PCMCAT .NET API

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 TrioMotion Namespace Reference

Namespaces

- namespace PCMCAT

4.2 TrioMotion.PCMCAT Namespace Reference

Data Structures

- class PCMCAT_API

  Implements the PC-MCAT shared memory API.

Chapter 5

Data Structure Documentation

5.1 PCMCAT_API.CallbackData Class Reference

Data type for the callback function.

Properties

- CallbackType Type `[get]`
  Type of data stored in the callback_union.
- String MessageData `[get]`
  Convert the ANSI callback data to a string.
- EventParameters EventData `[get]`
  Convert the ANSI callback data to a string.

5.1.1 Detailed Description

Data type for the callback function.

5.1.2 Property Documentation

5.1.2.1 EventData

```
EventParameters EventData  [get]
```

Convert the ANSI callback data to a string.

5.1.2.2 MessageData

```
String MessageData  [get]
```

Convert the ANSI callback data to a string.

5.1.2.3 Type

```
CallbackType Type  [get]
```

Type of data stored in the callback_union.

The documentation for this class was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.2 PCMCAT_API.EventParameters Struct Reference

Event registration definition. Not all parameters are used by all event types.

Data Fields

- Event EventType
  Event type.
- int Axis
  Motion event axis. -1 means all axes.
- int ChannelStart
  Input event start channel.
- int ChannelStop
  Input event stop channel.

5.2.1 Detailed Description

Event registration definition. Not all parameters are used by all event types.

5.2.2 Field Documentation

5.2.2.1 Axis

```
int Axis
```

Motion event axis. -1 means all axes.

5.2.2.2 ChannelStart

```
int ChannelStart
```

Input event start channel.

5.2.2.3 ChannelStop

```
int ChannelStop
```

Input event stop channel.

5.2.2.4 EventType

```
Event EventType
```

Event type.

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.3 PCMCAT_API.ObjectFrame Struct Reference

Robot object frame.

Data Fields

- double x
  Coordinates.
- double y
- double z
- double u
- double v
- double w

5.3.1 Detailed Description

Robot object frame.

5.3.2 Field Documentation

5.3.2.1 u

```
double u
```

5.3.2.2 v

```
double v
```

5.3.2.3 w

```
double w
```

5.3.2.4 x

```
double x
```

Coordinates.

5.3.2.5 y

```
double y
```

5.3.2.6 z

```
double z
```

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.4 PCMCAT_API Class Reference

Implements the PC-MCAT shared memory API.

Data Structures

- class CallbackData
  Data type for the callback function.
- struct EventParameters
  Event registration definition. Not all parameters are used by all event types.
- struct ObjectFrame
  Robot object frame.
- struct SerialPortParameter
  Serial port parameters.
- struct Target
  Robot target.
- struct Value
  Structure used to send/receive values.

Public Types

- enum CallbackType : int { Message , Event }

  Data type for the callback_data structure.
- enum ValueType : int {
  Float32 = 100 , Float64 , Int16 , Int32 ,
  Int64 , UInt16 , UInt32 , UInt64 }

  Supported value types.
- enum AxisParameterType : int {
  Accel , AddaxAxis , AFFGain , AType ,
  AxisAccel , AxisAddress , AxisDecel , AxisDpos ,
  AxisEnable , AxisErrorCount , AxisFSLimit , AxisJogSpeed ,
  AxisMode , AxisRSLimit , AxisSpeed , AxisUnits ,
  AxisStatus , BacklashDist , ChangeDirLast , CloseWin ,
  ClutchRate , CornerMode , CornerState , Creep ,
  DGain , DAC , DACScale , DACOut ,
  DatumIn , Decel , DecelAngle , DemandSpeed ,
  DPos , DriveControl , DriveControlWord , DriveControlWordMode ,
  DriveCurrent , DriveEnable , DriveFE , DriveFELimit ,
  DriveInputs , DriveMode , DriveMonitor , DriveProfile ,
  DriveStatus , DriveTorque , DriveType , DriveVelocity ,
  Encoder , EncoderBits , EncoderControl , EncoderFilter ,
  EncoderId , EncoderStatus , EncoderTurns , EndDirLast ,
  EndMove , EndMoveBuffer , EndMoveSpeed , ErrorMask ,
  FastJog , FastDec , FE , FELimit ,
  FELimitMode , FERange , FELatch , FHoldIn ,
  FHSpeed , ForceDwell , ForceRamp , ForceSpeed ,
  FrameRepDist , FrameScale , FSLimit , FullSpRadius ,
  FwdIn , FwdJog , IGain , Idle ,
  InPos , InPosDist , InPosSpeed , InterpFactor ,
  InvertStep , Jerk , JogSpeed , LinkAxis ,
  Loaded , LookaheadFactor , Mark , MarkB ,
  Merge , MoveCount , MovePA , MovePACont ,
  MovePAIdle , MovePB , MovePBCont , MovePBIdle ,
  MoveLinkModify , MovesBuffered , MPos , MSpeed ,
  MSpeedFilter , MspeedF , MType , NType ,
  OffPos , OpenWin , OutLimit , OVGain ,
  PGain , PosDelay , PPStep , PSEncoder ,
  RaiseAngle , RegDelay , RegPos , RegPosB ,
  RegSpeed , RegSpeedB , Remain , RepDist ,
  RepOption , RevIn , RevJog , RobotDpos ,
  RobotFSLimit , RobotRSLimit , RobotSPMode , RobotStatus ,
  RobotUnits , RSLimit , Servo , SlotNumber ,
  Speed , SRamp , SRef , SRrefOut ,
  StartDirLast , StartMoveSpeed , StopAngle , SyncAxis ,
  SyncControl , SyncDPos , SyncDwell , SyncFlight ,
  SyncPause , SyncTime , SyncTimer , SyncWithdraw ,
  TablePointer , TangDirection , TRef , TRrefOut ,
  Units , VectorBuffered , Verify , VFFGain ,
  VPAccel , VPJerk , VPMode , VPPosition ,
  VPSpeed , WorldAccel , WorldDecel , WorldDPos ,
  WorldFSLimit , WorldJogSpeed , WorldRSLimit , WorldSpeed ,
  WorldUnits , AxisEnableOverride , AxisDisplay , AxisZOutput ,
  DemandEdges , DriveCwMode , DriveIndex , DriveNegTorque ,
  DriveParameter , DrivePosTorque , DriveSetVal , DriveValue ,
  FeedOverride , ForceAccel , ForceJerk , Frame ,
  FwdStart , MposDelay , MposPreComp , PosiSeqDelay ,
  PosiSeqMode , PwmCycle , PwmMark , RegInputs ,

RegistControl , RegistDelay , RegistSpeed , RegistSpeedb ,
RevStart , TcpDpos , TcpFsLimit , TcpRsLimit ,
TcpUnits , RobotFe , RobotFeLatch , RobotFeRange }

List of supported axis parameters. See the TrioBASIC documentation for the description of these parameters.

- enum ProcessParameterType : int { Status , ProcNumber }

  List of supported process parameters. See the TrioBASIC documentation for the description of these parameters.
- enum SystemParameterType : int {
  WDog , PMove , SystemError , CurrentJitter ,
  AverageJitter , InterruptsMissed , FramesMissed , FramesMissing ,
  LimitBuffered , Address , AutoEthercat , Control ,
  CpuExceptions , Display , EpromStatus , ErrorAxis ,
  FpuExceptions , HmiProc , IpAddress , IpGateway ,
  IpMac , IpMemoryConfig , IpNetmask , IpProtocolConfig ,
  IpProtocolCtrl , IpTcpTimeout , IpTcpTxThreshold , IpTcpTxTimeout ,
  LastAxis , ModbusTxDelay , MotionError , MpeMode ,
  Nop , Nin , PcmcatProc , PlcConfig ,
  PlcError , PlcOverflow , PlcRun , RemoteProc ,
  ScheduleType , ScopeCycleCount , ScopeDelay , ScopePos ,
  ScopeTriggerPos , SerialNumber , ServoOffset , ServoPeriod ,
  SystemErrorMask , SystemLoad , SystemLoadMax , TextFileLoaderProc ,
  TSize , UnitError , Version }

  List of supported system parameters. See the TrioBASIC documentation for the description of these parameters.
- enum Event : int {
  None , MotionIdle , MotionLoaded , MotionMark ,
  MotionMarkB , MotionRMark , MotionWarning , MotionError ,
  InputDigitalIn , InputKey }

  Available event types.
- enum TargetPointType { GTA , GTAJ }

  Target type.

Public Member Functions

- delegate void Callback (IntPtr Context, ref CallbackData Data)

  Asynchronous event callback handler. Used to inform the user program of asyncrhonous events that happen in the PC-MCAT.

Static Public Member Functions

- static int Open (Callback Callback, IntPtr CallbackContext)

  Open a conection to the PC-MCAT.
- static bool IsOpen ()

  Check whether the connection to the PC-MCAT is open.
- static void Close ()

  Close the current connection to the PC-MCAT.
- static bool Ex (int Mode)

  Perform EX on the PC-MCAT. This command will automatically close the connection to the API.
- static int EventRegister (EventParameters Parameters)

  Register for an event on the PC-MCAT. This function can be called for multiple times to register for different events. When an event occurs the callback delegate specified in the Open function will be called.
- static int EventUnregister (Event EventType)

  Unregister from an event on the PC-MCAT. This function can be called for multiple times to unregister for different events.
- static int ChannelRead (Int32 channel, byte[ ] buffer, Int64 size, Int64 offset, Int64 length)

Read data from a PC-MCAT communications channel. This call will NOT block if there is not enough data available on the PC-MCAT communications channel.

- static int ChannelWrite (Int32 channel, byte[ ] buffer, Int64 offset, Int64 length)

  Write data to a PC-MCAT communications channel. This call will block if there is not enough space in the PC-MCAT communications channel.
- static int ChannelWriteAvailable (Int32 channel)

  Return the number of bytes available in the PC-MCAT communications channel write buffer. A write of more than this amount will block.
- static int ChannelReadAvailable (Int32 channel)

  Return the number of bytes available in the PC-MCAT communications channel read buffer.
- static int ChannelGetSerialParameters (Int32 channel, out SerialPortParameter parameters)

  Return the number of bytes available in the PC-MCAT communications channel read buffer.
- static int Dir (out string Response, int ResponseSize)

  Gets a directory listing from Motion Coordinator.
- static int EtherCAT (Int64[ ] Parameter, out string Response, int ResponseSize)

  Execute the corresponding ETHERCAT command on the PC-MCAT.
- static int Execute ([MarshalAs(UnmanagedType.LPStr)] String request)

  Execute a TrioBASIC statement on the PC-MCAT.
- static int GetVR (int VR, out double Value)

  Read a single VR value.
- static int GetVR (int FirstVR, ref double[ ] Values)

  Read an array of VR values.
- static int GetVR (int FirstVR, int Length, out double[ ] Values)

  Read VR values.
- static int GetVR (int FirstVR, int Offset, int Length, ref double[ ] Values)

  Read VR values into an area of an array.
- static int SetVR (int VR, double Value)

  Write VR value.
- static int SetVR (int FirstVR, double[ ] Values)

  Write an array of VR values.
- static int SetVR (int FirstVR, int Length, double[ ] Values)

  Write VR value.
- static int SetVR (int FirstVR, int Offset, int Length, double[ ] Values)

  Write VR value.
- static int GetTABLE (int TABLE, out double Value)

  Read a single TABLE value.
- static int GetTABLE (int FirstTABLE, ref double[ ] Values)

  Read an array of TABLE values.
- static int GetTABLE (int FirstTABLE, int Length, out double[ ] Values)

  Read TABLE values.
- static int GetTABLE (int FirstTABLE, int Offset, int Length, ref double[ ] Values)

  Read TABLE values into an area of an array.
- static int SetTABLE (int TABLE, double Value)

  Write TABLE value.
- static int SetTABLE (int FirstTABLE, double[ ] Values)

  Write an array of TABLE values.
- static int SetTABLE (int FirstTABLE, int Length, double[ ] Values)

  Write TABLE value.
- static int SetTABLE (int FirstTABLE, int Offset, int Length, double[ ] Values)

  Write TABLE value.
- static int GetDigitalOutput (int OutputNumber, out int Value)

Get single digital output.

- static int SetDigitalOutput (int OutputNumber, int Value)

  Set single digital output.
- static int SetDigitalInvertInput (int OutputNumber, int Value)

  Set single digital input inversion.
- static int GetDigitalInput (int InputNumber, out int Value)

  Get single digital input.
- static int GetDigitalInvertInput (int InputNumber, out Int32 Value)

  Get single digital input inversion.
- static int GetDigitalOutputRange (int FirstOutputNumber, int LastOutputNumber, out int Value)

  Get digital output bank. First and Last values are inclusive. Can return at most 32 bits.
- static int SetDigitalOutputRange (int FirstOutputNumber, int LastOutputNumber, int Value)

  Set digital output bank. First and last values are inclusive.
- static int GetDigitalInputRange (int FirstInputNumber, int LastInputNumber, out int Value)

  Get digital input bank. First and last values are inclusive.
- static int GetAnalogueOutput (int OutputNumber, out int Value)

  Get single analogue output.
- static int SetAnalogueOutput (int OutputNumber, int Value)

  Set single analogue output.
- static int GetAnalogueInput (int InputNumber, out int Value)

  Get single analogue input.
- static int GetAxisParameter (AxisParameterType AxisParameter, out Value Value)

  Read axis parameter value from current base axis.
- static int GetAxisParameter (AxisParameterType AxisParameter, int Axis, out Value Value)

  Read axis parameter value.
- static int SetAxisParameter (AxisParameterType AxisParameter, Value Value)

  Write current base axis parameter value.
- static int SetAxisParameter (AxisParameterType AxisParameter, int Axis, Value Value)

  Write axis parameter value.
- static int GetProcessParameter (ProcessParameterType ProcessParameter, int Process, out Value Value)

  Read process parameter values.
- static int SetProcessParameter (ProcessParameterType ProcessParameter, int Process, Value Value)

  Write process parameter value.
- static int GetSystemParameter (SystemParameterType SystemParameter, out Value Value)

  Read system parameter values.
- static int SetSystemParameter (SystemParameterType SystemParameter, Value Value)

  Write axis parameter value.
- static int RunProgram (string ProgramName)

  Run a TrioBASIC program on a given process.
- static int RunProgram (string ProgramName, int ProcessNumber)

  Run a TrioBASIC program on a given process.
- static int StopProgram (string ProgramName)

  Stop all running instances of a TrioBASIC program.
- static int StopProgram (string ProgramName, int ProcessNumber)

  Stop a TrioBASIC program on a given process.
- static int IsFile (string FileName, out bool FileExists)

  Check whether a file exists on the controller.
- static int SetBaseVector (int[ ] BaseVector)

  Set the BASE vector for the PC-MCAT API.
- static int SetBaseVector (int Offset, int Length, int[ ] BaseVector)

  Set the BASE vector for the PC-MCAT API.

- static int PSwitch (int PSwitchID, int Enable, int Axis, int Output, int State, double SetPosition, double Reset↩ Position)
- static int PSwitchOff (int PSwitchID, int ResetOutput)

  Performs the corresponding PSWITCH(Switch, OFF, Hold) command on the Motion Coordinator Disable a PSWITCH.
- static int Addax (int BaseAxis, int AddAxis)

  *Performs the corresponding ADDAX(...) command of Motion Controller Superimpose 2 or more movements to build up a more complex movement profile The **Addax** command takes the demand position changes from the specified axis and adds them to any movements running on the base axis. After the **Addax** command has been issued the link between the two axes remains until broken and any further moves on the specified axis will be added to the base axis. The specified axis can be any axis and does not have to physically exist in the system The **Addax** command therefore allows an axis to perform the moves specified on TWO axes added together.*
- static int Regist (int BaseAxis, int Mode, int Channel, int Source, int Edge, int Window, int RepeatCount, int TableStart)

  Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.
- static int Regist (int BaseAxis, int Mode, int Channel, int Source, int Edge, int Window)

  Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.
- static int Regist (int BaseAxis, int Mode)

  Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.
- static int UnitClear (int Slot)

  Clear all the bits in the UNIT_ERROR system parameter.
- static int Datum (int BaseAxis, int Mode)

  Peforms the corresponding DATUM(...)AXIS(...)command on the Motion Controller Perform a homing routine on the given axis.
- static int Defpos (int BaseAxis, double[ ] Position)

  Set the current DPOS on the Motion Coordinator.
- static int Defpos (int BaseAxis, int Offset, int Length, double[ ] Position)

  Set the current DPOS on the Motion Coordinator.
- static int Forward (int BaseAxis)

  Performs the corresponding FORWARD(...) AXIS(...) command on the Motion Controller Continuous forward movement on the given axis.
- static int Reverse (int BaseAxis)

  Performs the corresponding REVERSE(...) AXIS(...) command on the Motion Coordinator Continuous reverse movement on the given axis.
- static int Cam (int BaseAxis, int TableStart, int TableStop, int TableMultiplier, double LinkDistance)

  Performs the corresponding CAM(...)AXIS(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect time.
- static int Cambox (int BaseAxis, int TableStart, int TableStop, int TableMultiplier, double LinkDistance, int LinkAxis)

  Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.
- static int Cambox (int BaseAxis, int TableStart, int TableStop, int TableMultiplier, double LinkDistance, int LinkAxis, int LinkOptions)

  Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.
- static int Cambox (int BaseAxis, int TableStart, int TableStop, int TableMultiplier, double LinkDistance, int LinkAxis, int LinkOptions, double LinkPosition, double LinkOffsetDistance)

  Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.
- static int Connect (int BaseAxis, double Ratio, int LinkAxis)

  Performs the corresponding CONNECT(...) AXIS(...) command on the Motion Coordinator Links the demand position of the base axis to the measured movements of the driving axes to produce an electronic gearbox.
- static int FlexLink (int BaseAxis, double BaseDist, double ExciteDist, double LinkDist, double BaseIn, double BaseOut, double ExciteAcc, double ExciteDec, int LinkAxis, int LinkOptions, double LinkPosition)

*See FLEXLINK documentation in Motion Perfect helpfile The **FlexLink** command is used to generate movement of an axis according to a defined profile. The motion is linked to the measured motion of another axis. The profile is made up of 2parts,the base move and the excitation move both of which are specifiedin the parameters. The base move is a constant speed movement. The excitation movement uses sinusoidal or alternative profile and is applied on top of the base movement.*

- static int Sync (int BaseAxis, int Control, int SyncTime, double SyncAxisPosition, int SyncAxis, double[ ] SyncPosition)

  Syncronize movement of one axis to another.
- static int SyncClear (int BaseAxis, int SyncTime)

  Stop the syncronized movement.
- static int MoveAbsolute (int BaseAxis, double[ ] Position)

  Peforms the corresponding MOVEABS(...)AXIS(...) command on the Motion Coordinator Move axis to position(s) referenced with respect to the zero (home) position. Dispatch an absolute linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveAbsolute (int BaseAxis, int Offset, int Length, double[ ] Position)

  Peforms the corresponding MOVEABS(...)AXIS(...) command on the Motion Coordinator Move axis to position(s) referenced with respect to the zero (home) position. Dispatch an absolute linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveCircular (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction)

  Performs the corresponding MOVECIRC(...)AXIS(...) command on the Motion Coordinator Dispatch a circular move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveHelical (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction, double DistanceAxis3, int Mode)

  Performs the corresponding MHELICAL(...)AXIS(...) on command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveHelical (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction, double DistanceAxis3)

  Performs the corresponding MHELICAL(...)AXIS(...) on command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveRelative (int BaseAxis, double[ ] Distance)

  Performs the corresponding MOVE(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveRelative (int BaseAxis, int Offset, int Length, double[ ] Distance)

  Performs the corresponding MOVE(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveSpherical (int BaseAxis, double Pos1X, double Pos1Y, double Pos1Z, double Pos2X, double Pos2Y, double Pos2Z, int Mode, int Direction, int RotationU, int RotationV, int RotationW, int RotationP)

  Dispatch a spherical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveSpherical (int BaseAxis, double Pos1X, double Pos1Y, double Pos1Z, double Pos2X, double Pos2Y, double Pos2Z, int Mode, int Direction)

Dispatch a spherical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

- static int MoveRelSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius, int TableOutput, double TransitionAngle)

  Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.
- static int MoveRelSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius, int TableOutput)

  Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.
- static int MoveRelSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius)

  Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.
- static int MoveRelSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options)

  Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.
- static int MoveAbsSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius, int TableOutput, double TransitionAngle)
- static int MoveAbsSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius, int TableOutput)
- static int MoveAbsSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options, double Radius)
- static int MoveAbsSeq (int BaseAxis, int TableInput, int AxisCount, int MoveCount, int Options)
- static int MoveAbsolute_SP (int BaseAxis, double[ ] Position)

  Peforms the corresponding MOVEABSSP(...)AXIS(...) command on the Motion Coordinator Refer to MOVEABSSP documentation. Dispatch an absolute linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveAbsolute_SP (int BaseAxis, int Offset, int Length, double[ ] Position)

  Peforms the corresponding MOVEABSSP(...)AXIS(...) command on the Motion Coordinator Refer to MOVEABSSP documentation. Dispatch an absolute linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveCircular_SP (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction)

  Performs the Corresponding MOVECIRCSP(...) AXIS(...) command on the Motion Controller. Dispatch a circular move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveHelical_SP (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction, double DistanceAxis3, int Mode)

  Performs the corresponding MHELICALSP(...) AXIS(...) command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveHelical_SP (int BaseAxis, double EndAxis1, double EndAxis2, double CentreAxis1, double CentreAxis2, int Direction, double DistanceAxis3)

  Performs the corresponding MHELICALSP(...) AXIS(...) command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveRelative_SP (int BaseAxis, double[ ] Distance)

  Performs the corresponding MOVESP(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

- static int MoveRelative_SP (int BaseAxis, int Offset, int Length, double[ ] Distance)

  Performs the corresponding MOVESP(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveSpherical_SP (int BaseAxis, double Pos1X, double Pos1Y, double Pos1Z, double Pos2X, double Pos2Y, double Pos2Z, int Mode, int Direction, int RotationU, int RotationV, int RotationW, int RotationP)

  Performs the corresponding MSPHERICAL(...) command on the Motion Coordinator Dispatch a spherical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int MoveSpherical_SP (int BaseAxis, double Pos1X, double Pos1Y, double Pos1Z, double Pos2X, double Pos2Y, double Pos2Z, int Mode, int Direction)

  Dispatch a spherical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.
- static int Cancel (int BaseAxis, int Mode)

  Cancel a currently running movement on an axis.
- static int RapidStop (int Mode)

  Performs the corresponding RAPIDSTOP(...) command on the Motion Controller Cancel a currently running movement on all axes.
- static int SetGTA (int GTAId, string GTAName, ref Target GTAValue)

  Set a member of the Global Target Array.
- static int SetObjectFrame (int ObjectFrameId, string ObjectFrameName, ref ObjectFrame ObjectFrameValue)

  Set a member of the Object Frame Array.
- static int SyncToGta (int BaseAxis, int Control, int Time, double SyncPos, double SyncAxis, int ObjectFrame, int GTA)

  Synchronise the robot TCP GTA with a moving object.
- static int SyncToGta (int BaseAxis, int Control, int Time, double SyncPos, double SyncAxis, string ObjectFrame, string GTA)

  Synchronise the robot TCP GTA with a moving object.
- static int SyncToObjectFrame (int BaseAxis, int Control, int Time, int ObjectFrame, int GTA)

  Synchronise the robot TCP GTA with a moving object.
- static int SyncToObjectFrame (int BaseAxis, int Control, int Time, string ObjectFrame, string GTA)

  Synchronise the robot TCP GTA with a moving object.
- static int TcpCalibrate ()
- static int UserFrame (int FrameId, double XOffset, double YOffset, double ZOffset, double XRotation, double YRotation, double ZRotation)

  Set the user frame offset from the world coordinate origin.
- static int UserFrame (int FrameId, double XOffset, double YOffset, double ZOffset)

  Set the user frame offset from the world coordinate origin.
- static int UserFrame (int FrameId)

  Set the user frame offset from the world coordinate origin.
- static int ToolOffset (int ToolID, double XOffset, double YOffset, double ZOffset, double XRotation, double YRotation, double ZRotation)

  Set the tool offset from the world coordinate origin.
- static int ToolOffset (int ToolID, double XOffset, double YOffset, double ZOffset)

  Set the tool offset from the world coordinate origin.
- static int ToolSelect (int BaseAxis, int ToolID)

  Select the tool offset from the world coordinate origin.

- static int Sync (int BaseAxis, int Control, int SyncTime, double SyncAxisPosition, int SyncAxis, double[ ] SyncPosition, int UserFrameID, double[ ] SyncRotation)

  Syncronize movement to a user frame.
- static int Sync (int BaseAxis, int Control, int SyncTime, double SyncAxisPosition, int SyncAxis, double[ ] SyncPosition, int UserFrameID)

  Syncronize movement to a user frame.
- static int FrameGroupSet (int FrameGroup, int TableOffset, int[ ] Axis)

  Define the group of axes and the table offset which are used in a FRAME or USER_FRAME transformation.
- static int FrameGroupClear (int FrameGroup)

  Clear the frame group.

5.4.1 Detailed Description

Implements the PC-MCAT shared memory API.

5.4.2 Member Enumeration Documentation

5.4.2.1 AxisParameterType

```
enum AxisParameterType :  int
```

List of supported axis parameters. See the TrioBASIC documentation for the description of these parameters.

Enumerator

Accel	
AddaxAxis	
AFFGain	
AType	
AxisAccel	
AxisAddress	
AxisDecel	
AxisDpos	
AxisEnable	
AxisErrorCount	
AxisFSLimit	
AxisJogSpeed	
AxisMode	
AxisRSLimit	
AxisSpeed	
AxisUnits	
AxisStatus	
BacklashDist	
ChangeDirLast	
CloseWin	
ClutchRate	
CornerMode	
CornerState	
Creep	
DGain	

Enumerator

DAC	
DACScale	
DACOut	
DatumIn	
Decel	
DecelAngle	
DemandSpeed	
DPos	
DriveControl	
DriveControlWord	
DriveControlWordMode	
DriveCurrent	
DriveEnable	
DriveFE	
DriveFELimit	
DriveInputs	
DriveMode	
DriveMonitor	
DriveProfile	
DriveStatus	
DriveTorque	
DriveType	
DriveVelocity	
Encoder	
EncoderBits	
EncoderControl	
EncoderFilter	
EncoderId	
EncoderStatus	
EncoderTurns	
EndDirLast	
EndMove	
EndMoveBuffer	
EndMoveSpeed	
ErrorMask	
FastJog	
FastDec	
FE	
FELimit	
FELimitMode	
FERange	
FELatch	
FHoldIn	
FHSpeed	
ForceDwell	
ForceRamp	
ForceSpeed	
FrameRepDist	
FrameScale	
FSLimit	

Enumerator

FullSpRadius	
FwdIn	
FwdJog	
IGain	
Idle	
InPos	
InPosDist	
InPosSpeed	
InterpFactor	
InvertStep	
Jerk	
JogSpeed	
LinkAxis	
Loaded	
LookaheadFactor	
Mark	
MarkB	
Merge	
MoveCount	
MovePA	
MovePACont	
MovePAIdle	
MovePB	
MovePBCont	
MovePBIdle	
MoveLinkModify	
MovesBuffered	
MPos	
MSpeed	
MSpeedFilter	
MspeedF	
MType	
NType	
OffPos	
OpenWin	
OutLimit	
OVGain	
PGain	
PosDelay	
PPStep	
PSEncoder	
RaiseAngle	
RegDelay	
RegPos	
RegPosB	
RegSpeed	
RegSpeedB	
Remain	
RepDist	

Enumerator

RepOption	
RevIn	
RevJog	
RobotDpos	
RobotFSLimit	
RobotRSLimit	
RobotSPMode	
RobotStatus	
RobotUnits	
RSLimit	
Servo	
SlotNumber	
Speed	
SRamp	
SRef	
SRrefOut	
StartDirLast	
StartMoveSpeed	
StopAngle	
SyncAxis	
SyncControl	
SyncDPos	
SyncDwell	
SyncFlight	
SyncPause	
SyncTime	
SyncTimer	
SyncWithdraw	
TablePointer	
TangDirection	
TRef	
TRrefOut	
Units	
VectorBuffered	
Verify	
VFFGain	
VPAccel	
VPJerk	
VPMode	
VPPosition	
VPSpeed	
WorldAccel	
WorldDecel	
WorldDPos	
WorldFSLimit	
WorldJogSpeed	
WorldRSLimit	
WorldSpeed	
WorldUnits	

Enumerator

AxisEnableOverride	
AxisDisplay	
AxisZOutput	
DemandEdges	
DriveCwMode	
DriveIndex	
DriveNegTorque	
DriveParameter	
DrivePosTorque	
DriveSetVal	
DriveValue	
FeedOverride	
ForceAccel	
ForceJerk	
Frame	
FwdStart	
MposDelay	
MposPreComp	
PosiSeqDelay	
PosiSeqMode	
PwmCycle	
PwmMark	
RegInputs	
RegistControl	
RegistDelay	
RegistSpeed	
RegistSpeedb	
RevStart	
TcpDpos	
TcpFsLimit	
TcpRsLimit	
TcpUnits	
RobotFe	
RobotFeLatch	
RobotFeRange	

5.4.2.2 CallbackType

```
enum CallbackType :  int
```

Data type for the callback_data structure.

Enumerator

Message	A text message.
Event	An event.

5.4.2.3 Event

```
enum Event : int
```

Available event types.

Enumerator

None	Invalid event.
MotionIdle	IDLE event.
MotionLoaded	LOADED event.
MotionMark	Registration mark event.
MotionMarkB	Registration mark B event.
MotionRMark	Registration R mark event.
MotionWarning	Registration warning event.
MotionError	Registration error event.
InputDigitalIn	Digital input event.
InputKey	Channel input event.

5.4.2.4 ProcessParameterType

```
enum ProcessParameterType : int
```

List of supported process parameters. See the TrioBASIC documentation for the description of these parameters.

Enumerator

Status	
ProcNumber	

5.4.2.5 SystemParameterType

```
enum SystemParameterType : int
```

List of supported system parameters. See the TrioBASIC documentation for the description of these parameters.

Enumerator

WDog	
PMove	
SystemError	
CurrentJitter	
AverageJitter	
InterruptsMissed	
FramesMissed	
FramesMissing	
LimitBuffered	
Address	

Enumerator

AutoEthercat	
Control	
CpuExceptions	
Display	
EpromStatus	
ErrorAxis	
FpuExceptions	
HmiProc	
IpAddress	
IpGateway	
IpMac	
IpMemoryConfig	
IpNetmask	
IpProtocolConfig	
IpProtocolCtrl	
IpTcpTimeout	
IpTcpTxThreshold	
IpTcpTxTimeout	
LastAxis	
ModbusTxDelay	
MotionError	
MpeMode	
Nop	
Nin	
PcmcatProc	
PlcConfig	
PlcError	
PlcOverflow	
PlcRun	
RemoteProc	
ScheduleType	
ScopeCycleCount	
ScopeDelay	
ScopePos	
ScopeTriggerPos	
SerialNumber	
ServoOffset	
ServoPeriod	
SystemErrorMask	
SystemLoad	
SystemLoadMax	
TextFileLoaderProc	
TSize	
UnitError	
Version	

5.4.2.6 TargetPointType

```
enum TargetPointType
```

Target type.

Enumerator

GTA	Point.
GTAJ	Joint.

5.4.2.7 ValueType

```
enum ValueType :  int
```

Supported value types.

Enumerator

Float32	32 bit floating point type
Float64	64 bit floating point type
Int16	16 bit signed integer type
Int32	32 bit signed integer type
Int64	64 bit signed integer type
UInt16	16 bit unsigned integer type
UInt32	32 bit unsigned integer type
UInt64	64 bit unsigned integer type

5.4.3 Member Function Documentation

5.4.3.1 Addax()

```
static int Addax (
            int BaseAxis,
            int AddAxis )  [inline], [static]
```

Performs the corresponding ADDAX(...) command of Motion Controller Superimpose 2 or more movements to build up a more complex movement profile The **Addax** command takes the demand position changes from the specified axis and adds them to any movements running on the base axis. After the **Addax** command has been issued the link between the two axes remains until broken and any further moves on the specified axis will be added to the base axis. The specified axis can be any axis and does not have to physically exist in the system The **Addax** command therefore allows an axis to perform the moves specified on TWO axes added together.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the **Addax** connection is cleared, the default base axis is used.
AddAxis	The axis to superimpose. -1 breaks an existing superposition.

Returns

On success returns 0, otherwise returns -1

5.4.3.2 Callback()

```
delegate void Callback (
            IntPtr Context,
            ref CallbackData Data )
```

Asynchronous event callback handler. Used to inform the user program of asyncrhonous events that happen in the PC-MCAT.

Parameters

Context	callback_context parameter supplied in the call to the open function.
Data	Data associated to the asynchronous event.

Open

5.4.3.3 Cam()

```
static int Cam (
            int BaseAxis,
            int TableStart,
            int TableStop,
            int TableMultiplier,
            double LinkDistance )  [inline], [static]
```

Performs the corresponding CAM(...)AXIS(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect time.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableStart	Start of the CAM profile in the TABLE data.
TableStop	End of the CAM profile in the TABLE data.
TableMultiplier	The Scaling Factor to applied to the CAM pattern.
LinkDistance	The time for this CAM profile is calculated as LinkDistance/SPEED.The Speed can be modified during the move, and will affect directly the speed with which the cam is performed.

Returns

On success returns 0, otherwise returns -1

5.4.3.4 Cambox() [1/3]

```
static int Cambox (
            int BaseAxis,
```

```
            int TableStart,
            int TableStop,
            int TableMultiplier,
            double LinkDistance,
            int LinkAxis )  [inline], [static]
```

Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableStart	Start of the CAM profile in the TABLE data.
TableStop	End of the CAM profile in the TABLE data.
TableMultiplier	Factor to be applied to all the data points in the CAM profile.
LinkDistance	Distance that the link axis must travel to complete the profile.
LinkAxis	Input axis that drives this move.

Returns

On success returns 0, otherwise returns -1

5.4.3.5 Cambox() [2/3]

```
static int Cambox (
            int BaseAxis,
            int TableStart,
            int TableStop,
            int TableMultiplier,
            double LinkDistance,
            int LinkAxis,
            int LinkOptions )  [inline], [static]
```

Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableStart	Start of the CAM profile in the TABLE data.
TableStop	End of the CAM profile in the TABLE data.
TableMultiplier	Factor to be applied to all the data points in the CAM profile.
LinkDistance	Distance that the link axis must travel to complete the profile.
LinkAxis	Input axis that drives this move.
LinkOptions	A bitmap of zero or more options for the CAMBOX command. See the CAMBOX command documentation for more options. 1 - link commences exactly when registration event occurs on link axis. 2 -link commences at an absolute position on link axis (see LinkPos) 4 -CAMBOX repeats automatically and bi-directionally when this bit is set. 8 - Pattern Mode 32 - Link is only active during positive moves

Returns

On success returns 0, otherwise returns -1

5.4.3.6 Cambox() [3/3]

```
static int Cambox (
            int BaseAxis,
            int TableStart,
            int TableStop,
            int TableMultiplier,
            double LinkDistance,
            int LinkAxis,
            int LinkOptions,
            double LinkPosition,
            double LinkOffsetDistance )  [inline], [static]
```

Performs the corresponding CAMBOX(...) command on the Motion Coordinator Process a cam profile stored in TABLE data with respect the position of a different axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableStart	Start of the CAM profile in the TABLE data.
TableStop	End of the CAM profile in the TABLE data.
TableMultiplier	Factor to be applied to all the data points in the CAM profile.
LinkDistance	Distance that the link axis must travel to complete the profile.
LinkAxis	Input axis that drives this move.
LinkOptions	A bitmap of zero or more options for the CAMBOX command. See the CAMBOX command documentation for more details.
LinkPosition	See CAMBOX command documentation for more options. 1 - link commences exactly when registration event occurs on link axis. 2 -link commences at an absolute position on link axis (see LinkPos). 4 -CAMBOX repeats automatically and bi-directionally when this bit is set. 8 - Pattern Mode. 32 - Link is only active during positive moves.
LinkOffsetDistance	See CAMBOX command documentation for more details.

Returns

On success returns 0, otherwise returns -1

5.4.3.7 Cancel()

```
static int Cancel (
            int BaseAxis,
            int Mode )  [inline], [static]
```

Cancel a currently running movement on an axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Mode	Type of cancel to be performed.0 cancels the current move on the base axis, 1 cancels the buffered moves on the base axis. See the CANCEL command documentation.

Returns

On success returns 0, otherwise returns -1

5.4.3.8 ChannelGetSerialParameters()

```
static int ChannelGetSerialParameters (
            Int32 channel,
            out SerialPortParameter parameters )  [inline], [static]
```

Return the number of bytes available in the PC-MCAT communications channel read buffer.

Parameters

channel	Channel to check. The valid communications channels are 1 and 2.
parameters	Parameters read from controller.

Returns

On success returns the number of bytes available, otherwise returns -1

5.4.3.9 ChannelRead()

```
static int ChannelRead (
            Int32 channel,
            byte[] buffer,
            Int64 size,
            Int64 offset,
            Int64 length )  [inline], [static]
```

Read data from a PC-MCAT communications channel. This call will NOT block if there is not enough data available on the PC-MCAT communications channel.

Parameters

channel	Channel to be written. The valid communications channels are 1 and 2.
buffer	Buffer in which to store the bytes read.
size	Size of the buffer.
offset	Offset into the buffer to the first byte to be stored.
length	Number of bytes to read.

Returns

On success returns number of bytes read, otherwise returns -1

5.4.3.10 ChannelReadAvailable()

```
static int ChannelReadAvailable (
            Int32 channel )  [inline], [static]
```

Return the number of bytes available in the PC-MCAT communications channel read buffer.

Parameters

channel	Channel to check. The valid communications channels are 1 and 2.

Returns

On success returns the number of bytes available, otherwise returns -1

5.4.3.11 ChannelWrite()

```
static int ChannelWrite (
            Int32 channel,
            byte[] buffer,
            Int64 offset,
            Int64 length )  [inline], [static]
```

Write data to a PC-MCAT communications channel. This call will block if there is not enough space in the PC-MCAT communications channel.

Parameters

channel	Channel to be written. The valid communications channels are 1 and 2.
buffer	Buffer to be written.
offset	Offset into the buffer to the first byte to be written.
length	Number of bytes to be written.

Returns

On success returns number of bytes written, otherwise returns -1

5.4.3.12 ChannelWriteAvailable()

```
static int ChannelWriteAvailable (
            Int32 channel )  [inline], [static]
```

Return the number of bytes available in the PC-MCAT communications channel write buffer. A write of more than this amount will block.

Parameters

channel	Channel to check. The valid communications channels are 1 and 2.

Returns

On success returns the number of bytes available, otherwise returns -1

5.4.3.13 Close()

```
static void Close ( )  [inline], [static]
```

Close the current connection to the PC-MCAT.

5.4.3.14 Connect()

```
static int Connect (
            int BaseAxis,
            double Ratio,
            int LinkAxis )  [inline], [static]
```

Performs the corresponding CONNECT(...) AXIS(...) command on the Motion Coordinator Links the demand position of the base axis to the measured movements of the driving axes to produce an electronic gearbox.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Ratio	The number of edges to be moved on the output axis for every edge on the input axis.
LinkAxis	Input axis drives this move.

Returns

On success returns 0, otherwise returns -1

5.4.3.15 Datum()

```
static int Datum (
            int BaseAxis,
            int Mode )  [inline], [static]
```

Peforms the corresponding DATUM(...)AXIS(...)command on the Motion Controller Perform a homing routine on the given axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Mode	Type of homing routine to be performed. See the DATUM command documentation. 0-The current measured position is set as demand position (this is especially useful on stepper axes with position verification). TrioPC_DATUM(0) will also reset a following error condition in the AXISSTATUS register for all axes 1-The axis moves at creep speed forward till the Z marker is encountered. The Demand position is then reset to zero and the Measured position corrected so as to maintain the following error 2-The axis moves at creep speed in reverse till the Z marker is encountered. The Demand position is then reset to zero and the Measured position corrected so as to maintain the following error 3-The axis moves at the programmed speed forward until the datum switch is reached. The axis then moves backwards at creep speed until the datum switch is reset. The Demand position is then reset to zero and the Measured position corrected so as to maintain the following error 4-The axis moves at the programmed speed reverse until the datum switch is reached. The axis then moves at creep speed forward until the datum switch is reset. The Demand position is then reset to zero and the Measured position corrected so as to maintain the following error 5-The axis moves at programmed speed forward until the datum switch is reached. The axis then moves at creep speed until the datum switch is reset. The axis is then reset as in mode 2 6-The axis moves at programmed speed reverse until the datum switch is reached. The axis then moves at creep speed forward until the datum switch is reset. The axis is then reset as in mode 1

Returns

On success returns 0, otherwise returns -1

5.4.3.16 Defpos() [1/2]

```
static int Defpos (
            int BaseAxis,
            double[] Position )  [inline], [static]
```

Set the current DPOS on the Motion Coordinator.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Position	Absolution position to set for each axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.17 Defpos() [2/2]

```
static int Defpos (
            int BaseAxis,
            int Offset,
            int Length,
            double[] Position )  [inline], [static]
```

Set the current DPOS on the Motion Coordinator.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Offset	Offset into the Position array to the first value to be written.
Length	Number of values to write.
Position	Absolution position to set for each axis.

Returns

On success returns 0, otherwise returns -1.

5.4.3.18 Dir()

```
static int Dir (
            out string Response,
            int ResponseSize )  [inline], [static]
```

Gets a directory listing from Motion Coordinator.

Parameters

Response	Buffer to receive the command output
ResponseSize	Maximum size of the response buffer.

Returns

On success returns 0, otherwise returns -1

5.4.3.19 EtherCAT()

```
static int EtherCAT (
            Int64[] Parameter,
            out string Response,
            int ResponseSize )  [inline], [static]
```

Execute the corresponding ETHERCAT command on the PC-MCAT.

Parameters

Parameter	Array of parameters. See the ETHERCAT command documentation.
Response	Buffer to receive the command output
ResponseSize	Maximum size of the response buffer.

Returns

On success returns 0, otherwise returns -1

5.4.3.20 EventRegister()

```
static int EventRegister (
            EventParameters Parameters )  [inline], [static]
```

Register for an event on the PC-MCAT. This function can be called for multiple times to register for different events. When an event occurs the callback delegate specified in the Open function will be called.

Parameters

Parameters	Definition of the event to be registered.

Returns

On success returns 0, otherwise returns -1

5.4.3.21 EventUnregister()

```
static int EventUnregister (
            Event EventType )  [inline], [static]
```

Unregister from an event on the PC-MCAT. This function can be called for multiple times to unregister for different events.

Parameters

EventType	Event to be unregistered.

Returns

On success returns 0, otherwise returns -1

5.4.3.22 Ex()

```
static bool Ex (
            int Mode )  [inline], [static]
```

Perform EX on the PC-MCAT. This command will automatically close the connection to the API.

Parameters

Mode	EX mode to be performed 0 or none -Software restarts only the real-time firmware, leaving the Windows service running which maintains communications. 1-Software stops the real-time firmware and the Windows service. 2-Software stops the real-time firmware, stops all the Windows service, and stops Windows. 3-Software stops the real-time firmware, stops all the Windows service, and restarts Windows. 4-Software stops the real-time firmware and tells Windows service to restart it. This is as closest approximation to the physical controller EX(1).

5.4.3.23 Execute()

```
static int Execute (
            [MarshalAs(UnmanagedType.LPStr)] String request )  [inline], [static]
```

Execute a TrioBASIC statement on the PC-MCAT.

Parameters

request	Valid TrioBASIC command to be executed

Returns

On success returns 0, otherwise returns -1

5.4.3.24 FlexLink()

```
static int FlexLink (
            int BaseAxis,
            double BaseDist,
```

```
            double ExciteDist,
            double LinkDist,
            double BaseIn,
            double BaseOut,
            double ExciteAcc,
            double ExciteDec,
            int LinkAxis,
            int LinkOptions,
            double LinkPosition )  [inline], [static]
```

See FLEXLINK documentation in Motion Perfect helpfile The **FlexLink** command is used to generate movement of an axis according to a defined profile. The motion is linked to the measured motion of another axis. The profile is made up of 2parts,the base move and the excitation move both of which are specifiedin the parameters. The base move is a constant speed movement. The excitation movement uses sinusoidal or alternative profile and is applied on top of the base movement.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
BaseDist	The distance the axis should travel at a constant speed
ExciteDist	The distance of the profiled move on the output axis.
LinkDist	The distance that the link axis will move when the FlexLink prfile executes.
BaseIn	Percentage of the move that completes before the excitation move starts.
BaseOut	Percentage of the move that remains after the excitation move stops.
ExciteAcc	Percentage of the move during which the excitation move accelerates.
ExciteDec	Percentage of the move during which the excitation move decelerates.
LinkAxis	The Link axis to link to.
LinkOptions	Bit value options to customize how your FLEXLINK operates. See the FLEXLINK command in Motion Perfect helpfile for more details.
LinkPosition	See the FLEXLINK command documentation for more details.

Returns

On success returns 0, otherwise returns -1

5.4.3.25 Forward()

```
static int Forward (
            int BaseAxis )  [inline], [static]
```

Performs the corresponding FORWARD(...) AXIS(...) command on the Motion Controller Continuous forward movement on the given axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.

Returns

On success returns 0, otherwise returns -1

5.4.3.26 FrameGroupClear()

```
static int FrameGroupClear (
            int FrameGroup )  [inline], [static]
```

Clear the frame group.

Parameters

FrameGroup	Group number in the range 0-7.

Returns

On success returns 0, otherwise returns -1

5.4.3.27 FrameGroupSet()

```
static int FrameGroupSet (
            int FrameGroup,
            int TableOffset,
            int[] Axis )  [inline], [static]
```

Define the group of axes and the table offset which are used in a FRAME or USER_FRAME transformation.

Parameters

FrameGroup	Group number in the range 0-7.
TableOffset	Start position of the FRAME configuration in the TABLE data.
Axis	Array of integers that specify the axes in the frame group.

Returns

On success returns 0, otherwise returns -1

5.4.3.28 GetAnalogueInput()

```
static int GetAnalogueInput (
            int InputNumber,
            out int Value )  [inline], [static]
```

Get single analogue input.

Parameters

InputNumber	Number of the input to be read
Value	Integer value to receive the input status

Returns

On success returns 0, otherwise returns -1

5.4.3.29 GetAnalogueOutput()

```
static int GetAnalogueOutput (
            int OutputNumber,
            out int Value )  [inline], [static]
```

Get single analogue output.

Parameters

OutputNumber	Number of the output to be read
Value	Integer value to receive the output status

Returns

On success returns 0, otherwise returns -1

5.4.3.30 GetAxisParameter() [1/2]

```
static int GetAxisParameter (
            AxisParameterType AxisParameter,
            int Axis,
            out Value Value )  [inline], [static]
```

Read axis parameter value.

Parameters

AxisParameter	Axis parameter to be read
Axis	Axis to be read. If this parameter is -1 then reads the current base axis.
Value	Buffer to receive the value

Returns

On success returns 0, otherwise returns -1

5.4.3.31 GetAxisParameter() [2/2]

```
static int GetAxisParameter (
            AxisParameterType AxisParameter,
            out Value Value )  [inline], [static]
```

Read axis parameter value from current base axis.

Parameters

AxisParameter	Axis parameter to be read
Value	Buffer to receive the value

Returns

On success returns 0, otherwise returns -1

5.4.3.32 GetDigitalInput()

```
static int GetDigitalInput (
            int InputNumber,
            out int Value )  [inline], [static]
```

Get single digital input.

Parameters

InputNumber	Number of the input to be read
Value	Integer value to receive the output status

Returns

On success returns 0, otherwise returns -1

5.4.3.33 GetDigitalInputRange()

```
static int GetDigitalInputRange (
            int FirstInputNumber,
            int LastInputNumber,
            out int Value )  [inline], [static]
```

Get digital input bank. First and last values are inclusive.

Parameters

FirstInputNumber	Number of the first input to be read
LastInputNumber	Number of the last input to be read
Value	Integer value to receive the input status. This is a bitmap of the individial input statii.

Returns

On success returns 0, otherwise returns -1

5.4.3.34 GetDigitalInvertInput()

```
static int GetDigitalInvertInput (
```

```
            int InputNumber,
            out Int32 Value )  [inline], [static]
```

Get single digital input inversion.

Parameters

InputNumber	Number of the input inversion to be read
Value	Integer value to receive the input inversion status

Returns

On success returns 0, otherwise returns -1

5.4.3.35 GetDigitalOutput()

```
static int GetDigitalOutput (
            int OutputNumber,
            out int Value )  [inline], [static]
```

Get single digital output.

Parameters

OutputNumber	Number of the output to be read
Value	Integer value to receive the output status

Returns

On success returns 0, otherwise returns -1

5.4.3.36 GetDigitalOutputRange()

```
static int GetDigitalOutputRange (
            int FirstOutputNumber,
            int LastOutputNumber,
            out int Value )  [inline], [static]
```

Get digital output bank. First and Last values are inclusive. Can return at most 32 bits.

Parameters

FirstOutputNumber	Number of the first digital output to be read
LastOutputNumber	Number of the last digital output to be read
Value	Integer value to receive the output status. This is a bitmap of the individial output statii.

Returns

On success returns 0, otherwise returns -1

5.4.3.37 GetProcessParameter()

```
static int GetProcessParameter (
            ProcessParameterType ProcessParameter,
            int Process,
            out Value Value )  [inline], [static]
```

Read process parameter values.

Parameters

ProcessParameter	Process parameter to be read
Process	Process to be read.
Value	Buffer to receive the value

Returns

On success returns 0, otherwise returns -1

5.4.3.38 GetSystemParameter()

```
static int GetSystemParameter (
            SystemParameterType SystemParameter,
            out Value Value )  [inline], [static]
```

Read system parameter values.

Parameters

SystemParameter	System parameter to be read
Value	Buffer to receive the value

Returns

On success returns 0, otherwise returns -1

5.4.3.39 GetTABLE() [1/4]

```
static int GetTABLE (
            int FirstTABLE,
            int Length,
            out double[] Values )  [inline], [static]
```

Read TABLE values.

Parameters

FirstTABLE	Number of the first TABLE to be read
Length	Number of TABLE values to read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.40 GetTABLE() [2/4]

```
static int GetTABLE (
            int FirstTABLE,
            int Offset,
            int Length,
            ref double[] Values )  [inline], [static]
```

Read TABLE values into an area of an array.

Parameters

FirstTABLE	Number of the first TABLE to be read
Offset	Offset into the Value array to the first entry to be written
Length	Number of TABLE values to read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.41 GetTABLE() [3/4]

```
static int GetTABLE (
            int FirstTABLE,
            ref double[] Values )  [inline], [static]
```

Read an array of TABLE values.

Parameters

FirstTABLE	Number of the first TABLE to be read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.42 GetTABLE() [4/4]

```
static int GetTABLE (
            int TABLE,
            out double Value )  [inline], [static]
```

Read a single TABLE value.

Parameters

TABLE	TABLE to be read
Value	Double that will receive the value read

Returns

On success returns 0, otherwise returns -1

5.4.3.43 GetVR() [1/4]

```
static int GetVR (
            int FirstVR,
            int Length,
            out double[] Values )  [inline], [static]
```

Read VR values.

Parameters

FirstVR	Number of the first VR to be read
Length	Number of VR values to read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.44 GetVR() [2/4]

```
static int GetVR (
            int FirstVR,
            int Offset,
            int Length,
            ref double[] Values )  [inline], [static]
```

Read VR values into an area of an array.

Parameters

FirstVR	Number of the first VR to be read
Offset	Offset into the Value array to the first entry to be written
Length	Number of VR values to read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.45 GetVR() [3/4]

```
static int GetVR (
            int FirstVR,
            ref double[] Values )  [inline], [static]
```

Read an array of VR values.

Parameters

FirstVR	Number of the first VR to be read
Values	Array of double that will receive the values read

Returns

On success returns 0, otherwise returns -1

5.4.3.46 GetVR() [4/4]

```
static int GetVR (
            int VR,
            out double Value )  [inline], [static]
```

Read a single VR value.

Parameters

VR	VR to be read
Value	Double that will receive the value read

Returns

On success returns 0, otherwise returns -1

5.4.3.47 IsFile()

```
static int IsFile (
            string FileName,
            out bool FileExists )  [inline], [static]
```

Check whether a file exists on the controller.

Parameters

FileName	Name of the file to check.
FileExists	True if the file exists.

Returns

On success returns 0, otherwise returns -1

5.4.3.48 IsOpen()

```
static bool IsOpen ( )  [inline], [static]
```

Check whether the connection to the PC-MCAT is open.

Returns

FALSE means that the connection is closed, TRUE means that the connection is open.

5.4.3.49 MoveAbsolute() [1/2]

```
static int MoveAbsolute (
            int BaseAxis,
            double[] Position )  [inline], [static]
```

Peforms the corresponding MOVEABS(...)AXIS(...) command on the Motion Coordinator Move axis to position(s) referenced with respect to the zero (home) position. Dispatch an absolute linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Position	The absolute position that specifies where to stop.

Returns

On success returns 0, otherwise returns -1

5.4.3.50 MoveAbsolute() [2/2]

```
static int MoveAbsolute (
            int BaseAxis,
            int Offset,
            int Length,
            double[] Position )  [inline], [static]
```

Peforms the corresponding MOVEABS(...)AXIS(...) command on the Motion Coordinator Move axis to position(s) referenced with respect to the zero (home) position. Dispatch an absolute linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Offset	Offset into the Position array to the first value to be written.
Length	Number of axes involved,
Position	Position">The absolute position that specifies where to stop.

Returns

On success returns 0, otherwise returns -1.

5.4.3.51 MoveAbsolute_SP() [1/2]

```
static int MoveAbsolute_SP (
            int BaseAxis,
            double[] Position )  [inline], [static]
```

Peforms the corresponding MOVEABSSP(...)AXIS(...) command on the Motion Coordinator Refer to MOVEABSSP documentation. Dispatch an absolute linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Position	The absolute position that specifies where to stop.

Returns

On success returns 0, otherwise returns -1

5.4.3.52 MoveAbsolute_SP() [2/2]

```
static int MoveAbsolute_SP (
            int BaseAxis,
            int Offset,
            int Length,
            double[] Position )  [inline], [static]
```

Peforms the corresponding MOVEABSSP(...)AXIS(...) command on the Motion Coordinator Refer to MOVEABSSP documentation. Dispatch an absolute linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Offset	Offset into the Position array to the first value to be written.
Length	Number of values to write.
Position	The absolute position that specifies where to stop.

Returns

On success returns 0, otherwise returns -1.

5.4.3.53 MoveAbsSeq() [1/4]

```
static int MoveAbsSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
            int MoveCount,
            int Options )  [inline], [static]
```

Dispatch a sequence of absolute moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Bit 0 0 The sequence loads MOVEABS, MOVECIRC etc.Bit 0 1 Sequence will load embedded speed moves MOVEABSSP, MOVECIRCSP etc.Bit 1 ReservedBit 2 0 Load a sequence of standard move types.Bit 2 1 Load a sequence of MOVEABSSEQ/MOVEABSSEQSP move types.Bits 8-13 Set to specify an offset between data points other than Axes.See the TrioBASIC MOVEABSSEQ documentation for more details

Returns

On success returns 0, otherwise returns -1

5.4.3.54 MoveAbsSeq() [2/4]

```
static int MoveAbsSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
```

```
            int MoveCount,
            int Options,
            double Radius )  [inline], [static]
```

Dispatch a sequence of absolute moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Bit 0 0 The sequence loads MOVEABS, MOVECIRC etc.Bit 0 1 Sequence will load embedded speed moves MOVEABSSP, MOVECIRCSP etc.Bit 1 ReservedBit 2 0 Load a sequence of standard move types.Bit 2 1 Load a sequence of MOVEABSSEQ/MOVEABSSEQSP move types.Bits 8-13 Set to specify an offset between data points other than Axes.See the TrioBASIC MOVEABSSEQ documentation for more details
Radius	The merging/filleting radius to be applied. 0 for no filleting.

Returns

On success returns 0, otherwise returns -1

5.4.3.55 MoveAbsSeq() [3/4]

```
static int MoveAbsSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
            int MoveCount,
            int Options,
            double Radius,
            int TableOutput )  [inline], [static]
```

Dispatch a sequence of absolute moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.

Parameters

Options	Bit 0 0 The sequence loads MOVEABS, MOVECIRC etc.Bit 0 1 Sequence will load embedded speed moves MOVEABSSP, MOVECIRCSP etc.Bit 1 ReservedBit 2 0 Load a sequence of standard move types.Bit 2 1 Load a sequence of MOVEABSSEQ/MOVEABSSEQSP move types.Bits 8-13 Set to specify an offset between data points other than Axes.See the TrioBASIC MOVEABSSEQ documentation for more details
Radius	The merging/filleting radius to be applied. 0 for no filleting.
TableOutput	Output data to TABLE. 0 means do not output data to TABLE, otherwise is the TABLE start index for the data.

Returns

On success returns 0, otherwise returns -1

5.4.3.56 MoveAbsSeq() [4/4]

```
static int MoveAbsSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
            int MoveCount,
            int Options,
            double Radius,
            int TableOutput,
            double TransitionAngle )  [inline], [static]
```

Dispatch a sequence of absolute moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start location of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Modifiers for the movements.

Bit 0 0 The sequence loads MOVEABS, MOVECIRC etc.

Bit 0 1 Sequence will load embedded speed moves MOVEABSSP, MOVECIRCSP etc.

Bit 1 Reserved

Bit 2 0 Load a sequence of standard move types.

Bit 2 1 Load a sequence of MOVEABSSEQ/MOVEABSSEQSP move types.

Bits 8-13 Set to specify an offset between data points other than Axes.

See the TrioBASIC MOVEABSSEQ documentation for more details

Parameters

Radius	The merging/filleting radius to be applied. 0 for no filleting.
TableOutput	Output data to TABLE. 0 means do not output data to TABLE, otherwise is the TABLE start index for the data.
TransitionAngle	Transition angle (in Radians) for any filleted arcs that are generated. Set to 0 if no transition curve is to be applied.

Returns

On success returns 0, otherwise returns -1

5.4.3.57 MoveCircular()

```
static int MoveCircular (
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction )  [inline], [static]
```

Performs the corresponding MOVECIRC(...)AXIS(...) command on the Motion Coordinator Dispatch a circular move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.

Returns

On success returns 0, otherwise returns -1

5.4.3.58 MoveCircular_SP()

```
static int MoveCircular_SP (
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction )  [inline], [static]
```

Performs the Corresponding MOVECIRCSP(...) AXIS(...) command on the Motion Controller. Dispatch a circular move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.

Returns

On success returns 0, otherwise returns -1

5.4.3.59 MoveHelical() [1/2]

```
static int MoveHelical (
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction,
            double DistanceAxis3 )  [inline], [static]
```

Performs the corresponding MHELICAL(...)AXIS(...) on command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.
DistanceAxis3	Distance to move on the third axis in the base vector.

Returns

On success returns 0, otherwise returns -1

5.4.3.60 MoveHelical() [2/2]

```
static int MoveHelical (
```

```
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction,
            double DistanceAxis3,
            int Mode )  [inline], [static]
```

Performs the corresponding MHELICAL(...)AXIS(...) on command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.
DistanceAxis3	Distance to move on the third axis in the base vector.
Mode	See the MHELICAL command documentation for more details.

Returns

On success returns 0, otherwise returns -1

5.4.3.61 MoveHelical_SP() [1/2]

```
static int MoveHelical_SP (
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction,
            double DistanceAxis3 )  [inline], [static]
```

Performs the corresponding MHELICALSP(...) AXIS(...) command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.
DistanceAxis3	Distance to move on the third axis in the base vector.

Returns

On success returns 0, otherwise returns -1

5.4.3.62 MoveHelical_SP() [2/2]

```
static int MoveHelical_SP (
            int BaseAxis,
            double EndAxis1,
            double EndAxis2,
            double CentreAxis1,
            double CentreAxis2,
            int Direction,
            double DistanceAxis3,
            int Mode )  [inline], [static]
```

Performs the corresponding MHELICALSP(...) AXIS(...) command on the Motion Coordinator Dispatch a helical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
EndAxis1	Position on the base axis to finish at.
EndAxis2	Position on the next axis in the base vector to finish at.
CentreAxis1	Position on the base axis about which to move.
CentreAxis2	Position on the next axis in the base vector about which to move.
Direction	Direction in which to move around the arc.
DistanceAxis3	Distance to move on the third axis in the base vector.
Mode	See the MHELICAL command documentation for more details.

Returns

On success returns 0, otherwise returns -1

5.4.3.63 MoveRelative() [1/2]

```
static int MoveRelative (
            int BaseAxis,
            double[] Distance )  [inline], [static]
```

Performs the corresponding MOVE(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Distance	Distance to be moved on each axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.64 MoveRelative() [2/2]

```
static int MoveRelative (
             int BaseAxis,
             int Offset,
             int Length,
             double[] Distance )  [inline], [static]
```

Performs the corresponding MOVE(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Offset	Offset into the Position array to the first value to be written.
Length	Number of values to write.
Distance	Distance to be moved on each axis.

Returns

On success returns 0, otherwise returns -1.

5.4.3.65 MoveRelative_SP() [1/2]

```
static int MoveRelative_SP (
             int BaseAxis,
             double[] Distance )  [inline], [static]
```

Performs the corresponding MOVESP(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Distance	Distance to be moved on each axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.66 MoveRelative_SP() [2/2]

```
static int MoveRelative_SP (
            int BaseAxis,
            int Offset,
            int Length,
            double[] Distance )  [inline], [static]
```

Performs the corresponding MOVESP(...) command on the Motion Coordinator Dispatch a relative linear move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Offset	Offset into the Position array to the first value to be written.
Length	Number of values to write.
Distance	Distance to be moved on each axis.

Returns

On success returns 0, otherwise returns -1.

5.4.3.67 MoveRelSeq() [1/4]

```
static int MoveRelSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
            int MoveCount,
            int Options )  [inline], [static]
```

Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.

Dispatch a sequence of relative moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Modifiers for the movements. See the TrioBASIC MOVESEQ documentation for more details.

Returns

On success returns 0, otherwise returns -1

5.4.3.68 MoveRelSeq() [2/4]

```
static int MoveRelSeq (
             int BaseAxis,
             int TableInput,
             int AxisCount,
             int MoveCount,
             int Options,
             double Radius )  [inline], [static]
```

Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.

Dispatch a sequence of relative moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Modifiers for the movements. See the TrioBASIC MOVESEQ documentation for more details.
Radius	The merging/filleting radius to be applied. 0 for no filleting.

Returns

On success returns 0, otherwise returns -1

5.4.3.69 MoveRelSeq() [3/4]

```
static int MoveRelSeq (
             int BaseAxis,
             int TableInput,
             int AxisCount,
             int MoveCount,
             int Options,
             double Radius,
             int TableOutput )  [inline], [static]
```

Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.

Dispatch a sequence of relative moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Modifiers for the movements. See the TrioBASIC MOVESEQ documentation for more details.
Radius	The merging/filleting radius to be applied. 0 for no filleting.
TableOutput	Output data to TABLE. 0 means do not output data to TABLE, otherwise is the TABLE start index for the data.

Returns

On success returns 0, otherwise returns -1

5.4.3.70 MoveRelSeq() [4/4]

```
static int MoveRelSeq (
            int BaseAxis,
            int TableInput,
            int AxisCount,
            int MoveCount,
            int Options,
            double Radius,
            int TableOutput,
            double TransitionAngle )  [inline], [static]
```

Performs the corresponding MOVESEQ(...) command on the Motion Coordinator.

Dispatch a sequence of relative moves on the given axes.

This function will wait until the API move buffer is available before loading a move.

The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
TableInput	Start of the move sequence in the TABLE data.
AxisCount	Number of axes involved in the move, in the range 2-6.
MoveCount	Number of move segments in the move sequence. Each move point requires 2 or 3 table points.
Options	Modifiers for the movements. See the TrioBASIC MOVESEQ documentation for more details.
Radius	The merging/filleting radius to be applied. 0 for no filleting.
TableOutput	Output data to TABLE. 0 means do not output data to TABLE, otherwise is the TABLE start index for the data.
TransitionAngle	Transition angle (in Radians) for any filleted arcs that are generated. Set to 0 if no transition curve is to be applied.

Returns

On success returns 0, otherwise returns -1

5.4.3.71 MoveSpherical() [1/2]

```
static int MoveSpherical (
             int BaseAxis,
             double Pos1X,
             double Pos1Y,
             double Pos1Z,
             double Pos2X,
             double Pos2Y,
             double Pos2Z,
             int Mode,
             int Direction )  [inline], [static]
```

Dispatch a spherical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Pos1X	First position for the base axis. See the MSPHERICAL documentation.
Pos1Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos1Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Pos2X	First position for the base axis. See the MSPHERICAL documentation.
Pos2Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos2Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Mode	/// 0 - specify the end point and mid point on curve 1 - specify end point and centre of sphere 2 - two mid point are specifiedand the curve completes a full circle 3 - mid point on curve and centre of sphere are specified and the curve completes the circle
Direction	Direction of travel along the arc.

Returns

On success returns 0, otherwise returns -1

5.4.3.72 MoveSpherical() [2/2]

```
static int MoveSpherical (
             int BaseAxis,
             double Pos1X,
             double Pos1Y,
             double Pos1Z,
             double Pos2X,
             double Pos2Y,
             double Pos2Z,
             int Mode,
             int Direction,
```

```
            int RotationU,
            int RotationV,
            int RotationW,
            int RotationP )  [inline], [static]
```

Dispatch a spherical move to the Motion Coordinator. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Pos1X	First position for the base axis. See the MSPHERICAL documentation.
Pos1Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos1Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Pos2X	First position for the base axis. See the MSPHERICAL documentation.
Pos2Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos2Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Mode	/// 0 - specify the end point and mid point on curve 1 - specify end point and centre of sphere 2 - two mid point are specifiedand the curve completes a full circle 3 - mid point on curve and centre of sphere are specified and the curve completes the circle
Direction	Direction of travel along the arc.
RotationU	If this parameter is non zero then the fourth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationV	If this parameter is non zero then the fifth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationW	If this parameter is non zero then the sixth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationP	If this parameter is non zero then the seventh axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.

Returns

On success returns 0, otherwise returns -1

5.4.3.73 MoveSpherical_SP() [1/2]

```
static int MoveSpherical_SP (
            int BaseAxis,
            double Pos1X,
            double Pos1Y,
            double Pos1Z,
            double Pos2X,
            double Pos2Y,
            double Pos2Z,
            int Mode,
            int Direction )  [inline], [static]
```

Dispatch a spherical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Pos1X	First position for the base axis. See the MSPHERICAL documentation.
Pos1Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos1Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Pos2X	First position for the base axis. See the MSPHERICAL documentation.
Pos2Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos2Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Mode	/// 0 - specify the end point and mid point on curve 1 - specify end point and centre of sphere 2 - two mid point are specifiedand the curve completes a full circle 3 - mid point on curve and centre of sphere are specified and the curve completes the circle
Direction	Direction of travel along the arc.

Returns

On success returns 0, otherwise returns -1

5.4.3.74 MoveSpherical_SP() [2/2]

```
static int MoveSpherical_SP (
            int BaseAxis,
            double Pos1X,
            double Pos1Y,
            double Pos1Z,
            double Pos2X,
            double Pos2Y,
            double Pos2Z,
            int Mode,
            int Direction,
            int RotationU,
            int RotationV,
            int RotationW,
            int RotationP )  [inline], [static]
```

Performs the corresponding MSPHERICAL(...) command on the Motion Coordinator Dispatch a spherical move to the Motion Coordinator with vector speed control when using multiple moves in the look ahead buffer. This function will wait until the API move buffer is available before loading a move. The API move buffer will be loaded into the Motion Generator move buffers when there are buffers available. If all buffers are available then this motion will be started on the next servo cycle.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Pos1X	First position for the base axis. See the MSPHERICAL documentation.
Pos1Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos1Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Pos2X	First position for the base axis. See the MSPHERICAL documentation.
Pos2Y	First position for the next axis in the base vector. See the MSPHERICAL documentation.
Pos2Z	First position for the third axis in the base vector. See the MSPHERICAL documentation.
Mode	0 - specify the end point and mid point on curve 1 - specify end point and centre of sphere 2 - two mid point are specifiedand the curve completes a full circle 3 - mid point on curve and centre of sphere are specified and the curve completes the circle

Parameters

Direction	Direction of travel along the arc.
RotationU	If this parameter is non zero then the fourth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationV	If this parameter is non zero then the fifth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationW	If this parameter is non zero then the sixth axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.
RotationP	If this parameter is non zero then the seventh axis in the base vector will a linear interpolation for the duration of the move. See the MSPHERICAL documentation.

Returns

On success returns 0, otherwise returns -1

5.4.3.75 Open()

```
static int Open (
            Callback Callback,
            IntPtr CallbackContext )  [inline], [static]
```

Open a conection to the PC-MCAT.

Parameters

Callback	Asynchronous callback function
CallbackContext	Callback function context

Returns

On success returns 0, otherwise returns GetLastError() of the operation that failed

5.4.3.76 PSwitch()

```
static int PSwitch (
            int PSwitchID,
            int Enable,
            int Axis,
            int Output,
            int State,
            double SetPosition,
            double ResetPosition )  [inline], [static]
```

Performs the corresponding PSWITCH(...) command on the Motion Coordinator

The Pswitch command allows an output to be set when a predefined position is reached, and to be reset when a second position is reached. Enable a PSWITCH.

Multiple PSWITCH's can be assigned to a single output

Parameters

PSwitchID	PSWITCH Number in the range 0..63.
Enable	1 => Enable on MPOS. 0 => Disable. 5 => Enable on DPOS.
Axis	Axis for this command. If value is -1 then the default base axis is used.
Output	Digital output to be set.
State	1 => turn the output on at the set_pos. 0 => turn the output off set the set_pos.
SetPosition	Axis position at which to set the output status.
ResetPosition	Axis position at which to reset the output status.

Returns

On success returns 0, otherwise returns -1

5.4.3.77 PSwitchOff()

```
static int PSwitchOff (
            int PSwitchID,
            int ResetOutput )  [inline], [static]
```

Performs the corresponding PSWITCH(Switch, OFF, Hold) command on the Motion Coordinator Disable a PSWITCH.

Parameters

PSwitchID	PSWITCH Number in the range 0..63.
ResetOutput	0 => The PSWITCH output will be held at the state it is when the disable is called. 1=> The PSWITCH output is forced off.

Returns

On success returns 0, otherwise returns -1

5.4.3.78 RapidStop()

```
static int RapidStop (
            int Mode )  [inline], [static]
```

Performs the corresponding RAPIDSTOP(...) command on the Motion Controller Cancel a currently running movement on all axes.

Parameters

Mode	Type of cancel to be performed. See the CANCEL command documentation.

Returns

On success returns 0, otherwise returns -1

5.4.3.79 Regist() [1/3]

```
static int Regist (
            int BaseAxis,
            int Mode )  [inline], [static]
```

Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Mode	Registration mode. See TrioBASIC REGIST command.

Returns

On success returns 0, otherwise returns -1

5.4.3.80 Regist() [2/3]

```
static int Regist (
            int BaseAxis,
            int Mode,
            int Channel,
            int Source,
            int Edge,
            int Window )  [inline], [static]
```

Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Mode	Registration mode. See TrioBASIC REGIST command.
Channel	Channel to monitor. See TrioBASIC REGIST command.
Source	Signal source. See TrioBASIC REGIST command.
Edge	0 => Rising Edge. 1 => Falling Edge.
Window	0 => No windowing. 1 => Position inside OPEN_WIN..CLOSE_WIN. 2 => Position outside OPEN_WIN..CLOSE_WIN.

Returns

On success returns 0, otherwise returns -1

5.4.3.81 Regist() [3/3]

```
static int Regist (
```

```
            int BaseAxis,
            int Mode,
            int Channel,
            int Source,
            int Edge,
            int Window,
            int RepeatCount,
            int TableStart )  [inline], [static]
```

Initiate the capture of an axis position when it sees a registration input or the Z mark on the encoder.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Mode	Registration mode. See TrioBASIC REGIST command.
Channel	Channel to monitor. See TrioBASIC REGIST command.
Source	Signal source. See TrioBASIC REGIST command.
Edge	0 => Rising Edge. 1 => Falling Edge.
Window	0 => No windowing. 1 => Position inside OPEN_WIN..CLOSE_WIN. 2 => Position outside OPEN_WIN..CLOSE_WIN.
RepeatCount	Number of registration capture positions to store in the table. Must be one or more.
TableStart	Start position in the table to store the registration capture positions.

Returns

On success returns 0, otherwise returns -1

5.4.3.82 Reverse()

```
static int Reverse (
            int BaseAxis )  [inline], [static]
```

Performs the corresponding REVERSE(...) AXIS(...) command on the Motion Coordinator Continuous reverse movement on the given axis.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.

Returns

On success returns 0, otherwise returns -1

5.4.3.83 RunProgram() [1/2]

```
static int RunProgram (
            string ProgramName )  [inline], [static]
```

Run a TrioBASIC program on a given process.

Parameters

ProgramName	Name of program to run.

Returns

On success returns 0, otherwise returns -1

5.4.3.84 RunProgram() [2/2]

```
static int RunProgram (
            string ProgramName,
            int ProcessNumber )  [inline], [static]
```

Run a TrioBASIC program on a given process.

Parameters

ProgramName	Name of program to run.
ProcessNumber	Number of the TrioBASIC process on which to run this program. If value is -1 then the next available process will be used.

Returns

On success returns 0, otherwise returns -1

5.4.3.85 SetAnalogueOutput()

```
static int SetAnalogueOutput (
            int OutputNumber,
            int Value )  [inline], [static]
```

Set single analogue output.

Parameters

OutputNumber	Number of the output to be written
Value	Integer value to be written to the output status

Returns

On success returns 0, otherwise returns -1

5.4.3.86 SetAxisParameter() [1/2]

```
static int SetAxisParameter (
            AxisParameterType AxisParameter,
```

```
            int Axis,
            Value Value )  [inline], [static]
```

Write axis parameter value.

Parameters

AxisParameter	Axis parameter to be written
Axis	Axis to be written. If this parameter is -1 then writes the current base axis.
Value	Pointer to an buffer to contains the value

Returns

On success returns 0, otherwise returns: -ValueType.Float32 => Paremeter requires a 32 bit floating point value -ValueType.Float64 => Parameter requires a 64 bit floating point value -ValueType.Int16 => Parameter requires a 16 signed bit integer value -ValueType.Int32 => Parameter requires a 32 signed bit integer value -ValueType.Int64 => Parameter requires a 64 signed bit integer value -ValueType.UInt16 => Parameter requires an unsigned 16 bit integer value -ValueType.UInt32 => Parameter requires an unsigned 32 bit integer value -ValueType.UInt64 => Parameter requires an unsigned 64 bit integer value

5.4.3.87 SetAxisParameter() [2/2]

```
static int SetAxisParameter (
            AxisParameterType AxisParameter,
            Value Value )  [inline], [static]
```

Write current base axis parameter value.

Parameters

AxisParameter	Axis parameter to be written
Value	Pointer to an buffer to contains the value

Returns

On success returns 0, otherwise returns: -ValueType.Float32 => Paremeter requires a 32 bit floating point value -ValueType.Float64 => Parameter requires a 64 bit floating point value -ValueType.Int16 => Parameter requires a 16 signed bit integer value -ValueType.Int32 => Parameter requires a 32 signed bit integer value -ValueType.Int64 => Parameter requires a 64 signed bit integer value -ValueType.UInt16 => Parameter requires an unsigned 16 bit integer value -ValueType.UInt32 => Parameter requires an unsigned 32 bit integer value -ValueType.UInt64 => Parameter requires an unsigned 64 bit integer value

5.4.3.88 SetBaseVector() [1/2]

```
static int SetBaseVector (
            int Offset,
            int Length,
            int[] BaseVector )  [inline], [static]
```

Set the BASE vector for the PC-MCAT API.

Parameters

BaseVector	Sequence of axis numbers.
Offset	Offset into the BaseVector array to the first entry to be written
Length	Number of values to write

Returns

On success returns 0, otherwise returns -1

5.4.3.89 SetBaseVector() [2/2]

```
static int SetBaseVector (
            int[] BaseVector )  [inline], [static]
```

Set the BASE vector for the PC-MCAT API.

Parameters

BaseVector	Sequence of axis numbers.

Returns

On success returns 0, otherwise returns -1

5.4.3.90 SetDigitalInvertInput()

```
static int SetDigitalInvertInput (
            int OutputNumber,
            int Value )  [inline], [static]
```

Set single digital input inversion.

Parameters

OutputNumber	Number of the input inversion to be written
Value	Status to be set. 0 => off, others => on

Returns

On success returns 0, otherwise returns -1

5.4.3.91 SetDigitalOutput()

```
static int SetDigitalOutput (
            int OutputNumber,
            int Value )  [inline], [static]
```

Set single digital output.

Parameters

OutputNumber	Number of the output to be written
Value	Status to be set. 0 => off, others => on

Returns

On success returns 0, otherwise returns -1

5.4.3.92 SetDigitalOutputRange()

```
static int SetDigitalOutputRange (
             int FirstOutputNumber,
             int LastOutputNumber,
             int Value )  [inline], [static]
```

Set digital output bank. First and last values are inclusive.

Parameters

FirstOutputNumber	Number of the first output to be written
LastOutputNumber	Number of the last output to be written
Value	Status to be set. This is a bitmap of the individial output statii.

Returns

On success returns 0, otherwise returns -1

5.4.3.93 SetGTA()

```
static int SetGTA (
             int GTAId,
             string GTAName,
             ref Target GTAValue )  [inline], [static]
```

Set a member of the Global Target Array.

Parameters

GTAId	ID of the GTA to set.
GTAName	Name of the GTA to be set. If gta_id is not -1 then this parameter is ignored.
GTAValue	Value of the GTA to be set.

Returns

On success returns 0, otherwise returns -1

5.4.3.94 SetObjectFrame()

```
static int SetObjectFrame (
            int ObjectFrameId,
            string ObjectFrameName,
            ref ObjectFrame ObjectFrameValue )  [inline], [static]
```

Set a member of the Object Frame Array.

Parameters

ObjectFrameId	ID of the ObjectFrame to set.
ObjectFrameName	Name of the ObjectFrame to be set. If ObjectFrameId is not -1 then this parameter is ignored.
ObjectFrameValue	Value of the ObjectFrame to be set.

Returns

On success returns 0, otherwise returns -1

5.4.3.95 SetProcessParameter()

```
static int SetProcessParameter (
            ProcessParameterType ProcessParameter,
            int Process,
            Value Value )  [inline], [static]
```

Write process parameter value.

Parameters

ProcessParameter	Process parameter to be written
Process	Process to be written.
Value	Pointer to an buffer to contains the value

Returns

On success returns 0, otherwise returns: -ValueType.Float32 => Paremeter requires a 32 bit floating point value -ValueType.Float64 => Parameter requires a 64 bit floating point value -ValueType.Int16 => Parameter requires a 16 signed bit integer value -ValueType.Int32 => Parameter requires a 32 signed bit integer value -ValueType.Int64 => Parameter requires a 64 signed bit integer value -ValueType.UInt16 => Parameter requires an unsigned 16 bit integer value -ValueType.UInt32 => Parameter requires an unsigned 32 bit integer value -ValueType.UInt64 => Parameter requires an unsigned 64 bit integer value

5.4.3.96 SetSystemParameter()

```
static int SetSystemParameter (
            SystemParameterType SystemParameter,
            Value Value )  [inline], [static]
```

Write axis parameter value.

Parameters

SystemParameter	Axis parameter to be written
Value	Pointer to an buffer to contains the value

Returns

On success returns 0, otherwise returns: -ValueType.Float32 => Paremeter requires a 32 bit floating point value -ValueType.Float64 => Parameter requires a 64 bit floating point value -ValueType.Int16 => Parameter requires a 16 signed bit integer value -ValueType.Int32 => Parameter requires a 32 signed bit integer value -ValueType.Int64 => Parameter requires a 64 signed bit integer value -ValueType.UInt16 => Parameter requires an unsigned 16 bit integer value -ValueType.UInt32 => Parameter requires an unsigned 32 bit integer value -ValueType.UInt64 => Parameter requires an unsigned 64 bit integer value

5.4.3.97 SetTABLE() [1/4]

```
static int SetTABLE (
            int FirstTABLE,
            double[] Values )  [inline], [static]
```

Write an array of TABLE values.

Parameters

FirstTABLE	Number of the first TABLE to be written
Values	Array values to write

Returns

On success returns 0, otherwise returns -1

5.4.3.98 SetTABLE() [2/4]

```
static int SetTABLE (
            int FirstTABLE,
            int Length,
            double[] Values )  [inline], [static]
```

Write TABLE value.

Parameters

FirstTABLE	Number of the first TABLE to be written
Length	Number of TABLE values to write
Values	Array of double with at least length elements

Returns

On success returns 0, otherwise returns -1

5.4.3.99 SetTABLE() [3/4]

```
static int SetTABLE (
             int FirstTABLE,
             int Offset,
             int Length,
             double[] Values )  [inline], [static]
```

Write TABLE value.

Parameters

FirstTABLE	Number of the first TABLE to be written
Offset	Offset into the Value array to the first entry to be read
Length	Number of TABLE values to write
Values	Array of double with at least length elements

Returns

On success returns 0, otherwise returns -1

5.4.3.100 SetTABLE() [4/4]

```
static int SetTABLE (
             int TABLE,
             double Value )  [inline], [static]
```

Write TABLE value.

Parameters

TABLE	Number of the TABLE to be written
Value	Value to be written

Returns

On success returns 0, otherwise returns -1

5.4.3.101 SetVR() [1/4]

```
static int SetVR (
             int FirstVR,
             double[] Values )  [inline], [static]
```

Write an array of VR values.

Parameters

FirstVR	Number of the first VR to be written
Values	Array values to write

Returns

On success returns 0, otherwise returns -1

5.4.3.102 SetVR() [2/4]

```
static int SetVR (
            int FirstVR,
            int Length,
            double[] Values )  [inline], [static]
```

Write VR value.

Parameters

FirstVR	Number of the first VR to be written
Length	Number of VR values to write
Values	Array of double with at least length elements

Returns

On success returns 0, otherwise returns -1

5.4.3.103 SetVR() [3/4]

```
static int SetVR (
            int FirstVR,
            int Offset,
            int Length,
            double[] Values )  [inline], [static]
```

Write VR value.

Parameters

FirstVR	Number of the first VR to be written
Offset	Offset into the Value array to the first entry to be read
Length	Number of VR values to write
Values	Array of double with at least length elements

Returns

On success returns 0, otherwise returns -1

5.4.3.104 SetVR() [4/4]

```
static int SetVR (
            int VR,
            double Value )  [inline], [static]
```

Write VR value.

Parameters

VR	Number of the VR to be written
Value	Value to be written

Returns

On success returns 0, otherwise returns -1

5.4.3.105 StopProgram() [1/2]

```
static int StopProgram (
            string ProgramName )  [inline], [static]
```

Stop all running instances of a TrioBASIC program.

Parameters

ProgramName	Name of program to stop.

Returns

On success returns 0, otherwise returns -1

5.4.3.106 StopProgram() [2/2]

```
static int StopProgram (
            string ProgramName,
            int ProcessNumber )  [inline], [static]
```

Stop a TrioBASIC program on a given process.

Parameters

ProgramName	Name of program to stop.
ProcessNumber	Stop the program "ProgramName" running on this process. If value is -1 then all running instances of "ProgramName" will be stopped.

Returns

On success returns 0, otherwise returns -1

5.4.3.107 Sync() [1/3]

```
static int Sync (
            int BaseAxis,
            int Control,
            int SyncTime,
            double SyncAxisPosition,
            int SyncAxis,
            double[] SyncPosition )  [inline], [static]
```

Syncronize movement of one axis to another.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	Specifies the type of SYNC to perform. 1 = Start synchronisation, requires minimum first 5 parameters 4 = Stop synchronisation, requires minimum first 2 parameters 10 = Re-synchronise to another axis, requires minimum first 5 parameters 20 = Re-synchronise to USER_FRAMEB, requires minimum first 5 parameters
SyncTime	Time to complete the synchronisation movement in servo period units.
SyncAxisPosition	Absolute position to synchronize to on the sync_axis.
SyncAxis	The axis to synchronise with.
SyncPosition	Array of 1..3 absolute positions for the base axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.108 Sync() [2/3]

```
static int Sync (
            int BaseAxis,
            int Control,
            int SyncTime,
            double SyncAxisPosition,
            int SyncAxis,
            double[] SyncPosition,
            int UserFrameID )  [inline], [static]
```

Syncronize movement to a user frame.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	Specifies the type of SYNC to perform.
SyncTime	Time to complete the synchronisation movement in servo period units.
SyncAxisPosition	The captured position on the sync_axis.
SyncAxis	The axis to synchronise with.
UserFrameID	ID of the user frame to syncrhonize to.
SyncPosition	Array of 3 absolute positions for the base axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.109 Sync() [3/3]

```
static int Sync (
             int BaseAxis,
             int Control,
             int SyncTime,
             double SyncAxisPosition,
             int SyncAxis,
             double[] SyncPosition,
             int UserFrameID,
             double[] SyncRotation )  [inline], [static]
```

Syncronize movement to a user frame.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	Specifies the type of SYNC to perform.
SyncTime	Time to complete the synchronisation movement in servo period units.
SyncAxisPosition	The captured position on the sync_axis.
SyncAxis	The axis to synchronise with.
UserFrameID	ID of the user frame to syncrhonize to.
SyncPosition	Array of 3 absolute positions for the base axis.
SyncRotation	Array of 1..3 absolute rotations for the base axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.110 SyncClear()

```
static int SyncClear (
             int BaseAxis,
             int SyncTime )  [inline], [static]
```

Stop the syncronized movement.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
SyncTime	Time to complete the synchronisation movement in servo period units.

Returns

On success returns 0, otherwise returns -1

5.4.3.111 SyncToGta() [1/2]

```
static int SyncToGta (
             int BaseAxis,
             int Control,
             int Time,
             double SyncPos,
             double SyncAxis,
             int ObjectFrame,
             int GTA )  [inline], [static]
```

Synchronise the robot TCP GTA with a moving object.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	1=>Start; 4=>Stop; 10=>Resynchronise
Time	Time to complete the synchronisation in milliseconds
SyncPos	Captured position on syncAxis.
SyncAxis	Axis to synchronise with.
ObjectFrame	Object frame related to selected GTA.
GTA	GTA to sync with.

Returns

On success returns 0, otherwise returns -1

5.4.3.112 SyncToGta() [2/2]

```
static int SyncToGta (
             int BaseAxis,
             int Control,
             int Time,
             double SyncPos,
             double SyncAxis,
             string ObjectFrame,
             string GTA )  [inline], [static]
```

Synchronise the robot TCP GTA with a moving object.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	1=>Start; 4=>Stop; 10=>Resynchronise
Time	Time to complete the synchronisation in milliseconds
SyncPos	Captured position on syncAxis.
SyncAxis	Axis to synchronise with.
ObjectFrame	Object frame related to selected GTA.
GTA	GTA to sync with.

Returns

On success returns 0, otherwise returns -1

5.4.3.113 SyncToObjectFrame() [1/2]

```
static int SyncToObjectFrame (
             int BaseAxis,
             int Control,
             int Time,
             int ObjectFrame,
             int GTA )  [inline], [static]
```

Synchronise the robot TCP GTA with a moving object.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	1=>Start; 4=>Stop; 10=>Resynchronise
Time	Time to complete the synchronisation in milliseconds
ObjectFrame	Object frame related to selected GTA.
GTA	GTA to sync with.

Returns

On success returns 0, otherwise returns -1

5.4.3.114 SyncToObjectFrame() [2/2]

```
static int SyncToObjectFrame (
             int BaseAxis,
             int Control,
             int Time,
             string ObjectFrame,
             string GTA )  [inline], [static]
```

Synchronise the robot TCP GTA with a moving object.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
Control	1=>Start; 4=>Stop; 10=>Resynchronise
Time	Time to complete the synchronisation in milliseconds
ObjectFrame	Object frame related to selected GTA.
GTA	GTA to sync with.

Returns

On success returns 0, otherwise returns -1

5.4.3.115 TcpCalibrate()

```
static int TcpCalibrate ( )  [inline], [static]
```

Returns

On success returns 0, otherwise returns -1

5.4.3.116 ToolOffset() [1/2]

```
static int ToolOffset (
            int ToolID,
            double XOffset,
            double YOffset,
            double ZOffset )  [inline], [static]
```

Set the tool offset from the world coordinate origin.

Parameters

ToolID	ID of the user defined tool. 0 is the default world coordinate system, 1-31 are user defined tools.
XOffset	Linear distance from the world coordinate origin to the tool origin along the x axis.
YOffset	Linear distance from the world coordinate origin to the tool origin along the y axis.
ZOffset	Linear distance from the world coordinate origin to the tool origin along the z axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.117 ToolOffset() [2/2]

```
static int ToolOffset (
            int ToolID,
            double XOffset,
            double YOffset,
            double ZOffset,
            double XRotation,
            double YRotation,
            double ZRotation )  [inline], [static]
```

Set the tool offset from the world coordinate origin.

Parameters

ToolID	ID of the user defined tool. 0 is the default world coordinate system, 1-31 are user defined tools.
XOffset	Linear distance from the world coordinate origin to the tool origin along the x axis.
YOffset	Linear distance from the world coordinate origin to the tool origin along the y axis.
ZOffset	Linear distance from the world coordinate origin to the tool origin along the z axis.
XRotation	Angular rotation from the world coordinate origin to the tool origin around the x axis.
YRotation	Angular rotation from the world coordinate origin to the tool origin around the y axis.
ZRotation	Angular rotation from the world coordinate origin to the tool origin around the z axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.118 ToolSelect()

```
static int ToolSelect (
            int BaseAxis,
            int ToolID )  [inline], [static]
```

Select the tool offset from the world coordinate origin.

Parameters

BaseAxis	Base axis for this command. If value is -1 then the default base axis is used.
ToolID	ID of the user defined tool. 0 is the default world coordinate system, 1-31 are user defined tools.

Returns

On success returns 0, otherwise returns -1

5.4.3.119 UnitClear()

```
static int UnitClear (
            int Slot )  [inline], [static]
```

Clear all the bits in the UNIT_ERROR system parameter.

Parameters

Slot	Communications slot to be cleared.

Returns

On success returns 0, otherwise returns -1

5.4.3.120 UserFrame() [1/3]

```
static int UserFrame (
            int FrameId )  [inline], [static]
```

Set the user frame offset from the world coordinate origin.

Parameters

FrameId	ID of the user defined frame. 0 is the default world coordinate system, 1-31 are user defined frames.

Returns

On success returns 0, otherwise returns -1

5.4.3.121 UserFrame() [2/3]

```
static int UserFrame (
            int FrameId,
            double XOffset,
            double YOffset,
            double ZOffset )  [inline], [static]
```

Set the user frame offset from the world coordinate origin.

Parameters

FrameId	ID of the user defined frame. 0 is the default world coordinate system, 1-31 are user defined frames.
XOffset	Linear distance from the world coordinate origin to the user origin along the x axis.
YOffset	Linear distance from the world coordinate origin to the user origin along the y axis.
ZOffset	Linear distance from the world coordinate origin to the user origin along the z axis.

Returns

On success returns 0, otherwise returns -1

5.4.3.122 UserFrame() [3/3]

```
static int UserFrame (
            int FrameId,
            double XOffset,
            double YOffset,
            double ZOffset,
            double XRotation,
            double YRotation,
            double ZRotation )  [inline], [static]
```

Set the user frame offset from the world coordinate origin.

Parameters

FrameId	ID of the user defined frame. 0 is the default world coordinate system, 1-31 are user defined frames.
XOffset	Linear distance from the world coordinate origin to the user origin along the x axis.
YOffset	Linear distance from the world coordinate origin to the user origin along the y axis.
ZOffset	Linear distance from the world coordinate origin to the user origin along the z axis.
XRotation	Angular rotation from the world coordinate origin to the user origin around the x axis.
YRotation	Angular rotation from the world coordinate origin to the user origin around the y axis.
ZRotation	Angular rotation from the world coordinate origin to the user origin around the z axis.

Returns

On success returns 0, otherwise returns -1

The documentation for this class was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.5 PCMCAT_API.SerialPortParameter Struct Reference

Serial port parameters.

Data Fields

- int baudrate
  Number of bits per second.
- int databits
  Number of bits per byte.
- Parity parity
  Error detection bit.
- StopBits stopbits
  Number of bits at the end of the byte.
- Handshake handshake
  Flow control.

5.5.1 Detailed Description

Serial port parameters.

5.5.2 Field Documentation

5.5.2.1 baudrate

```
int baudrate
```

Number of bits per second.

5.5.2.2 databits

```
int databits
```

Number of bits per byte.

5.5.2.3 handshake

```
Handshake handshake
```

Flow control.

5.5.2.4 parity

```
Parity parity
```

Error detection bit.

5.5.2.5 stopbits

```
StopBits stopbits
```

Number of bits at the end of the byte.

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.6 PCMCAT_API.Target Struct Reference

Robot target.

Data Fields

- byte active
  Point is in use.
- double x
  Coordinates.
- double y
- double z
- double u
- double v
- double w
- TargetPointType PointType
  Point type.
- byte RobotNumber
  ID of the robot.
- UInt16 revs
  Not documented.
- byte RobotConfig
  Not documented.
- byte ObjectFrame
  Frame configuration.
- byte ToolOffset
- byte RobotFrame
- double UserData
  Not documented.

5.6.1 Detailed Description

Robot target.

5.6.2 Field Documentation

5.6.2.1 active

```
byte active
```

Point is in use.

5.6.2.2 ObjectFrame

```
byte ObjectFrame
```

Frame configuration.

5.6.2.3 PointType

```
TargetPointType PointType
```

Point type.

5.6.2.4 revs

```
UInt16 revs
```

Not documented.

5.6.2.5 RobotConfig

```
byte RobotConfig
```

Not documented.

5.6.2.6 RobotFrame

```
byte RobotFrame
```

5.6.2.7 RobotNumber

```
byte RobotNumber
```

ID of the robot.

5.6.2.8 ToolOffset

```
byte ToolOffset
```

5.6.2.9 u

```
double u
```

5.6.2.10 UserData

```
double UserData
```

Not documented.

5.6.2.11 v

```
double v
```

5.6.2.12 w

```
double w
```

5.6.2.13 x

```
double x
```

Coordinates.

5.6.2.14 y

```
double y
```

5.6.2.15 z

```
double z
```

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.7 PCMCAT_API.Value Struct Reference

Structure used to send/receive values.

Data Structures

- struct ValueData

Public Member Functions

- Value (Value v)

  Copy constructor.
- Value (float v)

  Create a 32 bit floating point value.
- Value (double v)

  Create a 64 bit floating point value.
- Value (Int64 v)

  Create a 64 bit signed integer value.
- Value (Int32 v)

  Create a 32 bit signed integer value.
- Value (Int16 v)

  Create a 16 bit signed integer value.
- Value (UInt64 v)

  Create a 64 bit unsigned integer value.
- Value (UInt32 v)

  Create a 32 bit unsigned integer value.
- Value (UInt16 v)

  Create a 16 bit unsigned integer value.
- bool BitGet (int BitNumber)

  Returns the corresponding bit of the value. If the value is in a floating point format, the value is converted to an integer first.
- void BitSet (int Number, bool Value)

  Sets the corresponding bit of the value. If the value is in a floating point format, the value is converted to an integer first.
- void BitSet (int Number)
- void BitClear (int Number)
- float AsFloat ()
- double AsDouble ()
- Int64 AsInt64 ()
- Int32 AsInt32 ()
- Int16 AsInt16 ()
- UInt64 AsUint64 ()
- UInt32 AsUint32 ()
- UInt16 AsUint16 ()

Static Public Member Functions

- static implicit operator Value (float v)

  Implicit cast from a 32 bit floating point value.
- static implicit operator Value (double v)

  Implicit cast from a 64 bit floating point value.
- static implicit operator Value (Int64 v)

  Implicit cast from a 64 bit signed integer value.
- static implicit operator Value (Int32 v)

  Implicit cast from a 32 bit signed integer value.

- static implicit operator Value (Int16 v)

  Implicit cast from a 16 bit signed integer value.
- static implicit operator Value (UInt64 v)

  Implicit cast from a 64 bit unsigned integer value.
- static implicit operator Value (UInt32 v)

  Implicit cast from a 32 bit unsigned integer value.
- static implicit operator Value (UInt16 v)

  Implicit cast from a 16 bit unsigned integer value.
- static implicit operator float (Value v)

  Implicit cast to a 32 bit floating point value.
- static implicit operator double (Value v)

  Implicit cast to a 64 bit floating point value.
- static implicit operator Int64 (Value v)

  Implicit cast to a 64 bit signed integer value.
- static implicit operator Int32 (Value v)

  Implicit cast to a 32 bit signed integer value.
- static implicit operator Int16 (Value v)

  Implicit cast to a 16 bit signed integer value.
- static implicit operator UInt64 (Value v)

  Implicit cast to a 64 bit unsigned integer value.
- static implicit operator UInt32 (Value v)

  Implicit cast to a 16 bit unsigned integer value.
- static implicit operator UInt16 (Value v)

  Implicit cast to a 16 bit unsigned integer value.

Data Fields

- ValueType Type

  Data type.
- ValueData Data

  Value.

5.7.1 Detailed Description

Structure used to send/receive values.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Value() [1/9]

```
Value (
            Value v )  [inline]
```

Copy constructor.

Parameters

v	Value to be copied

5.7.2.2 Value() [2/9]

```
Value (
            float v )  [inline]
```

Create a 32 bit floating point value.

Parameters

v	Value to be set

5.7.2.3 Value() [3/9]

```
Value (
            double v )  [inline]
```

Create a 64 bit floating point value.

Parameters

v	Value to be set

5.7.2.4 Value() [4/9]

```
Value (
            Int64 v )  [inline]
```

Create a 64 bit signed integer value.

Parameters

v	Value to be set

5.7.2.5 Value() [5/9]

```
Value (
            Int32 v )  [inline]
```

Create a 32 bit signed integer value.

Parameters

v	Value to be set

5.7.2.6 Value() [6/9]

```
Value (
            Int16 v )  [inline]
```

Create a 16 bit signed integer value.

Parameters

v	Value to be set

5.7.2.7 Value() [7/9]

```
Value (
            UInt64 v )  [inline]
```

Create a 64 bit unsigned integer value.

Parameters

v	Value to be set

5.7.2.8 Value() [8/9]

```
Value (
            UInt32 v )  [inline]
```

Create a 32 bit unsigned integer value.

Parameters

v	Value to be set

5.7.2.9 Value() [9/9]

```
Value (
            UInt16 v )  [inline]
```

Create a 16 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3 Member Function Documentation

5.7.3.1 AsDouble()

```
double AsDouble ( )  [inline]
```

5.7.3.2 AsFloat()

```
float AsFloat ( )  [inline]
```

5.7.3.3 AsInt16()

```
Int16 AsInt16 ( )  [inline]
```

5.7.3.4 AsInt32()

```
Int32 AsInt32 ( )  [inline]
```

5.7.3.5 AsInt64()

```
Int64 AsInt64 ( )  [inline]
```

5.7.3.6 AsUint16()

```
UInt16 AsUint16 ( )  [inline]
```

5.7.3.7 AsUint32()

```
UInt32 AsUint32 ( )  [inline]
```

5.7.3.8 AsUint64()

```
UInt64 AsUint64 ( )  [inline]
```

5.7.3.9 BitClear()

```
void BitClear (
            int Number )  [inline]
```

5.7.3.10 BitGet()

```
bool BitGet (
            int BitNumber )  [inline]
```

Returns the corresponding bit of the value. If the value is in a floating point format, the value is converted to an integer first.

Parameters

BitNumber	Number of the bit to be returned

Returns

The value of the BitNumber bit.

5.7.3.11 BitSet() [1/2]

```
void BitSet (
            int Number )  [inline]
```

5.7.3.12 BitSet() [2/2]

```
void BitSet (
            int Number,
            bool Value )  [inline]
```

Sets the corresponding bit of the value. If the value is in a floating point format, the value is converted to an integer first.

Parameters

Number	Number of the bit to be returned
Value	Value of the bit.

5.7.3.13 operator double()

```
static implicit operator double (
            Value v )  [inline], [static]
```

Implicit cast to a 64 bit floating point value.

Parameters

v	Value to be set

5.7.3.14 operator float()

```
static implicit operator float (
            Value v )  [inline], [static]
```

Implicit cast to a 32 bit floating point value.

Parameters

v	Value to be set

5.7.3.15 operator Int16()

```
static implicit operator Int16 (
            Value v )  [inline], [static]
```

Implicit cast to a 16 bit signed integer value.

Parameters

v	Value to be set

5.7.3.16 operator Int32()

```
static implicit operator Int32 (
            Value v )  [inline], [static]
```

Implicit cast to a 32 bit signed integer value.

Parameters

v	Value to be set

5.7.3.17 operator Int64()

```
static implicit operator Int64 (
            Value v )  [inline], [static]
```

Implicit cast to a 64 bit signed integer value.

Parameters

v	Value to be set

5.7.3.18 operator UInt16()

```
static implicit operator UInt16 (
            Value v )  [inline], [static]
```

Implicit cast to a 16 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3.19 operator UInt32()

```
static implicit operator UInt32 (
            Value v )  [inline], [static]
```

Implicit cast to a 16 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3.20 operator UInt64()

```
static implicit operator UInt64 (
            Value v )  [inline], [static]
```

Implicit cast to a 64 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3.21 operator Value() [1/8]

```
static implicit operator Value (
            double v )  [inline], [static]
```

Implicit cast from a 64 bit floating point value.

Parameters

v	Value to be set

5.7.3.22 operator Value() [2/8]

```
static implicit operator Value (
            float v )  [inline], [static]
```

Implicit cast from a 32 bit floating point value.

Parameters

v	Value to be set

5.7.3.23 operator Value() [3/8]

```
static implicit operator Value (
            Int16 v )  [inline], [static]
```

Implicit cast from a 16 bit signed integer value.

Parameters

v	Value to be set

5.7.3.24 operator Value() [4/8]

```
static implicit operator Value (
            Int32 v )  [inline], [static]
```

Implicit cast from a 32 bit signed integer value.

Parameters

v	Value to be set

5.7.3.25 operator Value() [5/8]

```
static implicit operator Value (
            Int64 v )  [inline], [static]
```

Implicit cast from a 64 bit signed integer value.

Parameters

v	Value to be set

5.7.3.26 operator Value() [6/8]

```
static implicit operator Value (
            UInt16 v )  [inline], [static]
```

Implicit cast from a 16 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3.27 operator Value() [7/8]

```
static implicit operator Value (
            UInt32 v )  [inline], [static]
```

Implicit cast from a 32 bit unsigned integer value.

Parameters

v	Value to be set

5.7.3.28 operator Value() [8/8]

```
static implicit operator Value (
            UInt64 v )  [inline], [static]
```

Implicit cast from a 64 bit unsigned integer value.

Parameters

v	Value to be set

5.7.4 Field Documentation

5.7.4.1 Data

```
ValueData Data
```

Value.

5.7.4.2 Type

```
ValueType Type
```

Data type.

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

5.8 PCMCAT_API.Value.ValueData Struct Reference

Public Member Functions

- ValueData (float v)
- ValueData (double v)
- ValueData (Int64 v)
- ValueData (Int32 v)
- ValueData (Int16 v)
- ValueData (UInt64 v)
- ValueData (UInt32 v)
- ValueData (UInt16 v)

Data Fields

- float f32

  32 bit floating point value
- double f64

  64 bit floating point value
- Int16 i16

  16 bit signed integer value
- Int32 i32

  32 bit signed integer value
- Int64 i64

  64 bit signed integer value
- UInt16 u16

  16 bit unsigned integer value
- UInt32 u32

  32 bit unsigned integer value
- UInt64 u64

  64 bit unsigned integer value

5.8.1 Constructor & Destructor Documentation

5.8.1.1 ValueData() [1/8]

```
ValueData (
            float v )  [inline]
```

5.8.1.2 ValueData() [2/8]

```
ValueData (
            double v )  [inline]
```

5.8.1.3 ValueData() [3/8]

```
ValueData (
            Int64 v )  [inline]
```

5.8.1.4 ValueData() [4/8]

```
ValueData (
            Int32 v )  [inline]
```

5.8.1.5 ValueData() [5/8]

```
ValueData (
            Int16 v )  [inline]
```

5.8.1.6 ValueData() [6/8]

```
ValueData (
            UInt64 v )  [inline]
```

5.8.1.7 ValueData() [7/8]

```
ValueData (
            UInt32 v )  [inline]
```

5.8.1.8 ValueData() [8/8]

```
ValueData (
            UInt16 v )  [inline]
```

5.8.2 Field Documentation

5.8.2.1 f32

```
float f32
```

32 bit floating point value

5.8.2.2 f64

```
double f64
```

64 bit floating point value

5.8.2.3 i16

```
Int16 i16
```

16 bit signed integer value

5.8.2.4 i32

```
Int32 i32
```

32 bit signed integer value

5.8.2.5 i64

```
Int64 i64
```

64 bit signed integer value

5.8.2.6 u16

```
UInt16 u16
```

16 bit unsigned integer value

5.8.2.7 u32

```
UInt32 u32
```

32 bit unsigned integer value

5.8.2.8 u64

```
UInt64 u64
```

64 bit unsigned integer value

The documentation for this struct was generated from the following file:

- platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs

Chapter 6

File Documentation

6.1 platform/x64/common/pcmcat_api/pcmcat_net/pcmcat_api.cs File Reference

Data Structures

- class PCMCAT_API

  Implements the PC-MCAT shared memory API.
- class PCMCAT_API.CallbackData

  Data type for the callback function.
- struct PCMCAT_API.Value

  Structure used to send/receive values.
- struct PCMCAT_API.Value.ValueData
- struct PCMCAT_API.SerialPortParameter

  Serial port parameters.
- struct PCMCAT_API.EventParameters

  Event registration definition. Not all parameters are used by all event types.
- struct PCMCAT_API.Target

  Robot target.
- struct PCMCAT_API.ObjectFrame

  Robot object frame.

Namespaces

- namespace TrioMotion
- namespace TrioMotion.PCMCAT