能。 masterEven: 手轮主轴比例。 slaveEven: 手轮从轴比例。 fixedMpgVel[MAX_CRDMPG_FIXEDMPGVEL]; : 手轮导引档位 Ratio 值, 当</p> | | |

| | |
|-------|--|
| | fixedMpgVelCount>0时，对应的速度档位ratio必须大于零，且为递增关系。 ratioUpdateTime：手轮导引速度倍率刷新时间，取值范围：[0, 1000]，单位：ms。如果ratioUpdateTime=0，则默认200ms刷新一次，如果ratioUpdateTime>0，根据设置的时间刷新倍率。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | 无 |
| 指令示例 | |

26. Event-Task 新增功能

指令 91 GTN_AddTask

| | | | |
|------------|--|------|---|
| 指令原型 | short GTN_AddTask(short core, short taskType, void *pTaskData, short *pTaskIndex) | | |
| 指令说明 | 设置事件触发的任务。（新增保存运控变量任务） | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| taskType | 任务类型。 TASK_SAVE_MC_VAR(50)：保存运控变量 | | |
| pTaskData | 任务相关数据结构体。 保存运控变量结构体 typedef struct { short count; TWatchVar var{TASK_SAVE_MC_VAR_MAX}; }TTaskSaveMcVar; count：该任务需要保存的运控变量个数。 var：该任务需要保存的运控变量信息结构体数组，数组大小为count。 TASK_SAVE_MC_VAR_MAX(5)：该任务最多保存五个运控变量。 | | |
| pTaskIndex | 返回该事件的索引值，正整数，取值范围：[1,32]。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | 无 | | |
| 指令示例 | <pre>// 事件： // 第一路GPI从0变到1时，保存编码器3、编码器4的位置，以及第2路模拟量输入的值 // 事件触发次数为无限次触发 short core,rtn; short eventIndex; short taskIndex; short eventTaskLinkIndex; TEvent event; TTaskSaveMcVar task; core = 1; // 设置保存变量触发事件</pre> | | |

```
memset(&event,0,sizeof(TEvent));
event.loop = 0;//0: 无限次触发
event.var.type = WATCH_VAR_GPI;
event.var.index = 1;
event.condition = WATCH_CONDITION_CHANGE_TO;
event.value = 1;

// 设置需要保存的变量信息
memset(&task,0,sizeof(TTaskSaveMcVar));
task.count = 3;// 需要保存三个变量
task.var[0].type = WATCH_VAR_ENC_POS;
task.var[0].index = 3;
task.var[1].type = WATCH_VAR_ENC_POS;
task.var[1].index = 4;
task.var[2].type = WATCH_VAR_AI;
task.var[2].index = 2;
    rtn = GTN_ClearEvent(core);
if ( 0 != rtn )
{
    return rtn;
}

rtn = GTN_ClearTask(core);
if ( 0 != rtn )
{
    return rtn;
}

rtn = GTN_ClearEventTaskLink(core);
if ( 0 != rtn )
{
    return rtn;
}

rtn = GTN_AddEvent(core,pEvent,&eventIndex);
if ( 0 != rtn )
{
    return rtn;
}

rtn = GTN_AddTask(core,TASK_SAVE_MC_VAR,pTask,&taskIndex);
if ( 0 != rtn )
{
    return rtn;
}

rtn = GTN_AddEventTaskLink(core,eventIndex,taskIndex,&eventTaskLinkIndex);
if ( 0 != rtn )
```



```

{
    return rtn;
}

rtn = GTN_EventOn(core,eventIndex,1);
if ( 0 != rtn )
{
    return rtn;
}
    
```

指令 92 GTN_GetTaskSaveMcVarResult

| | |
|------------|---|
| 指令原型 | short GTN_GetTaskSaveMcVarResult(short core, short taskIndex, TWatchVar *pVar, double *pValue, short count, short *pReadCount) |
| 指令说明 | 读取保存运控变量任务已保存的值。 |
| 指令类型 | 立即指令。 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 |
| taskIndex | 事件的索引值，正整数，取值范围：[1,32]。 |
| pVar | 需要读取的运控变量信息结构体。 |
| pValue | 需要读取的运控变量值数组，数组实际读取到的大小为pReadCount。 |
| count | 需要读取的数据个数。 |
| pReadCount | 返回实际读取的数据个数。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | GTN_AddTask |
| 重点说明 | 该指令读取方式为动态读取，已经读取过的数控制器将不在保存。 |
| 指令示例 | <pre> /***** /* 读取事件已经保存的数据 /***** short count,readCount,sumCount; double encPos3[10],encPos4[10],adc2[10]; short taskType; //先获取任务保存的变量信息 rtn = GTN_GetTask(core,taskIndex,&taskType,&task); if (0 != rtn) { return rtn; } sumCount = 0; //读取保存的前260个数据，每次读取十个数 count = 10; </pre> |

```

for (i=0; i<26; ++i)
{
    rtn =
    GTN_GetTaskSaveMcVarResult(core,taskIndex,&task.var[0],encPos3,count,&readCount);
    if ( 0 != rtn )
    {
        return rtn;
    }

    rtn =
    GTN_GetTaskSaveMcVarResult(core,taskIndex,&task.var[1],encPos4,count,&readCount);
    if ( 0 != rtn )
    {
        return rtn;
    }

    rtn =
    GTN_GetTaskSaveMcVarResult(core,taskIndex,&task.var[1],adc2,count,&readCount);
    if ( 0 != rtn )
    {
        return rtn;
    }

    sumCount += readCount;

    if ( 0 == readCount )
    {
        break;
    }
}

```

27. 编码器相关配置功能

指令 93 GTN_SetEncoderMapRelation

| | | | |
|----------------|---|------|---|
| 指令原型 | short GTN_SetEncoderMapRelation(short core,short masterEncType,short masterEncIndex,short mapEncType,short mapEncIndex) | | |
| 指令说明 | 设置编码器之间的映射关系 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| masterEncType | 主编码器类型。 | | |
| masterEncIndex | 主编码器的索引。 | | |
| mapEncType | 映射编码器的类型。 | | |
| mapEncIndex | 映射编码器的索引。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |

| | |
|------|--|
| 相关指令 | |
| 重点说明 | 控制器默认编码器一一对应，可以通过该指令进行修改，比如：轴编码器的计数来源修改为辅助编码器等。 |
| 指令示例 | //设置轴1的编码器计数来源于辅助编码器1，之后辅助编码器1的计数会累加到轴1的编码器上 short rtn; rtn = GTN_SetEncoderMapRelation(1,MC_ENCODER,1,MC_AU_ENCODER,1); |

指令 94 GTN_GetEncoderMapRelation

| | | | |
|----------------|---|------|---|
| 指令原型 | short GTN_GetEncoderMapRelation(short core,short masterEncType,short masterEncIndex,short *pMapEncType,short *pMapEncIndex) | | |
| 指令说明 | 读取编码器之间的映射关系 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| masterEncType | 主编码器类型。 | | |
| masterEncIndex | 主编码器的索引。 | | |
| pMapEncType | 映射编码器的类型。 | | |
| pMmapEncIndex | 映射编码器的索引。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | 控制器默认编码器一一对应，可以通过该指令进行修改，比如：轴编码器的计数来源修改为辅助编码器等。 | | |
| 指令示例 | 无 | | |

指令 95 GTN_SetEncoderDeltaLimit

| | | | |
|------------|---|------|---|
| 指令原型 | short GTN_SetEncoderDeltaLimit(short core,short encType,short encIndex,long deltaLimit) | | |
| 指令说明 | 设置编码器增量限制 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| encType | 主编码器类型。 | | |
| encIndex | 主编码器的索引。 | | |
| deltaLimit | 增量限制。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | 控制器默认增量式编码器增量限制为 32767，可以通过该指令进行修改，增量式编码器最大只能设置到 32767；绝对式编码器的增量和其线数有关系，无需设置。 | | |
| 指令示例 | 无 | | |

指令 96 GTN_GetEncoderDeltaLimit

| | | | |
|-------------|---|------|---|
| 指令原型 | short GTN_SetEncoderDeltaLimit(short core,short encType,short encIndex,long *pDeltaLimit) | | |
| 指令说明 | 读取编码器增量限制 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 6 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| encType | 主编码器类型。 | | |
| encIndex | 主编码器的索引。 | | |
| pDeltaLimit | 增量限制。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

28. GT 指令按照物理寻址

指令 97 GTN_ReadPhysicalMap

| | | | |
|-------|---|------|---|
| 指令原型 | short GTN_ReadPhysicalMap(void) | | |
| 指令说明 | 初始化物理资源和逻辑资源的关系 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共无参数 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | 该指令要在 使用 ConvertPhysical 功能函数之前调用 | | |
| 指令示例 | 无 | | |

指令 98 ConvertPhysical

| | | | |
|----------|---|------|---|
| 指令原型 | short ConvertPhysical(short core,short dataType,short terminal,short index) | | |
| 指令说明 | 通过物理寻址获取对应的逻辑序号 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共无参数 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| dataType | 资源名称 MC_GPO(12):通用输出资源 MC_GPI(4):通用输入资源 | | |
| terminal | 模块物理站号，正整数 从 0 开始寻址，本站对应的站号为 0 | | |
| index | 模块上相应的资源序号，正整数 从 1 开始寻址 | | |
| 指令返回值 | 1、指令返回的是对应的逻辑资源序号，目前只支持 MC_GPI 和 MC_GPO 两类资源 GTN_SetDoBit(1,MC_GPO,ConvertPhysical(1,MC_GPO,1,1),0x0)//设置核 1 模块 1 的第 1 路 Do 的通用输出为 0 | | |

| | |
|------|-------------------------|
| 相关指令 | 2、如果返回值为-1，则表示该功能函数执行失败 |
| 重点说明 | |
| 指令示例 | 无 |

29. 补偿功能

指令 99 GTN_SetLeadScrewCrossComp

| | | | |
|----------|---|------|---|
| 指令原型 | short GTN_SetLeadScrewCrossComp(short core,short axis,short n,long startPos,long lenPos,long *pCompPos,long *pCompNeg,short link) | | |
| 指令说明 | 加载交叉误差补偿表 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 8 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| axis | 被补偿轴轴号，正整数，取值范围请参照指令参数范围中的“轴”一栏。 | | |
| n | 补偿段数 | | |
| startPos | 补偿开始位置 | | |
| lenPos | 补偿总距离（每个补偿区间长度为： $lenPos/(n-1)$ ） | | |
| pCompPos | 正向补偿表数组地址 | | |
| pCompNeg | 负向补偿表数组地址 | | |
| link | 关联轴，即参考“link”轴进行补偿，正整数，取值范围请参照指令参数范围中的“轴”一栏。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

指令 100 GTN_EnableLeadScrewCrossComp

| | | | |
|-------|--|------|---|
| 指令原型 | short GTN_EnableLeadScrewCrossComp(short core,short axis,short mode) | | |
| 指令说明 | 开启或关闭交叉误差补偿功能 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| axis | 被补偿轴轴号，正整数，取值范围请参照指令参数范围中的“轴”一栏。 | | |
| mode | 0-关闭；1-开启 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

指令 101 GTN_SetTransformOrthogonal

| | |
|------|---|
| 指令原型 | short GT_SetTransformOrthogonal(short core,short index, |
|------|---|

| | | | |
|-------------|--|------|---|
| | TTransformOrthogonal *pOrthogonal) | | |
| 指令说明 | 设置平面坐标系非正交转换功能参数 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 补偿模块，取值范围[1,2] | | |
| pOrthogonal | 非正交转换参数结构体，定义如下： <pre>typedef struct { short source; short enable; short x; short y; double theta; } TTransformOrthogonal;</pre> source: 转换源，MC_PROFILE 为规划，MC_ENCODER 为编码器； enable: 0-不使能，1-使能； x: x 坐标系映射的规划轴，取值范围请参照指令参数范围中的“轴”一栏； y: y 坐标系映射的规划轴，取值范围请参照指令参数范围中的“轴”一栏； theta: x 和 y 之间的夹角，取值范围(0,180)，单位：度。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

指令 102 GTN_GetTransformOrthogonal

| | | | |
|-------------|--|------|---|
| 指令原型 | short GT_GetTransformOrthogonal(short core,short index, TTransformOrthogonal *pOrthogonal) | | |
| 指令说明 | 获取平面坐标系非正交转换功能参数 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照指令参数范围中的“内核”一栏。 | | |
| index | 补偿模块，取值范围[1,2]。 | | |
| pOrthogonal | 非正交转换参数结构体，定义如下： <pre>typedef struct { short source; short enable; short x; short y; double theta; } TTransformOrthogonal;</pre> source: 转换源，MC_PROFILE 为规划，MC_ENCODER 为编码器； enable: 0-不使能，1-使能； | | |

| | |
|-------|---|
| 指令返回值 | x: x 坐标系映射的规划轴, 取值范围请参照 指令参数范围 中的“轴”一栏; y: y 坐标系映射的规划轴, 取值范围请参照 指令参数范围 中的“轴”一栏; theta: x 和 y 之间的夹角, 取值范围(0,180), 单位: 度。 |
| 相关指令 | 请参照指令返回值列表。 |
| 重点说明 | |
| 指令示例 | 无 |

指令 103 GTN_GetTransformOrthogonalPosition

| | | | |
|------------|--|------|---|
| 指令原型 | short GT_GetTransformOrthogonalPosition(short index,double *pPositionX,double *pPositionY) | | |
| 指令说明 | 获取平面坐标系非正交转换后的各轴位置 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| index | 补偿模块, 取值范围[1,2]。 | | |
| pPositionX | X 轴的位置, 单位 pulse。 | | |
| pPositionY | Y 轴的位置, 单位 pulse。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

30. 插补单步执行功能

指令 104 GTN_CrdStepMode

| | | | |
|--------|--|------|-----------|
| 指令原型 | short GTN_CrdStepMode (short core, short mask, short option) | | |
| 指令说明 | 设置插补运动的单步执行模式 | | |
| 指令类型 | 立即指令, 调用后立即生效。 | 章节页码 | 错误!未定义书签。 |
| 指令参数 | 该指令共有 3 个参数, 参数的详细信息如下。 | | |
| core | 内核, 正整数, 取值范围请参照 错误!未找到引用源。 中的“内核”一栏 | | |
| mask | 从 bit0~bit1 按位表示坐标系号。 bit0 对应坐标系 1, bit1 对应坐标系 2。 0: 设置不生效, 1: 设置生效。 即, 0x1 表示设置坐标系 1 的参数生效, 0x2 表示设置坐标系 2 的参数生效, 0x3 表示同时设置坐标系 1 和坐标系 2 的参数生效。 | | |
| option | 单步执行插补段的停止方式。 bit0 对应坐标系 1, bit1 对应坐标系 2。 0: 平滑停止, 1: 紧急停止。 例如, 0x1 表示坐标系 1 单步执行停止时为紧急停止, 坐标系 2 为平滑停止。 | | |
| 指令返回值 | 请参照指令返回值列表 | | |
| 相关指令 | 无。 | | |

指令示例

指令 105 GTN_CrdStartStep

| | | | |
|--------|--|------|-----------|
| 指令原型 | short GTN_CrdStartStep (short core, short mask, short option) | | |
| 指令说明 | 启动插补运动的单步执行，每次调用这条指令，都执行一条插补指令。 | | |
| 指令类型 | 立即指令，调用后立即生效。 | 章节页码 | 错误!未定义书签。 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 错误!未找到引用源。 中的“内核”一栏 | | |
| mask | 从 bit0~bit1 按位表示坐标系号。 bit0 对应坐标系 1， bit1 对应坐标系 2。 0: 设置不生效， 1: 设置生效。 即， 0x1 表示设置坐标系 1 的参数生效， 0x2 表示设置坐标系 2 的参数生效， 0x3 表示同时设置坐标系 1 和坐标系 2 的参数生效。 | | |
| option | 从 bit0~bit1 按位表示坐标系需要单步启动执行的缓存区的编号。 bit0 对应坐标系 1， bit1 对应坐标系 2。 0: 单步启动坐标系中 FIFO0 的插补运动， 1: 单步启动坐标系中 FIFO1 的插补运动。 | | |
| 指令返回值 | 请参照指令返回值列表 | | |
| 相关指令 | 无。 | | |
| 指令示例 | | | |

31. 平滑功能

指令 106 GTN_SetStopSmoothTime

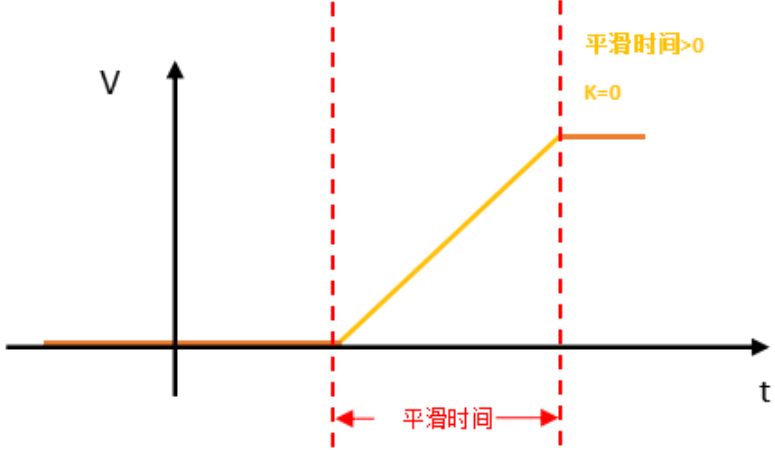
| | | | |
|------------|--|------|---|
| 指令原型 | short GTN_SetStopSmoothTime(short core,short profile,short smoothTime) | | |
| 指令说明 | 设置异常停止的平滑时间 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| profile | 规划器编号，从1开始。 | | |
| smoothTime | 异常停止的平滑时间，取值范围：[0,50]，单位：毫秒； | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

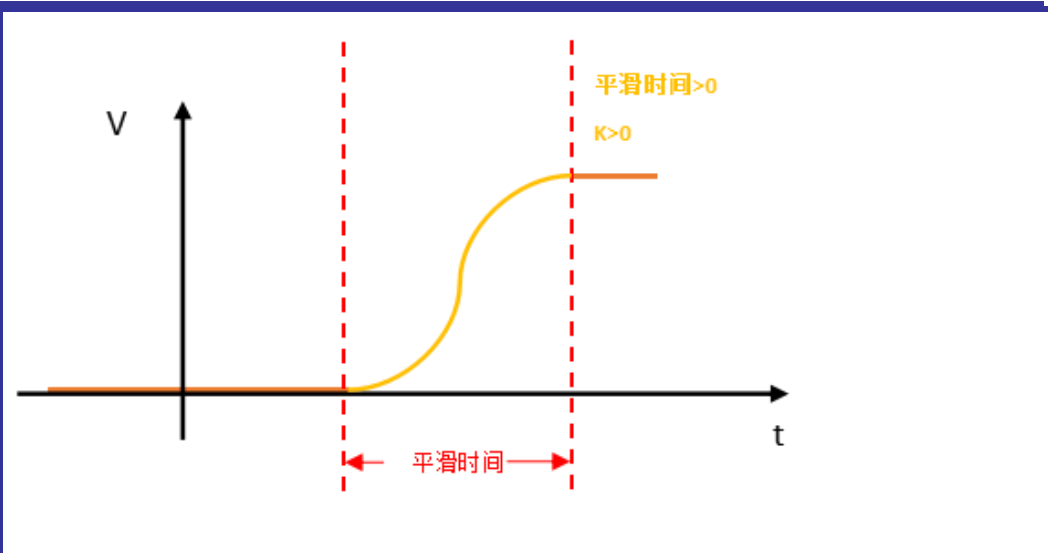
指令 107 GTN_GetStopSmoothTime

| | | | |
|------|--|------|---|
| 指令原型 | short GTN_GetStopSmoothTime(short core,short profile,short *pSmoothTime) | | |
| 指令说明 | 获取异常停止的平滑时间 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 3 个参数，参数的详细信息如下。 | | |

| | |
|-------------|---|
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 |
| profile | 规划器编号，从1开始。 |
| pSmoothTime | 获取到的异常停止的平滑时间，单位：毫秒； |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 无 |

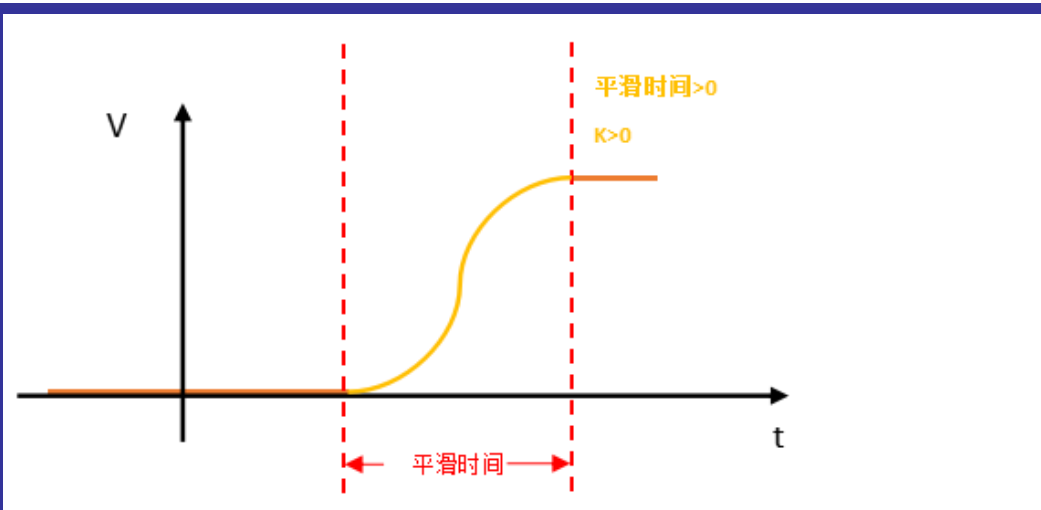
指令 108 GTN_SetAxisMotionSmooth

| | | | |
|-------|--|------|---|
| 指令原型 | short GTN_SetAxisMotionSmooth(short core,short axis,double time,double k) | | |
| 指令说明 | 设置轴的平滑时间和平滑系数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| axis | 轴编号，从1开始。 | | |
| time | 平滑时间，取值范围：[0,60]，单位：ms time 为 0 时，不进行轴平滑。一般建议设置为 20~50ms。 | | |
| k | 平滑系数，取值范围：[0,100)。一般建议设置为 15~18。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | <p>设置平滑时间之前的速度-时间曲线，如图所示：</p>  <p>设置平滑时间之后的速度-时间曲线，如图所示：</p> | | |



指令 109 GTN_GetAxisMotionSmooth

| | |
|-------|--|
| 指令原型 | short GTN_GetAxisMotionSmooth(short core,short axis,double *pTime, double *pK) |
| 指令说明 | 读取轴的平滑时间和平滑系数。 |
| 指令类型 | 立即指令。 章节页码 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 |
| axis | 轴编号，从1开始。 |
| pTime | 平滑时间，取值范围：[0,60]，单位：ms time 为 0 时，不进行轴平滑。一般建议设置为 20~50ms。 |
| pK | 平滑系数，取值范围：[0,100)。一般建议设置为 15~18。 |
| 指令返回值 | 请参照指令返回值列表。 |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | <p>设置平滑时间之前的速度-时间曲线，如图所示：</p> <p>设置平滑时间之后的速度-时间曲线，如图所示：</p> |



指令 110 GTN_SetCrdJerkTime

| | | | |
|----------|---|------|---|
| 指令原型 | short GTN_SetCrdJerkTime(short core, short crd, double jerkTime, double coef) | | |
| 指令说明 | 设置插补运动的平滑时间和平滑系数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系编号，从1开始。 | | |
| jerkTime | 平滑时间，取值范围：[0, 120]，单位：ms time 为 0 时，不进行轴平滑。一般建议设置为 20~50ms。 | | |
| coef | 平滑系数，取值范围：[0,100)。一般建议设置为 15~18。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

指令 111 GTN_GetCrdJerkTime

| | | | |
|-----------|---|------|---|
| 指令原型 | short GTN_GetCrdJerkTime(short core, short crd, double *pJerkTime, double *pCoef) | | |
| 指令说明 | 读取插补运动的平滑时间和平滑系数。 | | |
| 指令类型 | 立即指令。 | 章节页码 | 无 |
| 指令参数 | 该指令共有 4 个参数，参数的详细信息如下。 | | |
| core | 内核，正整数，取值范围请参照 指令参数范围 中的“内核”一栏。 | | |
| crd | 插补坐标系编号，从1开始。 | | |
| pJerkTime | 平滑时间，取值范围：[0, 120]，单位：ms time 为 0 时，不进行轴平滑。一般建议设置为 20~50ms。 | | |
| pCoef | 平滑系数，取值范围：[0,100)。一般建议设置为 15~18。 | | |
| 指令返回值 | 请参照指令返回值列表。 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

32. 模块断线状态及安全模式设置

指令 112 GTN_RN_GetStationOfflineCount

| | | |
|----------------------|---|------|
| 指令原型 | GT_API GTN_RN_GetStationOfflineCount(short cardIndex, long * pStationPhyId, short count, short *pStationOfflineCount) | |
| 指令说明 | 读取从站网络连接状态 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共有 4 个参数 | |
| cardIndex | 卡号，从1开始。 | |
| pStationPhyId | 网络中，断线站的详细 ID 信息，其输入为数组，大小为 count | |
| count | 网络中，接的站的总数，卡是第 0 号站，站的总数为（从站数量+1）。 | |
| pStationOfflineCount | 网络中，断线的站的个数 | |
| 指令返回值 | | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 无 | |

指令 113 GTN_RN_SetStationSafeModeOut

| | | |
|--------------|--|------|
| 指令原型 | GT_API GTN_RN_SetStationSafeModeOut(short cardIndex, short stationPhyId, short type, short index, short *pEnable, double *pValue, short count) | |
| 指令说明 | 设置从站物理资源的安全模式。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共有 7 个参数。 | |
| cardIndex | 卡号，从1开始。 | |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0 。 | |
| type | 从站资源类型，目前支持的资源类型： MC_ENABLE MC_GPO | |
| index | 从站资源的物理索引。 | |
| pEnable | 从站资源的安全模式的使能开关， 0：关闭安全模式，1：打开安全模式，定义的 pEnable 数组大小应和 count 保持一致。 | |
| pValue | 从站进入安全模式后，该资源的期望输出值，定义的 pValue 数组大小应和 count 保持一致。 type = MC_ENABLE 时，0：下使能，1：上使能； type = MC_GPO 时，0：拉高，1：拉低。 | |
| count | 需要设置安全模式的从站物理资源个数。 | |
| 指令返回值 | | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 无 | |

指令 114 GTN_RN_ClearStationSafeModeStatus

| | |
|------|--|
| 指令原型 | GT_API GTN_RN_ClearStationSafeModeStatus(short cardIndex,short stationPhyId) |
|------|--|

| | | |
|--------------|------------------------------------|------|
| 指令说明 | 手动清除从站安全模式状态。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 2 个参数。 | |
| cardIndex | 卡号，从1开始。 | |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0 。 | |
| 指令返回值 | | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 无 | |

指令 115 GTN_RN_SetStationSafeModeControl

| | | |
|--------------|---|------|
| 指令原型 | GT_API GTN_RN_SetStationSafeModeControl(short cardIndex, short stationPhyId, short enable, short clearMode) | |
| 指令说明 | 从站安全模式总开关以及安全模式清除类型。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 4 个参数。 | |
| cardIndex | 卡号，从1开始。 | |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0 。 | |
| enable | 从站安全模式使能开关，0：禁止安全模式，1：使能安全模式。 | |
| clearMode | 从站处于安全模式时，安全模式状态的清除方式， 0：手动清除安全状态，1：自动清除安全状态。 | |
| 指令返回值 | | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 无 | |

指令 116 GTN_RN_IlinkSetSafeModeOut

| | | |
|--------------|---|------|
| 指令原型 | GT_API GTN_RN_IlinkSetSafeModeOut(short cardIndex, short stationPhyId, short modulePhyId, short type, short index, short *pEnable, double *pValue, short count) | |
| 指令说明 | 设置扩展模块物理资源安全模式。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 8 个参数。 | |
| cardIndex | 卡号，从1开始。 | |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0 。 | |
| modulePhyId | 扩展模块的索引，用户接口从 1 开始 | |
| type | 扩展模块资源类型，目前支持的资源类型为： MC_EXT_DO MC_EXT_AO | |
| index | 扩展模块资源的物理索引 | |
| pEnable | 扩展模块资源安全模式的使能开关， 0：关闭安全模式，1：打开安全模式，定义的数组大小应和 count 保持一致。 | |
| pValue | 扩展模块进入安全模式后，该资源的期望输出值，定义的数组大小应和 count 保持一致， type = MC_EXT_DO 时，0：不输出，1：输出； | |

| | |
|---|------------------------------|
| count 指令返回值 相关指令 重点说明 指令示例 | type = MC_EXT_AO 时，根据期望的值输出。 |
| | 需要设置为安全模式的资源个数。 |
| | |
| | |
| | 无 |

指令 117 GTN_RN_IlinkClearSafeModeStatus

| | |
|---------------------|--|
| 指令原型 | GT_API GTN_RN_IlinkClearSafeModeStatus(short cardIndex, short stationPhyId, short modulePhyId) |
| 指令说明 | 扩展模块手动清除从站安全模式状态 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 3 个参数。 |
| cardIndex | 卡号，从1开始。 |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0。 |
| modulePhyId | 扩展模块的索引，用户接口从 1 开始。 |
| 指令返回值 | |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 无 |

指令 118 GTN_RN_IlinkSetSafeModeControl

| | |
|---------------------|--|
| 指令原型 | GT_API GTN_RN_IlinkSetSafeModeControl(short cardIndex, short stationPhyId, short modulePhyId, short enable, short clearMode) |
| 指令说明 | 设置扩展模块物理资源安全模式的总开关以及安全模式清除类型。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 5 个参数 |
| cardIndex | 卡号，从1开始。 |
| stationPhyId | 从站的索引,从 0 开始，控制卡或者有本地资源的控制器站号为 0 |
| modulePhyId | 扩展模块的索引，用户接口从 1 开始 |
| enable | 扩展模块安全模式使能开关，0：禁止安全模式，1：使能安全模式 |
| clearMode | 扩展模块处于安全模式时，安全模式状态的清除方式， 0：手动清除安全状态，1：自动清除安全状态。 |
| 指令返回值 | |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 无 |

33. 网络恢复指令

指令 119 GTN_RN_Recover

| | |
|-------------|--|
| 指令原型 | GT_API GTN_RN_Recover(short cardIndex) |
| 指令说明 | 网络恢复指令。 |

| | | | |
|-----------|------------|------|--|
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 1 个参数 | | |
| cardIndex | 卡号，从1开始。 | | |
| 指令返回值 | | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 无 | | |

34. 批量读取控制器状态

指令 120 GTN_GetMcVarArray

| | |
|--------------|--|
| 指令原型 | GT_API GTN_GetMcVarArray(short core, short resInfoCount, TMcVarResInfo *pResInfo) |
| 指令说明 | 批量读取控制器状态数据。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 3 个参数 |
| core | 核号，从1开始。 |
| resInfoCount | 资源个数，该值为pResInfo数组的大小 读取控制器的资源状态的结构体数组 |
| pResInfo | <pre>struct typedef { short resType; short resIndex; short resSubIndex; short resCount; short resValue[256]; }TMcVarResInfo;</pre> <p>resType: 设置资源的类型，填watch功能支持采集的类型 resIndex: 设置资源的逻辑索引 resSubIndex: 设置资源的二级逻辑索引 resValue: 获取的资源状态数据</p> <p>resType可填的类型有：（即watch功能支持的类型）</p> <pre>//MC 时钟，以ms 为单位 #define WATCH_VAR_CLOCK (1200) //MC 时钟，以规划周期为单位（us） #define WATCH_VAR_PRF_LOOP (1201) // Profile 规划位置 #define WATCH_VAR_PRF_POS (6000) // Profile 规划速度 #define WATCH_VAR_PRF_VEL (6001) // Profile 规划加速度 #define WATCH_VAR_PRF_ACC (6002) // Profile 运动状态 #define WATCH_VAR_PRF_RUN (6200)</pre> |

```
// 插补运动合成规划位置
#define WATCH_VAR_CRD_PRF_POS (8000)
// 插补运动合成规划速度
#define WATCH_VAR_CRD_PRF_VEL (8001)
// 插补运动合成规划加速度
#define WATCH_VAR_CRD_PRF_ACC (8002)
// 插补运动状态
#define WATCH_VAR_CRD_RUN (8200)
// 插补段号
#define WATCH_VAR_CRD_SEGMENT_NUMBER (8202)
// 插补用户段号
#define WATCH_VAR_CRD_SEGMENT_NUMBER_USER (8203)
// 插补接收指令
#define WATCH_VAR_CRD_COMMAND_RECEIVE (8204)
// 插补执行指令
#define WATCH_VAR_CRD_COMMAND_EXECUTE (8205)
// SCAN 合成规划位置
#define WATCH_VAR_SCAN_PRF_POS (18000)
// SCAN 合成规划速度
#define WATCH_VAR_SCAN_PRF_VEL (18001)
// SCAN 合成规划加速度
#define WATCH_VAR_SCAN_PRF_ACC (18002)
// SCAN 的X 轴规划位置
#define WATCH_VAR_SCAN_PRF_POS_X (18010)
// SCAN 的Y 轴规划位置
#define WATCH_VAR_SCAN_PRF_POS_Y (18020)
// SCAN 运动状态
#define WATCH_VAR_SCAN_RUN (18200)
// SCAN 段号
#define WATCH_VAR_SCAN_SEG_NUMBER (18202)
// 激光HSIO
#define WATCH_VAR_LASER_HSIO (18600)
// 激光能量
#define WATCH_VAR_LASER_POWER (18601)
// AXIS 规划位置
#define WATCH_VAR_AXIS_PRF_POS (20000)
// AXIS 规划速度
#define WATCH_VAR_AXIS_PRF_VEL (20001)
// AXIS 规划加速度
#define WATCH_VAR_AXIS_PRF_ACC (20002)
// 编码器位置
#define WATCH_VAR_ENC_POS (30000)
// GPI
#define WATCH_VAR_GPI (31000)
// GPO
#define WATCH_VAR_GPO (32000)
```


| | |
|-------|--|
| | <pre>// 捕获状态 #define WATCH_VAR_TRIGGER_STATUS (38001) // 位置环跟随误差 #define WATCH_VAR_POS_LOOP_ERROR (40000) // Watch 采集次数 #define WATCH_VAR_WATCH_TIME</pre> |
| 指令返回值 | |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 无 |

35. 串行通讯指令（本地 485/232 通信指令）

本章节的 485 / 232 通信功能是通过 FPGA 实现的，非控制器系统(Windows 系统)的 232 通信。对于等环网控制器，也可以使用 35. 串行通讯指令（通用 485/232 通信指令）章节的指令。

指令 121 GT_RN_ComOpen

| | |
|-------|---|
| 指令原型 | short GT_RN_ComOpen(short index) |
| 指令说明 | 打开FPGA转串口模块。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 1 个参数 |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | GT_RN_ComClose |
| 重点说明 | |
| 指令示例 | 例程 5-6: |

指令 122 GT_RN_ComClose

| | |
|-------|---|
| 指令原型 | short GT_RN_ComClose(short index) |
| 指令说明 | 关闭FPGA转串口模块。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 1 个参数 |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | GT_RN_ComOpen |
| 重点说明 | |
| 指令示例 | 例程 5-6: |

指令 123 GT_RN_ComRead

| | |
|------|--|
| 指令原型 | short GT_RN_ComRead(short index, unsigned long readLen, unsigned long* pResLen, unsigned char * pData) |
| 指令说明 | 从串口读取数据。 |

| | | | |
|---------|--------------------------------|------|--|
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 4 个参数 | | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | | |
| readLen | 预计读取的数据长度 (byte)。 | | |
| pResLen | 实际读取的数据长度 (byte)。 | | |
| pData | 读取的数据的指针首地址。 | | |
| 指令返回值 | 参照指令返回值列表 | | |
| 相关指令 | GT_RN_ComWrite | | |
| 重点说明 | | | |
| 指令示例 | 例程 5-6: | | |

指令 124 GT_RN_ComWrite

| | | | |
|----------|---|------|--|
| 指令原型 | short GT_RN_ComWrite(short index, unsigned long writeLen , unsigned long* pResLen, unsigned char * pData) | | |
| 指令说明 | 从串口发送数据。 | | |
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 4 个参数 | | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | | |
| writeLen | 预计发送的数据长度 (byte)。 | | |
| pResLen | 实际发送的数据长度 (byte)。 | | |
| pData | 发送的数据的指针首地址。 | | |
| 指令返回值 | 参照指令返回值列表 | | |
| 相关指令 | GT_RN_ComRead | | |
| 重点说明 | | | |
| 指令示例 | 例程 5-6: | | |

指令 125 GT_RN_ComGetState

| | | | |
|--------|---|------|--|
| 指令原型 | short GT_RN_ComGetState(short index, unsigned char * pState) | | |
| 指令说明 | 获取串口状态。 | | |
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 2 个参数 | | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | | |
| pState | 获取的状态值指针地址，返回值的具体含义如下： bit[3:0]错误指示；bit[5:4]发送接收缓存区状态 Bit0: 奇偶校验错误位，1表示有错误，0表示无错误 Bit1: 停止位错误位，1表示有错误，0表示无错误 Bit2: 接收buffer溢出标志位，1表示有错误，0表示无错误 Bit3: 发送缓冲区溢出，1表示有错误，0表示无错误 Bit4: 发送缓存区状态指示，1 - 满此时不能够在向发送缓冲区写数据，0 - 缓冲区有空余，可以写数据 Bit5: 接收缓冲区状态指示，- 缓冲区内部有有效数据，可以读取，0 - 缓冲区内部没有数据，不可以读取 | | |
| 指令返回值 | 参照指令返回值列表 | | |

| | |
|------|---------|
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 例程 5-6: |

指令 126 GT_RN_ComSetSettings

| | | |
|----------|---|------|
| 指令原型 | short GT_RN_ComSetSettings(short index, unsigned long baudrate, unsigned char stopBits, unsigned char parity) | |
| 指令说明 | 设置串口参数。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 4 个参数 | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | |
| baudrate | 波特率，默认9600，支持100、300、600、1200、2400、4800、9600、19200、57600、115200 | |
| stopBits | stopbits为停止位 参数0为一个停止位，参数2为两个停止位 | |
| parity | parity为校验位 参数 0 为无校验，参数 1 为 ODD 校验，参数 2 为 EVEN 校验 | |
| 指令返回值 | 参照指令返回值列表 | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 例程 5-6: | |

指令 127 GT_RN_ComClearErr

| | | |
|-------|--|------|
| 指令原型 | short GT_RN_ComClearErr(short index, unsigned char flag) | |
| 指令说明 | 清除串口状态。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 2 个参数 | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | |
| flag | 清除错误状态 参数1清除发送buffer，参数2清除接收buffer other无效 | |
| 指令返回值 | 参照指令返回值列表 | |
| 相关指令 | | |
| 重点说明 | | |
| 指令示例 | 例程 5-6: | |

指令 128 GT_RN_ComSetMode

| | | |
|---------|---|------|
| 指令原型 | short GT_RN_ComSetMode(short index, unsigned short comMode) | |
| 指令说明 | 设置串口电平模式。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 2 个参数 | |
| index | 卡号，取值从0开始。该版本目前只支持index=0。 | |
| comMode | 电平参数，默认值为0 | |

| | |
|-------|--------------------------------------|
| | 参数意义如下： 0-RS232, 1-RS485, 2-RS422 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 例程 5-6: |

36. 串行通讯指令（通用 485/232 通信指令）

本章节的 485/232 通信功能是通过 FPGA 实现的，非控制器系统(Windows 系统)的 232 通信，适用于本地和远程的通信。

gt_rn.dll 的版本号从 2021, 08, 23, 01 开始支持对本地的串口操作。gt_rn.dll 的版本号可以通过点击文件，右键查看属性中查看详细信息。

指令 129 GTN_RN_SerialComOpen

| | |
|--------------|---|
| 指令原型 | short GTN_RN_SerialComOpen(short cardIndex, short stationphyId, short comIndex) |
| 指令说明 | 打开FPGA转串口模块。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 3 个参数 |
| cardIndex | 卡号，取值从1开始。 |
| stationphyId | 站号，取值范围[0..64]。 |
| comIndex | 串口号，取值范围[1..4]。 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | GTN_RN_SerialComClose |
| 重点说明 | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） |

指令 130 GTN_RN_SerialComClose

| | |
|--------------|--|
| 指令原型 | short GTN_RN_SerialComClose(short cardIndex, short stationphyId, short comIndex) |
| 指令说明 | 关闭FPGA转串口模块。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 3 个参数 |
| cardIndex | 卡号，取值从1开始。 |
| stationphyId | 站号，取值范围[0..64]。 |
| comIndex | 串口号，取值范围[1..4]。 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | GTN_RN_SerialComOpen |
| 重点说明 | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） |

指令 131 GTN_RN_SerialComRead

| | | |
|--------------|---|------|
| 指令原型 | short GTN_RN_SerialComRead(short cardIndex, short stationphyId, short comIndex, unsigned long readLen, unsigned long* pResLen, unsigned char * pData) | |
| 指令说明 | 从串口读取数据。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 6 个参数 | |
| cardIndex | 卡号，取值从1开始。 | |
| stationphyId | 站号，取值范围[0..64]。 | |
| comIndex | 串口号，取值范围[1..4]。 | |
| readLen | 预计读取的数据长度 (byte)。 | |
| pResLen | 实际读取的数据长度 (byte)。 | |
| pData | 读取的数据的指针首地址。 | |
| 指令返回值 | 参照指令返回值列表 | |
| 相关指令 | GTN_RN_SerialComWrite | |
| 重点说明 | | |
| 指令示例 | 例程 5-7: 485串行通讯 (通用485/232通讯指令) | |

指令 132 GTN_RN_SerialComWrite

| | | |
|--------------|---|------|
| 指令原型 | short GTN_RN_SerialComWrite(short cardIndex, short stationphyId, short comIndex, unsigned long writeLen, unsigned long* pResLen, unsigned char * pData) | |
| 指令说明 | 从串口发送数据。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 6 个参数 | |
| cardIndex | 卡号，取值从1开始。 | |
| stationphyId | 站号，取值范围[0..64]。 | |
| comIndex | 串口号，取值范围[1..4]。 | |
| writeLen | 预计发送的数据长度 (byte)。 | |
| pResLen | 实际发送的数据长度 (byte)。 | |
| pData | 发送的数据的指针首地址。 | |
| 指令返回值 | 参照指令返回值列表 | |
| 相关指令 | GTN_RN_SerialComRead | |
| 重点说明 | | |
| 指令示例 | 例程 5-7: 485串行通讯 (通用485/232通讯指令) | |

指令 133 GTN_RN_SerialComGetState

| | | |
|------|--|------|
| 指令原型 | short GTN_RN_SerialComGetState(short cardIndex, short stationphyId, short comIndex, unsigned char *pState) | |
| 指令说明 | 获取串口状态。 | |
| 指令类型 | 立即指令。 | 章节页码 |
| 指令参数 | 该指令共 4 个参数 | |

| | |
|--------------|---|
| cardIndex | 卡号，取值从1开始。 |
| stationphyId | 站号，取值范围[0..64]。 |
| comIndex | 串口号，取值范围[1..4]。 |
| pState | 获取的状态值指针地址，返回值的具体含义如下： bit[3:0]错误指示；bit[5:4]发送接收缓存区状态 Bit0:奇偶校验错误位，1表示有错误，0表示无错误 Bit1: 停止位错误位，1表示有错误，0表示无错误 Bit2:接收buffer溢出标志位，1表示有错误，0表示无错误 Bit3: 发送缓冲区溢出，1表示有错误，0表示无错误 Bit4: 发送缓存区状态指示，1 - 满此时不能够在向发送缓冲区写数据，0 - 缓冲区有空余，可以写数据 Bit5: 接收缓冲区状态指示，- 缓冲区内部有有效数据，可以读取，0 - 缓冲区内部没有数据，不可以读取 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） |

指令 134 GTN_RN_SerialComSetSettings

| | |
|--------------|---|
| 指令原型 | short GTN_RN_SerialComSetSettings(short cardIndex, short stationphyId, short comIndex, unsigned long baudrate,unsigned char stopBits, unsigned char parity) |
| 指令说明 | 设置串口参数。 |
| 指令类型 | 立即指令。 章节页码 |
| 指令参数 | 该指令共 4 个参数 |
| cardIndex | 卡号，取值从1开始。 |
| stationphyId | 站号，取值范围[0..64]。 |
| comIndex | 串口号，取值范围[1..4]。 |
| baudrate | 波特率，默认9600，支持100、300、600、1200、2400、4800、9600、19200、57600、115200 |
| stopBits | stopbits为停止位 参数0为一个停止位，参数2为两个停止位 |
| parity | parity为校验位 参数0为无校验，参数1为ODD校验，参数2为EVEN校验 |
| 指令返回值 | 参照指令返回值列表 |
| 相关指令 | |
| 重点说明 | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） |

指令 135 GTN_RN_SerialComClearErr

| | |
|------|---|
| 指令原型 | short GTN_RN_SerialComClearErr(short cardIndex, short stationphyId, short comIndex, unsigned char flag) |
| 指令说明 | 清除串口状态。 |

| | | | |
|--------------|---|------|--|
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 4 个参数 | | |
| cardIndex | 卡号，取值从1开始。 | | |
| stationphyId | 站号，取值范围[0..64]。 | | |
| comIndex | 串口号，取值范围[1..4]。 | | |
| flag | 清除错误状态 参数1清除发送buffer，参数2清除接收buffer other无效 | | |
| 指令返回值 | 参照指令返回值列表 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） | | |

指令 136 GTN_RN_SerialComSetMode

| | | | |
|--------------|--|------|--|
| 指令原型 | short GTN_RN_SerialComSetMode(short cardIndex, short stationphyId, short comIndex, unsigned short comMode) | | |
| 指令说明 | 设置串口电平模式。 | | |
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 4 个参数 | | |
| cardIndex | 卡号，取值从1开始。 | | |
| stationphyId | 站号，取值范围[0..64]。 | | |
| comIndex | 串口号，取值范围[1..4]。 | | |
| comMode | 电平参数，默认值为0。 参数意义如下： 0-RS232, 1-RS485, 2-RS422 | | |
| 指令返回值 | 参照指令返回值列表 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） | | |

指令 137 GTN_RN_SerialComGetRecvFifoCnt

| | | | |
|--------------|--|------|--|
| 指令原型 | short GTN_RN_SerialComGetRecvFifoCnt(short cardIndex, short stationPhyId, short comIndex, short* pCount) | | |
| 指令说明 | 获取串口接收缓冲区数据数量。 | | |
| 指令类型 | 立即指令。 | 章节页码 | |
| 指令参数 | 该指令共 4 个参数 | | |
| cardIndex | 卡号，取值从1开始。 | | |
| stationphyId | 站号，取值范围[0..64]。 | | |
| comIndex | 串口号，取值范围[1..4]。 | | |
| pCount | 读取的串口接收缓冲区数据的数量(Byte)。 | | |
| 指令返回值 | 参照指令返回值列表 | | |
| 相关指令 | | | |
| 重点说明 | | | |
| 指令示例 | 例程 5-7: 485串行通讯（通用485/232通讯指令） | | |

五、例程

1、例程 5-1：重频模式例程

如果激光器是使用上升沿触发的外控模式控制出光，使用 PSO 功能时，如果 PSO 的脉冲输出的上升沿和激光器的固定频率的上升沿不在同一时刻，则 PSO 的输出会丢失，出现 PSO 输出不均匀的现象。

使用 PSO 重频功能可以避免这样问题，PSO 重频功能会自动根据激光器的固定频率调整 PSO 的输出频率，确保 PSO 的输出均匀。

激光器的固定频率可以通过软件设置（内部重频模式），也可以通过外部激光器信号接入（外部重频模式）。具体设置详见 `GTN_SetPosComparePsoSyncPrm` 指令。

```
// 重频模式例程
// 设置PSO内部重频模式，内部重频频率为kHz
// ... ..
// 打开运动控制器
// ... ..
short sRtn;
// 循环变量
int i;
// 设置控制权的变量
short permit;
// 位置比较输出模式结构体
TPosCompareMode mode;
TPosComparePsoPrm psoPrm;

// 设置控制权
permit = 0x2;
// 设置第一路HSO为第一路位置比较输出模式
sRtn = GTN_SetTerminalPermitEx(core,
                                1,           // 模块栈号
                                MC_HSO,     // MC_HSO高速io输出
                                &permit,    // 设置第一路位置比较输出有效bit1：第一路位置
                                比较输出，bit2第二路位置比较输出，0：无效1：有效
                                1,           // 设置起始硬件信号输出通道索引为
                                1);         // 设置的硬件信号输出通道个数为

// 设置第一路GPO为第一路位置比较输出模式
sRtn = GTN_SetTerminalPermitEx(core,1, MC_GPO, &permit, 1, 1);

// 停止位置比较输出
sRtn = GTN_PosCompareStop(core, index);

// 清空位置比较输出数据
sRtn = GTN_PosCompareClear(core, index);
```



```
TPosCompareMode posCompareMode;
sRtn = GTN_GetPosCompareMode(1, 1, &posCompareMode);

posCompareMode.dimension = 1;          // 一维线性位置比较输出
posCompareMode.mode = 2;               // 0: FIFO模式 1:linear 2:等间距输出模式
posCompareMode.outputMode = 0;        // 0: 脉冲
posCompareMode.outputPulseWidth = 20; // 设置脉冲宽度
posCompareMode.sourceMode = 1;        // 0: 编码器 1: 规划器
posCompareMode.sourceX = 1;           // x轴对应轴
posCompareMode.sourceY = 1;           // x轴对应轴
posCompareMode.outputCounter = 1;
sRtn = GTN_SetPosCompareMode(1, 1, &posCompareMode);

// 设置PSO内部重频模式, 内部重频频率为100kHz
// 如果是PSO外部重频模式, 需要接入外部重频信号(激光器)
short posSyncMode;
posSyncMode = POS_COMPARE_MODE_PSO_SYNC_INTERNAL_SIGNAL;

sRtn = GTN_SetPosComparePsoSyncPrm(1, 1, posSyncMode, 100);

sRtn = GTN_GetPosComparePsoPrm(1, 1, &psoPrm);
psoPrm.count = 1;
psoPrm.syncPos = 112; // 重频功能开启后, PSO输出频率由8.928kHz调整为9.09kHz

sRtn = GTN_SetPosComparePsoPrm(1, 1, &psoPrm);
// 启动位置比较功能
sRtn = GTN_PosCompareStart(1, 1);

// 将轴设置为Jog模式
TJogPrm jog;

sRtn = GTN_ZeroPos(1, 1, 8);

sRtn = GTN_PrJog(1, 1);

sRtn = GTN_GetJogPrm(1, 1, &jog);

jog.acc = 1000;
jog.dec = 1000;
sRtn = GTN_SetJogPrm(1, 1, &jog);

sRtn = GTN_SetVel(1, 1, 1000);

// 启动轴的Jog运动
sRtn = GTN_Update(1, 1);
```

2、例程 5-2：设置驱动器为周期同步速度模式

```

// *****
// 设置驱动器为周期同步速度模式，并进行点位运动
// *****
void SetAxisErrorStsLink (void)
{
    short rtn;
    short core = 1, axis = 1;
    //开卡
    rtn = GTN_Open(5,2);
    //控制卡重置
    rtn = GTN_Reset(1);
    //取消轴报警
    rtn = GTN_AlarmOff(core,axis);
    //取消轴限位
    rtn = GTN_LmtsOff(core,axis);
    //清除轴状态
    rtn = GTN_ClrSts(core,axis);
    //清除位置信息
    rtn = GTN_ZeroPos(core,axis);
    short motionMode,mode;
    mode = 0;
    //设置控制卡环路模式
    rtn = GTN_CtrlMode(core,axis,mode);//mode=0，控制卡上轴设置为位置环，mode=1控制卡设置为脉冲
    模式
    //读取驱动器环路模式
    rtn = GTN_GetMotionMode(core,axis,&motionMode);
    motionMode = 6;
    //设置驱动器环路模式
    rtn = GTN_SetMotionMode(core,axis, motionMode);// motionMode= 6 驱动器设置为周期同步速度模式，
    motionMode = 5 驱动器设置为周期同步位置模式；当motionMode= 6时 mode必须为0，当motionMode=5
    时，mode必须为1。
    rtn = GTN_GetMotionMode(core,axis,&motionMode);
    //pid参数，根据实际需要调试
    TPid pid;
    //读取一组PID
    rtn = GTN_GetPid(core,axis,1,&pid);
    UpdateData(true);
    pid.kp = 0.05;
    pid.limit = 100;
    //设置一组PID
    rtn = GTN_SetPid(core,axis,1,&pid);

```

```

//设置axis轴为点位运动模式
rtn = GTN_PrftTrap(core,axis);
//点位运动参数
    TTrapPrm trap;
//读取axis轴点位运动参数
rtn = GTN_GetTrapPrm(core,axis, &trap);
// 设置需要修改的运动参数
trap.acc = 0.25;
trap.dec = 0.125;
trap.smoothTime = 25;
// 设置点位运动参数
rtn = GTN_SetTrapPrm(core,axis, &trap);
    long pos;
pos = 1000;
// 设置AXIS轴的目标位置
rtn = GTN_SetPos(core,axis, pos);
// 设置AXIS轴的目标速度
    double vel;
    vel =50;
rtn = GTN_SetVel(core,axis, vel);
// AXIS轴的上使能
rtn = GTN_AxisOn(core,axis);
// 启动AXIS轴的运动
rtn = GTN_Update(1,1<<(axis-1));
long sts;
double prfPos;
do
{
    // 读取AXIS轴的状态
    rtn = GTN_GetSts(1,1, &sts);
    // 读取AXIS轴的规划位置
    rtn = GTN_GetPrfPos(1,1, &prfPos);
    printf("sts=0x%-10lprfPos=%-10.1lf\r", sts, prfPos);
} while(sts&0x400);// 等待AXIS轴规划停止
}

```

3、例程 5-3：设置驱动器为周期同步速度模式，并通过 setdac 控制运动。

```

// *****
// 设置驱动器为周期同步速度模式，并通过 setdac 控制运动
// *****

short rtn;
short motionMode;
short axis;
axis = 1;
//控制卡开卡并初始化
.....

```

```

//目标轴下伺服使能
rtn = GTN_AxisOff(1,axis);
//控制卡上轴设置为开环
rtn = GTN_CtrlMode(1,axis,1);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);
//驱动器设置为周期同步速度模式
motionMode = 6 ;
rtn = GTN_SetMotionMode(1,axis,6);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);
//周期同步速度模式下的目标速度参数（假设驱动器最大速度为5000rpm）
short dacValue;
dacValue = 327; //目标速度为327*5000/32767=49.897rpm
//目标轴上伺服使能
rtn = GTN_AxisOn(1,axis);
//设置目标速度，参数范围[-32767,32767],当dacValue为32767时，目标速度达到驱动器设置的最大速度
rtn = GTN_SetDac(1,axis,&dacValue);

//等待运动完成
.....

dacValue = 0;
//设置目标速度（dacValue设为0时表示目标轴停止运动）
rtn = GTN_SetDac(1,axis,&dacValue);

```

4、例程 5-4：设置驱动器为周期同步速度模式，并通过 SetDrvPrfVel 控制运动。

```

// *****
// 设置驱动器为周期同步速度模式，并通过 SetDrvPrfVel 控制运动
// *****

short rtn;
short motionMode;
short axis;
axis = 1;
//控制卡开卡并初始化
....
//目标轴下伺服使能
rtn = GTN_AxisOff(1,axis);
//控制卡上轴设置为开环
rtn = GTN_CtrlMode(1,axis,1);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);
//驱动器设置为周期同步速度模式
motionMode = 6 ;
rtn = GTN_SetMotionMode(1,axis,6);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);

```

```

//周期同步速度模式下的目标速度参数（假设驱动器最大速度为5000rpm）
long Value;
Value = 10000; //目标速度为10000*5000/16777215=2.980rpm
//目标轴上伺服使能
rtn = GTN_AxisOn(1,axis);
//设置目标速度，参数范围[-16776704, 16776704],当Value为16776704时，目标速度达到驱动器设置的
最大速度
rtn = GTN_SetDrvPrfVel(1,axis,&Value);

//等待运动完成
.....

dacValue = 0;
//设置目标速度（Value设为0时表示目标轴停止运动）
rtn = GTN_SetDrvPrfVel(1,axis,&Value);

```

5、例程 5-5：设置驱动器为周期同步力矩模式，并通过 GTN_SetPrfTorque 控制运动。

```

// *****
// 设置驱动器为周期同步力矩模式，并通过 GTN_SetPrfTorque 控制运动
// *****

short rtn;
short motionMode;
short axis;
axis = 1;
//控制卡开卡并初始化
....
//目标轴下伺服使能
rtn = GTN_AxisOff(1,axis);
//控制卡上轴设置为开环
rtn = GTN_CtrlMode(1,axis,1);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);
//驱动器设置为周期同步力矩模式
motionMode = 7 ;
rtn = GTN_SetMotionMode(1,axis, motionMode);
//读取驱动器模式
rtn = GTN_GetMotionMode(1,axis,&motionMode);
//周期同步速度模式下的目标速度参数（假设驱动器最大速度为5000rpm）
double prfTorqueNM;           // 力矩N (N/m)
double current2Torque=0.4;    // 力矩系数，驱动器电机参数
double ratedCurrent=2.5;     // 额定电流，驱动器电机参数
short prfTorque;              //输入参数
short pAtI_Torque;           //读取输入的参数
prfTorque = 1000 * prfTorqueNM / current2Torque / ratedCurrent;
//目标轴上伺服使能
rtn = GTN_AxisOn(1,axis);

```

```

//设置目标力矩。输入参数需经过转换公式计算
rtn = GTN_SetPrfTorque(core,profile,prfTorque);
//读取目标力矩，检查是否设置成功
rtn = GTN_GetAtlTorque(core,profile,&pAtlTorque);

//等待运动完成
.....

prfTorque = 0;
//设置目标力矩（prfTorque设为0时表示目标轴停止运动）
rtn = GTN_SetPrfTorque(core,profile, prfTorque);

```

6、例程 5-6：485 串行通讯（本地 485/232 通讯指令）

```

// *****
// 485串行通讯（本地485/232通讯指令）使用示例
// *****
#include "gt_com.h"
// 打开串口
rtn = GT_RN_ComOpen(index);

// set mode, (* 0-RS232, -RS485, -RS422 *)
rtn = GT_RN_ComSetMode(index,1);

// set com
rtn = GT_RN_ComSetSettings(index, 115200, 0, 0);

// 产生随机数，发送数据
unsigned char data[20];
for(int i = 0; i < 10; i++)
{
    data[i] = (unsigned char)rand();
    printf("|%X ",data[i]);
}
unsigned long len;
rtn = GT_RN_ComWrite(index, 10, &len, data);

// 读数据
rtn = GT_RN_ComRead(index,10,&res_len,data);
if(res_len > 1)
{
    // 处理接收到的数据
}

```

7、例程 5-7：485 串行通讯（通用 485/232 通讯指令）

```

// *****

```

```
// 485串行通讯（通用485/232通讯指令）使用示例
// *****
#include "gt_ringnet.h"

short cardIndex = 1;
short stationPhyId = 0;
short comIndex = 1;
// 打开串口
rtn = GTN_RN_SerialComOpen(cardIndex, stationPhyId, comIndex);

// set mode, (* 0-RS232, -RS485, -RS422 *)
rtn = GTN_RN_SerialComSetMode(cardIndex, stationPhyId, comIndex,1);

// set com
rtn = GTN_RN_SerialComSetSettings(cardIndex, stationPhyId, comIndex, 115200, 0, 0);

// 产生随机数，发送数据
unsigned char data[20];
for(int i = 0; i < 10; i++)
{
    data[i] = (unsigned char)rand();
    printf("|%X ",data[i]);
}
unsigned long len;
rtn = GTN_RN_SerialComWrite(cardIndex, stationPhyId, comIndex, 10, &len, data);

// 读数据
rtn = GTN_RN_SerialComRead(cardIndex, stationPhyId, comIndex,10,&res_len,data);
if(res_len > 1)
{
    // 处理接收到的数据
}
```