

# MX Robot SPEL function documentation

---

## SPEL Modules

- Quick Reference
- JSON String Format

## Initialization Modules

- GTInitialization
- GTSetupPoints
- GTSetupCassette

## Generic Modules

- GTCassetteUtilities
- GTCassetteType

- GTCassetteProbing
- GTSampleMounting

## Cassette Specific Modules

- GTNormalCassetteUtilities
- GTSuperPuckUtilities

## Dumbbell Motion Modules

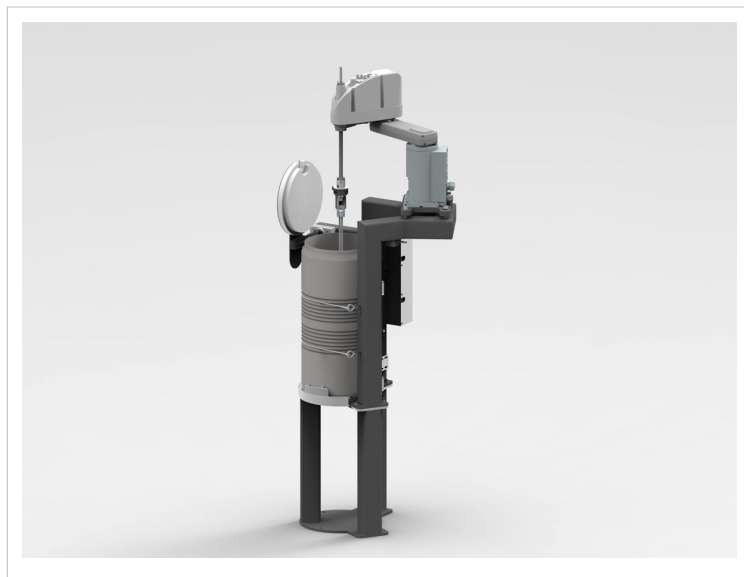
- GTMagnetManeuvers
- GTRobotSpeed

## Math Modules

- GTNumericManipulations

## Interface Modules

- GTPythonInterface
- GTJSONUtilities



# QuickReference

---

## Probe Functions

To run the probing mechanism, start the robot from P0 or from any position inside the dewar (except dumbbell inside the cassette ports), then call any of the following functions

- Call **ProbeCassettes** - with **g\_PortsRequestString\$(cassette\_position) = "1111100...111000"**
  - 1 = Probe the port
  - 0 = donot Probe the port
- Call **GTMountSamplePort** - with **g\_RunArgs\$ = "[LMR] [ABCDEFGHJKLM] [1-16] "**
  - i.e. **g\_RunArgs\$ = <cassette\_position> <column/puck Name> <row/ puckPort Index>**
- Call **JSONDataRequest** - with **g\_RunArgs\$ = "[LMR]{1,3} [APDCS]{1,5}"**
  - L = Left Cassette, M = Middle Cassette, R = Right Cassette
  - A = PUCK\_STATES, P = PORTS\_STATES, D = SAMPLE\_DISTANCES, C = CASSETTE\_TYPE, S = SAMPLE\_STATE

## Include files

**GTcassettedefs.inc** contains the Properties of Normal and Calibration Cassettes

**GTsuperpuckdefs.inc** contains the Properties of Super Puck Adaptor Cassettes

**GTgenericdefs.inc** contains the cassette positions, cassette types, toolset names, sample port properties, shrink factor and probing thresholds

## Global Variables

To identify the **Cassette Type** at a position,

Global Variable	Parameters	Value Contained
g_CassetteType(cassette_position)	cassette_position: 0 = Left, 1 = Middle, 2 = Right	0 = unknown 1 = calibration cassette, 2 = cassette, 3 = adaptor

```
Print Str$(g_CassetteType(0)) ' ' left position
```

**For SuperPuck Adaptor**

Global Variable	Parameters	Value Contained
<code>g_PuckStatus(cassette_position, puckIndex)</code>	cassette_position: 0 = Left, 1 = Middle, 2 = Right	-1 = PUCK_PRESENT 0 = PUCK_UNKNOWN 1 = PUCK_ABSENT 2 = PUCK_JAM
<code>g_SP_SampleDistanceError(cassette_position, puckIndex, portIndex)</code>	puckIndex: 0 = A, 1 = B, 2 = C, 3 = D,	Real value
<code>g_SP_PortStatus(cassette_position, puckIndex, portIndex)</code>	portIndex 0-15 = 1-16	-1 = Present 0 = Unknown 1 = Empty 2 = Error

```
Print Str$(g_PuckStatus(2,3)) ' ' right, D    position, puckIndex
Print Str$(g_SP_SampleDistanceError(2,1,0)) ' ' right, B, 1 position, puck index, port index
Print Str$(g_SP_PortStatus(2, 1, 0))
```

### For Normal/Calibration cassettes

Global Variable	Parameters	Value Contained
	cassette_position: 0 = Left, 1 = Middle, 2 = Right	
<code>g_CASSampleDistanceError(cassette_position, rowIndex, columnIndex)</code>	rowIndex: 0-7 = 1-8	Real value
<code>g_CAS_PortStatus(cassette_position, rowIndex, columnIndex)</code>	columnIndex 0-11 = A-L	-1 = Present 0 = Unknown 1 = Empty 2 = Error

```
Print Str$(g_SP_SampleDistanceError(2,1,0)) ' ' right, B, 1 position, puck index, port index
Print Str$(g_CAS_PortStatus(2, 1, 0)) 'Value of -1 to 2
```

### For Sample Properties

Global Variable	Value Contained
g_InterestedCassettePosition	0 = Left, 1 = Middle, 2 = Right
g_InterestedPuckColumnIndex	0-11 = A-L
g_InterestedRowPuckPortIndex	0-15 = 1-16
g_InterestedSampleStatus	0 = SAMPLE_STATUS_UNKNOWN 1 = SAMPLE_IN_CASSETTE 2 = SAMPLE_IN_PICKER 3 = SAMPLE_IN_PLACER 4 = SAMPLE_IN_CRADLE 5 = SAMPLE_IN_CAVITY 6 = SAMPLE_IN_GONIO

```
Print Str$(g_InterestedCassettePosition) ' ' Value of 0 to 2
Print Str$(g_InterestedPuckColumnIndex) ' 'Value of 0 to 11
Print Str$(g_InterestedRowPuckPortIndex) ' ' Value of 0 to 15
Print Str$(g_InterestedSampleStatus) ' 'Value of 0 to 6
```

# GTJSONstringFormat

## Value Definitions

The following tables are noted for parsing port\_status and puck\_status from the JSON strings:

### PortStatusString\$

PortStatus	Value
PORT_OCCUPIED	-1
PORT_VACANT	1
PORT_ERROR	2
None of the above (PORT_UNKNOWN)	0

### PuckStatusString\$

PuckStatus	Value
PUCK_PRESENT	-1
PUCK_ABSENT	1
PUCK_JAM	2
None of the above (PUCK_UNKNOWN)	0

## Probing

### Cassette Type

```
{'set':'cassette_type', 'position':'left', 'min_height_error':0.123, 'value':'calibration'}
```

### Cassette Port Properties

```
{'set':'port_states', 'position':'middle', 'start':6, 'value':-1}  
{'set':'sample_distances', 'position':'middle', 'start':6, 'value':0.123}
```

### Comprehensive Data Packet

```
{'set':'port_states','position':'middle','type':'normal','start':0,'value':[-1,1,...,2,0,]}  
{'set':'sample_distances','position':'middle','type':'normal','start':0,'value':[-0.12,1.45,...,0.78,0.00,]}
```

### SuperPuck Port Properties

```
{'set':'puck_states', 'position':'right', 'start':0, 'value':-1}  
{'set':'port_states', 'position':'right', 'start':16,'value':1}  
{'set':'sample_distances', 'position':'right', 'start':16,'value':0.123}
```

### Comprehensive Data Packet

```
{'set':'puck_states','position':'right','type':'superpuck','value':[-1,2,1,0,]}  
{'set':'ports_states','position':'right','type':'superpuck','start':0,'value':[-1,1,...,2,0,]}  
{'set':'sample_distances','position':'right','type':'superpuck','start':0,'value':[-0.12,1.45,...,0.78,0.00,]}
```

## Mounting

Sample Status	Value
SAMPLE_STATUS_UNKNOWN	0
SAMPLE_IN_CASSETTE	1
SAMPLE_IN_PICKER	2
SAMPLE_IN_PLACER	3
SAMPLE_IN_CRADLE	4
SAMPLE_IN_CAVITY	5
SAMPLE_IN_GONIO	6

```
{'set':'sample_state', 'position':'left', 'start':0, 'value':2}
```

# GTSetupPoints

---

## Description

Contains functions for checking points and toolsets and to derive basic points, magnet points, cassette points and goniometer points from the points obtained from executing the calibration process.

## GTCheckPoint

GTCheckPoint takes 1 Pointnum as Integer parameter and no output parameters

If the Point is not defined or if its X or Y co-ordinate is set to zero, GTCheckPoint returns false

Otherwise GTCheckPoint returns true

## GTCheckTool

GTCheckTool takes 1 Toolnum as Integer parameter and no output parameters

If the Tool's X or Y co-ordinate is set to zero, GTCheckTool returns false

Otherwise GTCheckTool returns true

## GTInitBasicPoints

Calls GTCheckPoint to check the following points:

- P0
- P1
- P18

If GTCheckPoint returns True for all these points, then GTInitBasicPoints returns True

Otherwise GTInitBasicPoints returns false

## GTInitMagnetPoints

Calls GTCheckPoint to check the following points:

- P6
- P16
- P26
- P10
- P11
- P12

And calls GTCheckTool to check the following Tools:

- Tool(1) - picker Toolset
- Tool(2) - placer Toolset

GTInitMagnetPoints exits by returning false if any of the above checks fail.

Then GTInitMagnetPoints sets the global variables, g\_dumbbell\_Perfect\_Angle, g\_dumbbell\_Perfect\_cosValue, and g\_dumbbell\_Perfect\_sinValue based from the point P6

Then GTInitMagnetPoints derives the following points in order

- P3
  - P2
-

- P4
- P17
- P27
- P93
- P94
- P15
- P25
- P5

## GTInitCassettePoints

Calls GTCheckPoint to check the following points

- P6
- Left Cassette Points - P34, P41, P44
- Middle Cassette Points - P35, P42, P45
- Right Cassette Points - P36, P43, P46

GTInitCassettePoints exits by returning false if any of the above checks fail.

Then calls GTSetupCassetteAllProperties for left, middle and right cassette. If it fails then GTInitCassettePoints exits by returning false.

## GTInitGoniometerPoints

Calls GTCheckPoint to check P20 and if it is a valid point, the function then extracts the g\_goniometer\_Angle variable from P20

## GTInitAllPoints

This function calls the above functions in the following order

1. GTInitBasicPoints
2. GTInitMagnetPoints
3. GTInitCassettePoints
4. GTInitGoniometerPoints

It returns true if all the functions called complete successfully otherwise it returns false.

## Global Variables

*Note: All angle values in GT domain are in degrees*

- Global Real **g\_dumbbell\_Perfect\_Angle**
  - Global Real **g\_dumbbell\_Perfect\_cosValue**
  - Global Real **g\_dumbbell\_Perfect\_sinValue**
  - Global Real **g\_goniometer\_Angle**
  - Global Real **g\_goniometer\_cosValue**
  - Global Real **g\_goniometer\_sinValue**
-

# GTSetupCassette

---

## Description

Contains functions for setting up actual position, actual orientation angle, and tilt information for each cassette. The following functions are defined in this module:

- GTSetupDirection(cassette\_position As Integer, column\_A\_Angle As Real, standbyU As Real, secondaryStandbyU As Real)
- GTSetupCoordinates(cassette\_position As Integer, pointNum As Integer)
- GTSetupTilt(cassette\_position As Integer, topPointNum As Integer, bottomPointNum As Integer) As Boolean
- GTSetupCassetteAllProperties(cassette\_position As Integer) As Boolean

## Global Variables

The following global variables (arrays) are set by the functions in GTSetupPoints

Angle of First Column in Cassette and U value for each cassette

- Global Real **g\_AngleOfFirstColumn(NUM\_CASSETTES)**
- Global Real **g\_UForNormalStandby(NUM\_CASSETTES)**
- Global Real **g\_UForSecondaryStandby(NUM\_CASSETTES)**
- Global Real **g\_UForPuckStandby(NUM\_CASSETTES)**

Tilt Information for Each Cassette

- Global Real **g\_tiltDX(NUM\_CASSETTES)**
- Global Real **g\_tiltDY(NUM\_CASSETTES)**

*Cassette Bottom Center's X,Y, Z Coordinates*

- Global Real **g\_CenterX(NUM\_CASSETTES)**
- Global Real **g\_CenterY(NUM\_CASSETTES)**
- Global Real **g\_BottomZ(NUM\_CASSETTES)**

*Angle offset of cassette from theoretical cassette orientation angle*

- Global Real **g\_AngleOffset(NUM\_CASSETTES)**
-



# GTCassetteUtilities

---

## Description

Contains functions for setting tilt offsets, getting a point on the circumference centered around the cassette center based on u and for probing individual ports on calibration and normal cassette. The following functions are defined in this module:

- `GTParseCassettePosition(cassetteChar$ As String, ByRef cassette_position As Integer) As Boolean`
- `GTCassetteName$(cassette_position As Integer) As String`
- `GTApplyTiltToOffsets(cassette_position As Integer, PerfectXoffset As Real, PerfectYoffset As Real, PerfectZoffset As Real, ByRef Actualoffsets() As Real)`
- `GTSetCircumferencePointFromU(cassette_position As Integer, U As Real, radius As Real, ZoffsetFromBottom As Real, pointNum As Integer)`
- `GTadjustUAngle(cassette_position As Integer, UAngle As Real) As Real`

# GTCassetteType

---

## Description

Contains functions for finding the cassette top, function for classifying the cassette type based on height, and to find the average height of the cassette from different points on the cassette, and the helper functions to setup the standby point for the before mentioned functions. The following functions are defined in this module:

- `GTCassetteType$(cassetteType As Integer) As String`
- `GTResetCassette(cassette_position As Integer)`
- `GTSetScanCassetteTopStandbyPoint(cassette_position As Integer, pointNum As Integer, uOffset As Real, ByRef scanZdistance As Real)`
- `GTScanCassetteTop(standbyPointNum As Integer, maxZdistanceToScan As Real, ByRef cassetteTopZvalue As Real) As Boolean`
- `GTCassetteTypeFromHeight(cassette_position As Integer, cassetteHeight As Real, ByRef guessedCassetteType As Integer, ByRef min_height_error As Real) As Boolean`
- `GTsetfindAvgHeightStandbyPoint(cassette_position As Integer, pointNum As Integer, index As Integer, guessedCassetteType As Integer, ByRef scanZdistance As Real)`
- `GTfindAverageCassetteHeight(cassette_position As Integer, cassetteFirstHeight As Real, guessedCassetteType As Integer, ByRef average_height As Real) As Boolean`

## Global Variables

The following global variables (arrays) are set by the functions in GTCassetteType

- Global Preserve Integer `g_CassetteType(NUM_CASSETTES)`
-

# GTCassetteProbing

---

## Description

Contains functions for probing a cassette to classify its type, function to probe all the sample ports in the cassette and to reset the probed information to absent. The following functions are defined in this module:

- **GTProbeCassetteType**(cassette\_position As Integer) - Probes the cassette\_position for detecting whether a cassette is present or absent. If a cassette is present, it identifies what type of cassette is present.
- **GTProbeSpecificPorts**(cassette\_position As Integer) - Only after GTProbeCassetteType is successfully completed, this function should be called to run to the probe the specific sample ports in the cassette as requested in g\_PortsRequestString\$(cassette\_position).
- **GTResetSpecificPorts**(cassette\_position As Integer) - This function resets the status of specific pucks and the specific ports to unknown in the corresponding global variables as requested in g\_PortsRequestString\$(cassette\_position).

# GTSampleMounting

---

## Description

Contains functions for finding whether the Goniometer can be reached, function for deriving the goniometer points from P20, and to set the ports of interest and mount these interesting ports. The following functions are defined in this module:

- GTGonioReachable() As Boolean
- GTSetGoniometerPoints(dx As Real, dy As Real, dz As Real, du As Real) As Boolean
- GTsetInterestPoint(cassette\_position As Integer, puckColumnIndex As Integer, rowPuckPortIndex As Integer) As Boolean
- GTMoveToSPMountPortStandbyPoint(cassette\_position As Integer, puckIndex As Integer, puckPortIndex As Integer)
- GTMoveToCASMOUNTPortStandbyPoint(cassette\_position As Integer, rowIndex As Integer, columnIndex As Integer)
- GTMountInterestedPort

## Global Variables

The following global variables are set by the functions in GTSampleMounting

- Global Preserve Integer **g\_InterestedCassettePosition**
  - Global Preserve Integer **g\_InterestedPuckColumnIndex**
  - Global Preserve Integer **g\_InterestedRowPuckPortIndex**
  - Global Preserve Integer **g\_InterestedSampleStatus**
-

# GTNormalCassetteUtilities

---

## Description

Contains functions for setting tilt offsets, getting a point on the circumference centered around the cassette center based on u and for probing individual ports on calibration and normal cassette. The following functions are defined in this module:

- `GTParseColumnIndex(columnChar$ As String, ByRef columnIndex As Integer) As Boolean`
- `GTcolumnName$(columnIndex As Integer)`
- `GTprobeCassettePort(cassette_position As Integer, rowIndex As Integer, columnIndex As Integer, jumpToStandbyPoint As Boolean)`
- `GTProbeSpecificPortsInCassette(cassette_position As Integer) As Boolean`
- `GTResetSpecificPortsInCassette(cassette_position As Integer)`
- `GTsetCassetteMountStandbyPoints(cassette_position As Integer, rowIndex As Integer, columnIndex As Integer, standbyDistanceFromCASSurface As Real)`

## Global Variables

The following global variables (arrays) are set by the functions in GTCassetteUtilities

- Global Preserve Real `g_CASSampleDistanceError(NUM_CASSETTES, NUM_ROWS, NUM_COLUMNS)`
- Global Preserve Integer `g_CAS_PortStatus(NUM_CASSETTES, NUM_ROWS, NUM_COLUMNS)`

# GTSuperPuckUtilities

---

## Description

Contains functions for probing super puck adaptor for angle correction, and to setup the correction angle, probing the presence of pucks and individual ports, and the helper functions to setup perfect point, standby and destination points for the before mentioned functions.

## Global Variables

The following global variables (arrays) are set by the functions in GTSuperPuckUtilities

- `g_PuckPresent(NUM_CASSETTES, NUM_PUCKS)`
  - `g_SampleDistancefromPuckSurface(NUM_CASSETTES, NUM_PUCKS, NUM_PUCK_PORTS)`
  - `g_SP_PortStatus(NUM_CASSETTES, NUM_PUCKS, NUM_PUCK_PORTS)`
-

## Function Details

The following functions are defined in this module:

### initSuperPuckConstants

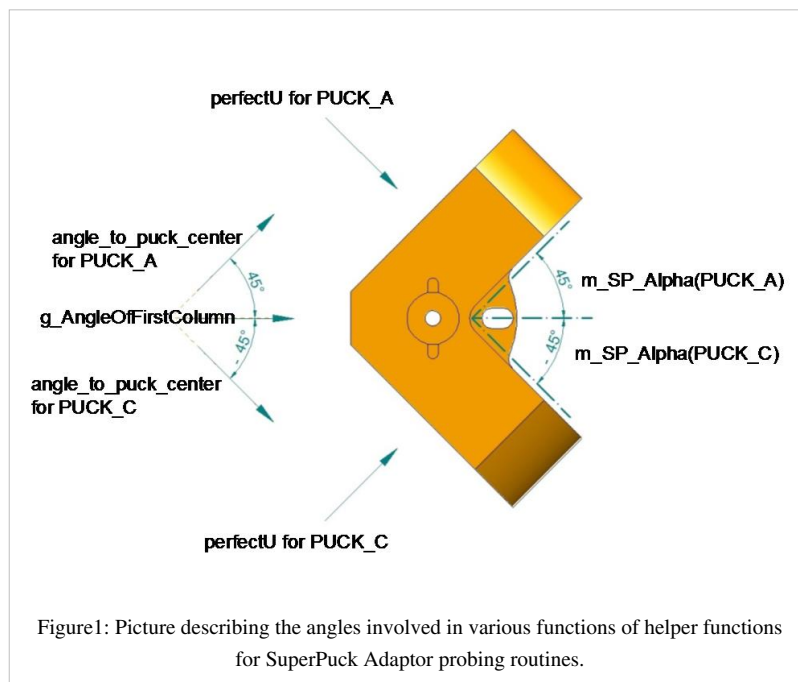
initSuperPuckConstants() sets the values for the constants which are declared as module variables such as

1. **m\_SP\_Alpha** - Angle between the line pointing the center of the Superpuck Adaptor and the rear face of the Puck
2. **m\_SP\_Puck\_Radius** - Radius of each puck
3. **m\_SP\_Puck\_Thickness** - Thickness of each puck
4. **m\_SP\_PuckCenter\_Height** - Vertical distance between the puck center and the bottom of the SuperPuck Adaptor
5. **m\_SP\_Puck\_RotationAngle** - The rotation of each puck with respect to PUCK\_A
6. **m\_SP\_Ports\_1\_5\_Circle\_Radius** - Radius of the circle formed by the ports 1 to 5 in each puck
7. **m\_SP\_Ports\_6\_16\_Circle\_Radius** - Radius of the circle formed by the ports 6 to 16 in each puck

### GTpuckName\$

GTpuckName\$(puckIndex As Integer) As String returns a string representing the human readable Puck Name according to the following table

puckIndex	returned String\$
0	PUCK_A
1	PUCK_B
2	PUCK_C
3	PUCK_D



### GTParsePuckIndex

GTParsePuckIndex(puckChar\$ As String, ByRef puckIndex As Integer) As Boolean sets the corresponding puckIndex and returns True if puckChar\$ is valid otherwise returns false

### GTSPpositioningMove

GTSPpositioningMove(cassette\_position As Integer, puckIndex As Integer) As Boolean

Before Adaptor Angle Correction probing mechanism is invoked, this function should be called to push the puck Adaptor once at the edge along the vertical center line of the Super

Puck Adaptor. This is done to alleviate the errors in Adaptor Angle Correction probing due to improper orientation of the puck adaptor.

The angle variables used in this function are depicted in Figure1: MX\_Robot\_SuperPuck\_Angles\_Info.jpg

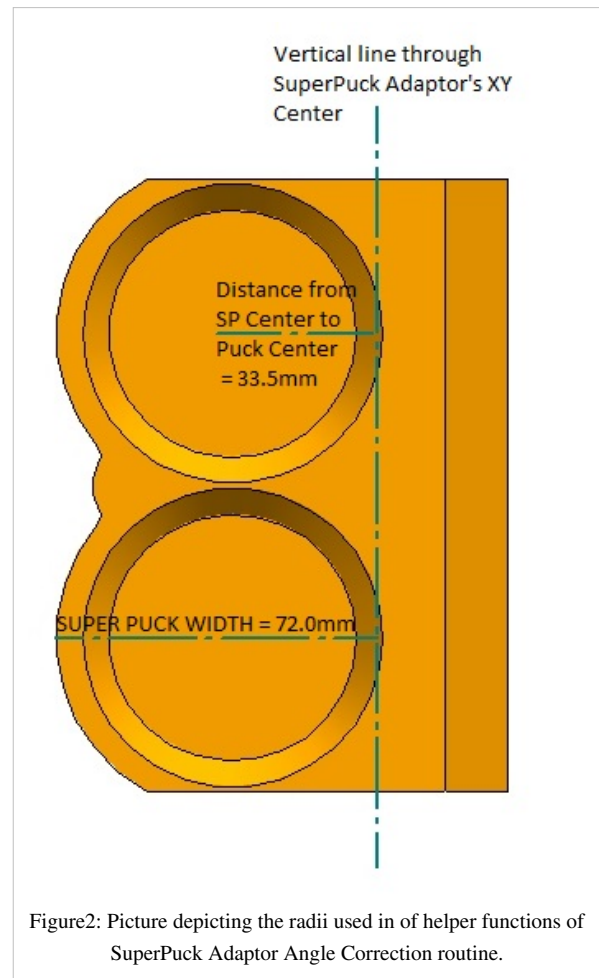
### GTgetAdaptorAngleErrorProbePoint

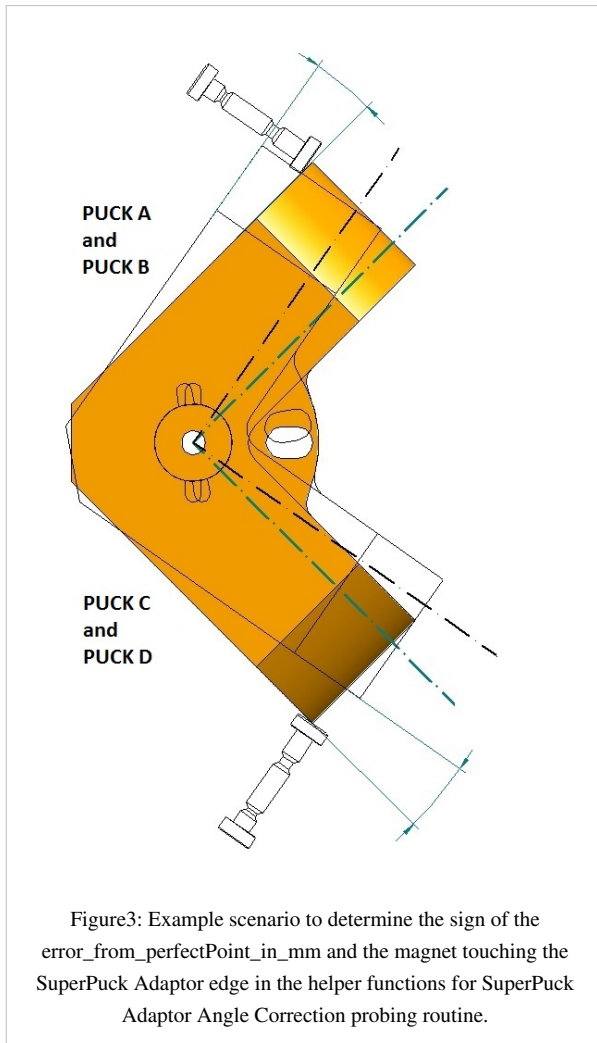
GTgetAdaptorAngleErrorProbePoint(cassette\_position As Integer, puckIndex As Integer, perfectPointNum As Integer, standbyPointNum As Integer, destinationPointNum As Integer) is a helper function to set the perfect, standby and destination points for the SuperPuck Adaptor Angle Correction

Here variables storing angles are as shown in Figure1

The probe for angle correction is done at the intersection of the SuperPuck Edge and the line joining the puckCenter and Port 11 for PUCK\_A and PUCK\_B whereas

The probe for angle correction is done at the intersection of the SuperPuck Edge and the line joining the puckCenter and Port 10 for PUCK\_C and PUCK\_D due to the rotation angle of 180 for pucks C and D.





### GTprobeAdaptorAngleCorrection

GTprobeAdaptorAngleCorrection(cassette\_position As Integer, puckIndex As Integer) As Boolean This function calls the GTSPpositioningMove and GTgetAdaptorAngleErrorProbePoint. Then starts from the standby point and move towards the destination point. Once ForceTouch exceeds the threshold, the distance to the magnet's center is measured from the perfectPoint.

Look at Figure3, for an example scenario. In this figure the actual position is solid and the perfect position is transparent. Now in Figure 3 for PUCK\_A and PUCK\_B, the cassette is detected only after moving beyond the perfectPosition, this is considered positive (+) movement. But, for PUCK\_C and PUCK\_D, the cassette is detected before even reaching the perfectPosition, this is considered negative(-) movement. So the signs of error\_from\_perfectPoint\_in\_mm is set accordingly.

### GTsetupAdaptorAngleCorrection

GTsetupAdaptorAngleCorrection(cassette\_position As Integer, puckIndex As Integer, error\_from\_perfectPoint\_in\_mm As Real) As Boolean In the example shown in Figure3, when we look closely, we see that when the cassette is detected only after moving beyond the perfectPosition, such as for PUCK\_A, we can observe that the magnet edge is touching the cassette at

the moment when ForceTouch exceeds the threshold. But, when the cassette is detected before reaching the perfectPosition, such as for PUCK\_C, we can observe that the magnet center is touching the cassette at the moment when ForceTouch exceeds the threshold. Hence, this is taken into account when calculating the angle error.

To view the angle errors from global coordinates and to follow easy conventions, sign of all angle errors are considered with respect to PUCK\_A's angleError.

## GTperfectSPPortOffset

GTperfectSPPortOffset(cassette\_position As Integer, portIndex As Integer, puckIndex As Integer, distanceFromPuckSurface As Real, ByRef dx As Real, ByRef dy As Real, ByRef dz As Real, ByRef u As Real) sets **dx**, **dy**, and **dz** with the distances that have to be travelled to reach the port from the super puck adaptor center. **u** is set to the u angle for probing the puck.

The calculations involve calculating the puckCenter and then the Horizontal and Vertical distances from puck center to port center and then the distance from port's deepest end.

- Figure1 is used as reference for the angles involved.
- Figure4 is used as reference to the Horizontal and Vertical distances from puck center to port center for Port 7.
- For calculating the distance from port's deepest end, the parameter **distanceFromPuckSurface** is used with the following rule:
  1. **distanceFromPuckSurface** > 0 is the offset away from the puck
  2. **distanceFromPuckSurface** < 0 is the offset into the puck (port)

## GTsetSPPortPoint

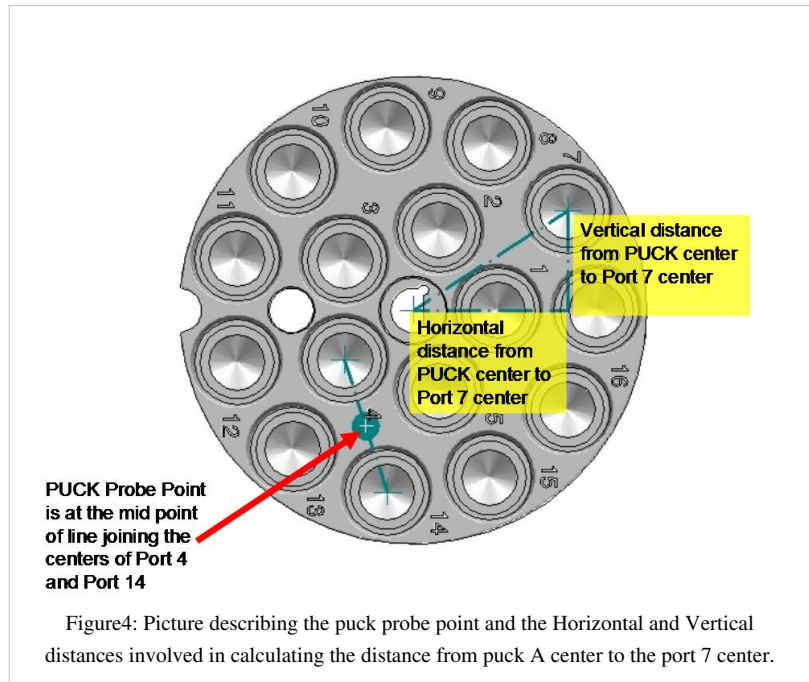
GTsetSPPortPoint(cassette\_position As Integer, portIndex As Integer, puckIndex As Integer, distanceFromPuckSurface As Real, pointNum As Integer)

## GTsetSPPuckProbeStandbyPoint

GTsetSPPuckProbeStandbyPoint(cassette\_position As Integer, puckIndex As Integer, standbyPointNum As Integer, ByRef scanDistance As Real)

## GTprobeSPPuck

GTprobeSPPuck(cassette\_position As Integer, puckIndex As Integer)



**GTprobeSPPort**

GTprobeSPPort(cassette\_position As Integer, puckIndex As Integer, portIndex As Integer)

**GTProbeSpecificPortsInSuperPuck**

Function GTProbeSpecificPortsInSuperPuck(cassette\_position As Integer) As Boolean

**GTResetSpecificPortsInSuperPuck**

Function GTResetSpecificPortsInSuperPuck(cassette\_position As Integer)

**GTsetSPMountStandbyPoints**

Function GTsetSPMountStandbyPoints(cassette\_position As Integer, puckIndex As Integer, puckPortIndex As Integer, standbyDistanceFromSPSurface As Real)

## GTMagnetManeuvers

---

**Description**

Contains functions to jump from home to cooling point and wait, for checking whether the magnet is in the gripper, and to pick and return magnet from cradle. The following functions are defined in this module:

- GTJumpHomeToCoolingPointAndWait As Boolean
  - GTIsMagnetInGripper As Boolean
  - GTCheckAndPickMagnet As Boolean
  - GTReturnMagnet As Boolean
  - GTReturnMagnetAndGoHome As Boolean
  - GTTwistOffMagnet
-



# GTRobotSpeed

---

## Description

Contains functions for saving current robot speed mode, setting new robot speed mode and to load previous robot speed mode. The following functions are defined in this module:

- GTSaveCurrentRobotSpeedMode
- SetProbeSpeed
- SetInsideLN2Speed
- SetOutsideLN2Speed
- GTsetRobotSpeedMode(speed\_mode As Byte)
- GTLoadPreviousRobotSpeedMode

# GTNumericManipulations

---

## Description

Contains mathematical helper functions. The following functions are defined in this module:

- GTBoundAngle(lowerBound As Real, upperBound As Real, Angle As Real) As Real
- GTAngleToPerfectOrientationAngle(Angle As Real) As Real

# GTJSONUtilities

---

## Description

Contains functions setting up JSON strings and send it to the client using UpdateClient(CLIENT\_UPDATE, JSONString\$, INFO\_LEVEL) call. The following dataToSend argument (whereever required) can take the following (defined in jsondefs.inc) values

1. define SAMPLE\_PORT\_STATUS 0
2. define SAMPLE\_DISTANCE\_ERROR 1
3. define PUCK\_STATUS 2
4. define CASSETTE\_TYPE 3

## PortStatusString\$

PortStatusString\$(PortStatus As Integer) As String returns a corresponding string based on the below format

---

PortStatus	PortStatusString\$
PORT_OCCUPIED	'OCC'
PORT_VACANT	'VAC'
PORT_ERROR	'ERR'
None of the above (PORT_UNKNOWN)	'UNK'

## PuckStatusString\$

- PuckStatusString\$(PuckStatus As Integer) As String returns a corresponding string based on the below format. (To maintain consistency with Port status, similar notations are used)

PuckStatus	PuckStatusString\$
PUCK_PRESENT	'OCC'
PUCK_ABSENT	'VAC'
PUCK_JAM	'ERR'
None of the above (PUCK_UNKNOWN)	'UNK'

## GTsendNormalCassetteData

GTsendNormalCassetteData(dataToSend As Integer, cassette\_position As Integer) packs and sends JSON formatted strings of a Normal or Calibration Cassette located at cassette\_position based on the supplied dataToSend Argument as mentioned above. If any of the arguments are invalid, the function just exits without sending anything.

```
{'set':'cassette_ports_status','position':'middle','type':'normal','start':0,'end':23,'value':['OCC','VAC',...,'ERR','UNK',]}
{'set':'cassette_distance_errors','position':'middle','type':'normal','start':0,'end':23,'value':[-0.12,1.45,...,0.78,0.00]}
```

## GTsendSuperPuckData

GTsendSuperPuckData(dataToSend As Integer, cassette\_position As Integer) packs and sends JSON formatted strings of a SuperPuck Cassette located at cassette\_position based on the supplied dataToSend Argument as mentioned above. If any of the arguments are invalid, the function just exits without sending anything.

```
{'set':'cassette_ports_status','position':'right','type':'superpuck','start':0,'end':15,'value':['OCC','VAC',...,'ERR','UNK',]}
{'set':'cassette_distance_errors','position':'right','type':'superpuck','start':0,'end':15,'value':[-0.12,1.45,...,0.78,0.00]}
```

## GTsendPuckData

GTsendPuckData(cassette\_position As Integer) packs and sends the Puck Presence information in JSON formatted strings of a SuperPuck Cassette located at cassette\_position. If cassette\_position is invalid, the function just exits without sending anything.

```
{'set':'cassette_pucks_status','position':'right','type':'superpuck','value':['OCC','VAC','ERR','UNK',]}
```

## GTsendCassetteType

GTsendCassetteType(cassette\_position As Integer) sends the cassette type in JSON formatted strings for a cassette located at cassette\_position.

```
{'set':'cassette_type','position':'left','min_height_error':0.123,'value':'calibration'}
```

**GTsendCassetteData**

GTsendCassetteData(dataToSend As Integer, cassette\_position As Integer) packs and sends JSON formatted strings for any type of cassette located at cassette\_position based on the supplied dataToSend Argument as mentioned above. If any of the arguments are invalid, the function just exits without sending anything.

# Article Sources and Contributors

**MX Robot SPEL function documentation** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55330> *Contributors:* Gautam, RobbieClarcken

**QuickReference** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55391> *Contributors:* Gautam, RobbieClarcken

**GTJSONStringFormat** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55389> *Contributors:* Gautam

**GTSetupPoints** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55331> *Contributors:* Gautam

**GTSetupCassette** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55332> *Contributors:* Gautam

**UTCassetteUtilities** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55336> *Contributors:* Gautam

**UTCassetteType** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55334> *Contributors:* Gautam

**UTCassetteProbing** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55335> *Contributors:* Gautam

**GTSampleMounting** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55337> *Contributors:* Gautam

**GTNormalCassetteUtilities** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55339> *Contributors:* Gautam

**GTSuperPuckUtilities** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55338> *Contributors:* Gautam

**GTMagnetManeuvers** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55341> *Contributors:* Gautam

**GTRobotSpeed** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55342> *Contributors:* -

**GTNumericManipulations** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55343> *Contributors:* Gautam

**GTJSONUtilities** *Source:* <http://wiki.synchrotron.org.au/index.php?oldid=55369> *Contributors:* Gautam

# Image Sources, Licenses and Contributors

**File:MX\_Robot\_Assembly.jpg** *Source:* [http://wiki.synchrotron.org.au/index.php?title=File:MX\\_Robot\\_Assembly.jpg](http://wiki.synchrotron.org.au/index.php?title=File:MX_Robot_Assembly.jpg) *License:* unknown *Contributors:* -

**File:MX Robot SuperPuck Angles Info.jpg** *Source:* [http://wiki.synchrotron.org.au/index.php?title=File:MX\\_Robot\\_SuperPuck\\_Angles\\_Info.jpg](http://wiki.synchrotron.org.au/index.php?title=File:MX_Robot_SuperPuck_Angles_Info.jpg) *License:* unknown *Contributors:* Gautam

**File:MX Robot Adaptor Angle Correction Info.jpg** *Source:* [http://wiki.synchrotron.org.au/index.php?title=File:MX\\_Robot\\_Adaptor\\_Angle\\_Correction\\_Info.jpg](http://wiki.synchrotron.org.au/index.php?title=File:MX_Robot_Adaptor_Angle_Correction_Info.jpg) *License:* unknown *Contributors:* Gautam

**File:MX Robot AdaptorAngleCorrection Signs.jpg** *Source:* [http://wiki.synchrotron.org.au/index.php?title=File:MX\\_Robot\\_AdaptorAngleCorrection\\_Signs.jpg](http://wiki.synchrotron.org.au/index.php?title=File:MX_Robot_AdaptorAngleCorrection_Signs.jpg) *License:* unknown *Contributors:* Gautam

**File:MX Robot Puck Probe Points.jpg** *Source:* [http://wiki.synchrotron.org.au/index.php?title=File:MX\\_Robot\\_Puck\\_Probe\\_Points.jpg](http://wiki.synchrotron.org.au/index.php?title=File:MX_Robot_Puck_Probe_Points.jpg) *License:* unknown *Contributors:* Gautam