Path Objects

Introduction

A **Path** object manages coordinated multi-axis motion profiles. It is used when the motion profiles in an N-Dimensional space are required to follow a specific coordinated trajectory. Motion paths are constructed with high level linear and arc segments and downloaded to the controller. The controller calculates the real-time individual axis profiles.

Generally, Path motion is used when the trajectory through space is more important than the final target position. Several different algorithms can be applied to convert the linear and arc segment path into an interpolated trajectory.

Path trajectory generation is now supported by PT, PVT, SPLINE, BESSEL, BSPLINE, and BSPLINE2 algorithms. Blending of the corners is only available for the 2 bspline algorithms. Blending of a corner is when the path does not hit the corner but goes through a smooth arc.

Methods

Create, Delete, Validate Methods

mpiPath**Create**	Create a Path object
mpiPath**Delete**	Delete a Path object
mpiPath**Validate**	Validate a Path object

Configuration and Information Methods

mpiPath**ParamsGet**

mpiPath**ParamsSet**

Relational Methods

mpiPath**Append**

Action Methods

mpiPath**MotionParamsGenerate**

Data Types

- MPIPath**Arc**
- MPIPath**ArcCenter**
- MPIPath**ArcEndPoint**
- MPIPath**Attr**
- MPIPath**Direction**
- MPIPath**Element**
- MPIPath**ElementAttributes**
- MPIPath**ElementAttrMask**
- MPIPath**ElementType**
- MPIPath**Line**
- MPIPath**Message**
- MPIPath**Params**
- MPIPath**Point**

Macros

- mpiPath**ElementTYPE**
- mpiPath**ElementAttrMaskBIT**
- mpiPath**ElementATTR**

Constants

- MPIPathPoint**DIMENSION_MAX**

mpiPathCreate

Declaration

`MPIPath mpiPathCreate();`

Required Header

stdmpi.h

Description

PathCreate creates a Path object.

Return Values	
handle	to a Path object
MPIHandleVOID	if the object could not be created

See Also

mpiPathDelete | mpiPathValidate

mpiPathDelete

Declaration

```
long mpiPathDelete(MPIPath path);
```

Required Header stdmpi.h

Description **PathDelete** deletes a Path object and invalidates its handle (***path***). PathDelete is the equivalent of a C++ destructor.

Return Values

MPIMessageOK	if *PathDelete* successfully deletes the Path object and invalidates its handle

See Also mpiPathCreate | mpiPathValidate

mpiPathValidate

Declaration

```
long mpiPathValidate(MPIPath path);
```

Required Header stdmpi.h

Description **PathValidate** validates the Path object and its handle (***command***)

Return Values	
MPIMessageOK	if the Path object and its handle are valid

See Also mpiPathCreate | mpiPathDelete

mpiPathParamsGet

Declaration

```
long mpiPathParamsGet(MPIPath         path,
                      MPIPathParams   *params,
                      void            *external),
```

Required Header stdmpi.h

Description

PathParamsGet reads the parameters for a path object and writes them into the structure pointed to by ***params***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL)

path	a handle to a Path object
***params**	a pointer to a MPIPathParams structure
***external**	a pointer to a void or NULL

Return Values	
MPIMessageOK	if *PathParamsGet* successfully reads the path parameters.

See Also mpiPathParamsSet

mpiPathParamsSet

Declaration

```
long mpiPathParamsSet(MPIPath         path,
                      MPIPathParams   *params,
                      void            *external),
```

Required Header

stdmpi.h

Description

PathParamsSet writes the parameters from the structure pointed to by ***params*** into the Path object. Also, it writes the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL) into the Path object.

path	a handle to a Path object
***params**	a pointer to a MPIPathParams structure
***external**	a pointer to a void or NULL

Return Values

MPIMessageOK	if *PathParamsSet* successfully writes the path parameters.

See Also

mpiPathParamsGet

mpiPathAppend

Declaration

```
long mpiPathAppend(MPIPath        path,
                   MPIPathElement *element);
```

Required Header

stdmpi.h

Description

PathAppend adds an array of path elements pointed to by ***element*** to the end of the ***path*** stored in the Path object.

Return Values	
MPIMessageOK	if *PathAppend* successfully adds the elements to the path.

See Also

mpiPathCreate | mpiPathMotionParamsGenerate

mpiPathMotionParamsGenerate

Declaration

```
long mpiPathMotionParamsGenerate(MPIPath            path,
                                 MPIMotionParams   *params);
```

Required Header

stdmpi.h

Description

PathMotionParamsGenerate calculates a list of points from the path object and writes them into the motion params structure pointed to by params. After the motion params are generated, they can be passed to mpiMotionStart(...). Path generated params are supported with the following motion types: MPIMotionTypePT

- MPIMotionTypePTF
- MPIMotionTypePVT
- MPIMotionTypePVTF
- MPIMotionTypeSPLINE
- MPIMotionTypeBESSEL
- MPIMotionTypeBSPLINE
- MPIMotionTypeBSPLINE2

To create a path, use mpiPathCreate(...) to create a path object. Initialize the path parameters with mpiPathParamsGet(...) / mpiPathParamsSet(...). Then add path elements (line, arc, etc.) with mpiPathAppend(...). Before calling mpiPathMotionParamsGenerate(...) make sure to specify the MPIMotionPoint{...} values in the MPIMotionParams structure. It is very important to set point.final = TRUE or FALSE before calling mpiPathMotionParamsGenerate(...).

path	a handle to a Path object
***params**	a pointer to a MPIMotionParams structure.

Return Values

See Also

mpiPathCreate | mpiPathParamsGet | mpiPathParamsSet | mpiPathAppend | mpiMotionStart

MPIPathArc

MPIPathArc

```
typedef struct MPIPathArc {
        struct {
                double  start;
                double  included;
        } angle;
        double  radius;
} MPIPathArc;
```

Description

PathArc specifies the parameters for an arc path element. It supports 2 dimensional arcs only. All arcs start at the end position for the last path element added to the path or the present command position if the arc is the first element in the path.

start	This value defines the arc's starting angle. Units are in degrees.
included	This value defines the relative travel angle. Units are in degrees. Positive values specify counter-clockwise motion and negative values specify clockwise motion.
radius	This value defines the distance from the center to the arc edge. Units are in counts.

See Also

MPIPathElement | MPIPathParams | MPIPathArcCenter

MPIPathArcCenter

MPIPathArcCenter

```
typedef struct MPIPathArcCenter {
        MPIPathPoint    center;
        double                  angle;
} MPIPathArcCenter;
```

Description

PathArcCenter specifies the parameters for an arc path element. It supports 2 dimensional arcs only. All arcs start at the end position for the last path element added to the path or the present command position if the arc is the first element in the path.

center	This structure defines the coordinates for the center point of the arc. Please see MPIPathPoint data type documentation for more information.
angle	This value defines the relative travel angle. Units are in degrees. Positive values specify counter-clockwise motion and negative values specify clockwise motion.

See Also

MPIPathElement | MPIPathParams | MPIPathArc

MPIPathArcEndPoint

MPIPathArcEndPoint

```
typedef struct MPIPathArcEndPoint {
        MPIPathPoint            center;
        MPIPathPoint            endPoint;
        MPIPathDirection        direction;
} MPIPathArcEndPoint;
```

Description

PathArcEndPoint specifies the parameters for an arc path element. It supports 2 dimensional arcs only. All arcs start at the end position for the last path element added to the path or the present command position if the arc is the first element in the path.

center	This structure defines the coordinates for the center point of the arc. Please see MPIPathPoint data type documentation for more information.
endPoint	This structure defines the coordinates for the final point of the arc. Please see MPIPathPoint data type documentation for more information.
direction	This value defines the travel direction, counter-clockwise or clockwise. Please see MPIPathDirection data type documentation for more information.

See Also

MPIPathElement | MPIPathParams | MPIPathDirection

MPIPathAttr

MPIPathAttr

```
    typedef enum {
            MPIPathElementAttrINVALID    = -1,

            MPIPathElementAttrRELATIVE   = MPIPathElementAttrLAST - 1,
            MPIPathElementAttrID         = MPIPathElementAttrLAST - 2,
            MPIPathElementAttrVELOCITY   = MPIPathElementAttrLAST - 3,
            MPIPathElementAttrACCEL      = MPIPathElementAttrLAST - 4,
            MPIPathElementAttrTIMESLICE  = MPIPathElementAttrLAST - 5,
            MPIPathElementAttrCOUNT      = MPIPathElementAttrLAST -
    MPIPathElementAttrFIRST,
    } MPIPathAttr;
```

Description

In **PathAttr**, the path attributes are used to generate the path attribute masks to enable features with mpiPathAppend(…). Please see MPIPathElementAttrMask data type for more information.

See Also

mpiPathAppend

MPIPathDirection

MPIPathDirection

```
typedef enum {
    MPIPathDirectionCW  = -1,
    MPIPathDirectionCCW = 1,
} MPIPathDirection;
```

Description

MPIPathDirectionCW	This value defines the clockwise direction.
MPIPathDirectionCCW	This value defines the counter-clockwise direction.

See Also

MPIPathArcEndPoint

MPIPathElement

MPIPathElement

```
typedef struct MPIPathElement {
        MPIPathElementType              type;
        long                            blending;
        union {
                MPIPathArc              arc;
                MPIPathArcCenter        arcCenter;
                MPIPathArcEndPoint      arcEndPoint;
                MPIPathLine             line;
        } params;

        MPIPathElementAttributes        attributes;
} MPIPathElement;
```

Description

type	This value defines the type of path element. Please see MPIPathElementType{…} data type documentation for more information.
blending	This value determines whether the corners of the path are rounded or sharp. When set to TRUE, blending is enabled, causing rounded corners. When set to FALSE, blending is disabled, causing sharp corners.
arc	This structure defines the arc's start angle, included angle, and radius. This structure is used when the type is MPIPathElementTypeARC. Please see MPIPathArc data type documentation for more information.
arcCenter	This structure defines the arc's center and angle. This structure is used when the type is MPIPathElementTypeARC_CENTER. Please see MPIPathArcCenter data type documentation for more information.
arcEndPoint	This structure defines the arc's center, end point, and direction. This structure is used when the type is MPIPathElementTypeARC_END_POINT. Please see MPIPathArcEndPoint data type documentation for more information.
line	This structure defines the coordinates for a linear element. This structure is used when the type is MPIPathElementTypeLINE. Please see MPIPathLine data type documentation for more information.
attributes	This structure defines the attributes for a path element. Please see MPIPathElementAttributes data type documentation for more information.

See Also

mpiPathAppend

MPIPathElementAttributes

MPIPathElementAttributes

```
typedef struct MPIPathElementAttributes {
        long    id;                                     /* MPIPathAttrID
*/
        double  velocity;                       /* MPIPathAttrVELOCITY          */
        double  acceleration;           /* MPIPathAttrACCELERATION      */
        double  timeSlice;                      /* MPIPathAttrTIMESLICE         */
} MPIPathElementAttributes;
```

Description

In **PathElementAttributes**, the path attributes define the parameters to be used when specific features are enabled with the path element attribute masks. When using these attributes, be sure to enable the feature with the appropriate MPIPathElementAttrMask{…}.

id	This value defines an identification number to be stored in the path element. During path profile execution, at the start of each element the controller loads the id into the axis' ElementID field. The application can query the controller's axis memory to monitor the path element execution. The id is limited to 16-bit resolution by the controller firmware.
velocity	This value defines the velocity for the path element.
acceleration	This value defines the acceleration for the path element.
timeSlice	This value defines the time between interpolation points for the path element. The practical range for the time slice is from 10 msec (.01) to 100 msec (.1). Larger time slice values produce smoother (lower acceleration), less accurate paths. Smaller time slice values produce more accurate (both position and velocity) paths with higher peak accelerations.

See Also

MPIPathElementAttrMask

MPIPathElementAttrMask

MPIPathElementAttrMask

```
typedef enum {
        MPIPathElementAttrMaskRELATIVE,   =
mpiPathElementAttrMaskBIT(MPIPathElementAttrRELATIVE),
        MPIPathElementAttrMaskID,         =
mpiPathElementAttrMaskBIT(MPIPathElementAttrID),
        MPIPathElementAttrMaskVELOCITY,   =
mpiPathElementAttrMaskBIT(MPIPathElementAttrVELOCITY),
        MPIPathElementAttrMaskACCEL,      =
mpiPathElementAttrMaskBIT(MPIPathElementAttrACCEL),
        MPIPathElementAttrMaskTIMESLICE,  =
mpiPathElementAttrMaskBIT(MPIPathElementAttrTIMESLICE),

        MPIPathElementAttrMaskALL         = -1 << MPIPathElementAttrFIRST,
} MPIPathElementAttrMask;
```

Description

In **PathElementAttrMask**, the path attribute masks are used to enable features with mpiPathAppend(…). The masks are ORed with the MPIPathElementType to enable each feature.

MPIPathElementAttrMaskRELATIVE	This mask enables relative coordinates for path motion. This feature is not supported and is reserved for future use.
MPIPathElementAttrMaskID	This mask enables an identification tag to be stored in the path. Each element can have a unique identification. Please see MPIPathElementAttributes{…} data type documentation for more information.
MPIPathElementAttrMaskVELOCITY	This mask enables a path velocity to be specified for each element. Please see MPIPathElementAttributes{…} data type documentation for more information.
MPIPathElementAttrMaskACCEL	This mask enables a path acceleration to be specified for each element. Please see MPIPathElementAttributes{…} data type documentation for more information.

See Also

MPIPathElementType | mpiPathAppend

MPIPathElementType

MPIPathElementType

```
typedef enum {
        MPIPathElementTypeINVALID,

        MPIPathElementTypeARC,  /* only 2D */
        MPIPathElementTypeARC_CENTER,  /* only 2D */
        MPIPathElementTypeARC_END_POINT, /* both 2D and 3D */
        MPIPathElementTypeHELIX, /* not currently supported */
        MPIPathElementTypeIO, /* not currently supported */
        MPIPathElementTypeLINE, /* both 2D and 3D */

        MPIPathElementTypeMASK,
} MPIPathElementType;
```

Description

MPIPathElementTypeARC	This type generates an arc specified by the arc's start angle, included angle, and radius.
MPIPathElementTypeARC_CENTER	This type generates an arc specified by the arc's center and angle.
MPIPathElementTypeARC_END_POINT	This type generates an arc specified by the arc's center, end point, and direction.
MPIPathElementTypeLINE	This type generates a line specified by the position coordinates.

See Also

MPIPathArc | MPIPathLine

MPIPathLine

MPIPathLine

```
typedef struct MPIPathLine {
        MPIPathPoint    point;
} MPIPathLine;
```

Description

PathLine specifies the parameters for a linear path element. It supports up to MPIPathPointDIMENSION_MAX dimensions. All lines start at the end position for the last path element added to the path or the present command position if the line is the first element in the path.

point	This structure defines the end point coordinates for the linear segment.

See Also

MPIPathElement | MPIPathParams | MPIPathPointDIMENSION_MAX

MPIPathMessage

MPIPathMessage

```
typedef enum {

        MPIPathMessagePATH_INVALID,
        MPIPathMessageILLEGAL_DIMENSION,
        MPIPathMessageILLEGAL_ELEMENT,
        MPIPathMessageARC_ILLEGAL_DIMENSION,
        MPIPathMessageHELIX_ILLEGAL_DIMENSION,
        MPIPathMessageILLEGAL_RADIUS,
        MPIPathMessagePATH_TOO_LONG,
        MPIPathMessageILLEGAL_VELOCITY,
        MPIPathMessageILLEGAL_ACCELERATION,
        MPIPathMessageILLEGAL_TIMESLICE,
        MPIPathMessageINVALID_BLENDING,
} MPIPathMessage;
```

Description

MPIPathMessagePATH_INVALID

The path object is not valid. This message code is returned by a path method if the path object handle is not valid. This problem can be caused by a failed mpiPathCreate(...). To prevent this problem, check your path objects after creation by using mpiPathValidate(...).

MPIPathMessageILLEGAL_DIMENSION

The path dimensions are not valid. This message code is returned by mpiPathParamsSet(...) or mpiPathMotionParamsGenerate(...) if the path dimension is less than one or greater than or equal to MPIPathPointDIMENSION_MAX. Also, this message code is returned if specific path element types have dimension restrictions. For example, the ARC type is limited to 2 dimensions and the ARC_END_POINT type is limited to 3 dimensions. To correct this problem, select an appropriate dimension for the path element type.

MPIPathMessageILLEGAL_ELEMENT

The path element type is not valid. This message code is returned by mpiPathAppend(...) if the specified path element type is not a member of the MPIPathElementType enumeration.

MPIPathMessageARC_ILLEGAL_DIMENSION

The path element arc dimension is not valid. This message code is returned by mpiPathAppend(...) if the ARC or ARC_CENTER element is not 2 dimensions. To correct this problem, set the path dimension to 2.

MPIPathMessageHELIX_ILLEGAL_DIMENSION

Not supported.

MPIPathMessageILLEGAL_RADIUS

The path element arc radius is not valid. This message code is returned by mpiPathAppend(...) if the ARC element radius is less than or equal to zero. To correct this problem, set the arc radius to a value greater than zero.

MPIPathMessagePATH_TOO_LONG

The path length is not valid. This message code is returned by mpiPathMotionParamsGenerate(…) if the path length is greater than MAX_PATH_POINTS. To correct the problem, specify a path with fewer points than MAX_PATH_POINTS.

MPIPathMessageILLEGAL_VELOCITY

The path element velocity is not valid. This message code is returned by mpiPathAppend(…) if the specified velocity is less than or equal to zero. To correct this problem, set the element velocity to a value greater than zero.

MPIPathMessageILLEGAL_ACCELERATION

The path element velocity is not valid. This message code is returned by mpiPathAppend(…) if the specified velocity is less than or equal to zero. To correct this problem, set the element velocity to a value greater than zero.

MPIPathMessageILLEGAL_TIMESLICE

The path element time slice is not valid. This message code is returned by mpiPathAppend(…) if the specified time slice is less than or equal to zero. To correct this problem, set the element time slice to a value greater than zero.

MPIPathMessageINVALID_BLENDING

The path element blending is not valid. This message code is returned by mpiPathMotionParamsGenerate(…) if the element blending is set to TRUE and the motion type does not support blending. To correct this problem, either set the element blending to FALSE or select a different motion type.

See Also

MPIPathParams

MPIPathParams

```
typedef struct MPIPathParams {
    long              dimension;
    MPIPathPoint      start;
    double            velocity;
    double            acceleration;
    double            deceleration;
    MPIMotionType     interpolation;
    double            timeSlice;
    double            conversion
                      [MPIPathPointDIMENSION_MAX][MPIPathPointDIMENSION_MAX];
} MPIPathParams;
```

Description

dimension	This value defines the number of axes to coordinate. Please see MPIPathPoint data type documentation for more information.
start	This structure defines the initial point for the path.
velocity	This value defines the speed along the path. The units are in counts per second.
acceleration	This value defines the rate of change of speed to reach the velocity along the path. The units are in counts per second * second.
deceleration	This value defines the rate of change of speed to reach zero velocity along the path. The units are in counts per second * second.
interpolation	This value specifies the motion algorithm to generate the path. Please see MPIMotionType data type documentation for more information.
conversion	This value is an *N* x *N* matrix (where *N* is the number of dimensions in the path motion) that scales and rotates the axes used in the path motion. This is useful when using two axes with different resolutions for each axis. **For two axes with different resolution:** Set conversion[0][0] to (desired x resolution / actual x resolution) Set conversion [1][1] to (desired y resolution / actual y resolution) Set conversion[0][1] and conversion[1][0] = 0 **For a coordinate rotation, where alpha is the rotation of the coordinate system:** Set conversion[0][0] and conversion [1][1] = cos(alpha) Set conversion[0][1] = sin(alpha) Set conversion[1][0] = -sin(alpha)

See Also

mpiPathParamsGet | mpiPathParmsSet | mpiPathMotionParamsGenerate | MPIPathPointDIMENSION_MAX

MPIPathPoint

MPIPathPoint

```
typedef struct MPIPathPoint {
    double      position[MPIPathPointDIMENSION_MAX];
} MPIPathPoint;
```

Description

position	This array defines the axis command positions for a path point. There must be one position value for each dimension.

See Also

MPIPathParams | mpiPathParamsGet | mpiPathParmsSet | mpiPathPointDIMENSION_MAX

mpiPathElementTYPE

mpiPathElementTYPE

```
#define mpiPathElementTYPE(type) ((type) & MPIPathElementTypeMASK)
```

Description **PathElementTYPE** is a macro that masks off all other bits in type, leaving the path element type.

See Also MPIPathElementType

mpiPathElementAttrMaskBIT

mpiPathElementAttrMaskBIT

```
#define mpiPathElementAttrMaskBIT(attr) (0x1 << (attr))
```

Description

PathElementAttrMaskBIT is a macro that converts the path element attribute into the path element attribute mask.

See Also

MPIPathElementAttrs | MPIPathElementAttrMask

mpiPathElementATTR

mpiPathElementATTR

```
        #define mpiPathElementATTR(type,attr)
                                  ((type) |= mpiPathElementAttrMaskBIT(attr))
```

Description **PathElementATTR** is a macro that turns on the specified path element attribute mask bits in the path element type.

See Also MPIPathAttr | MPIPathElementAttrMask

MPIPathPointDIMENSION_MAX

MPIPathPointDIMENSION_MAX

```
#define MPIPathPointDIMENSION_MAX      (16)
```

Description **PathPointDIMENSION_MAX** defines the maximum dimensions for path objects.

See Also MPIPathParams | MPIPathPoint