ACE Sight

Reference Guide

I609-E-01

Copyright Notice

Table Of Contents

Recent Changes

For the most recent change information, please see the ReadMe.rtf file, which can be viewed by selecting the following from the Windows Start button:

Start > Program > Omron> ACE > ACE ReadMe File

Compatibility Differences

This section describes the compatibility differences between AdeptSight 2.x and ACE Sight.

1. A standard collection of ACE Sight V+ programs are contained in a file called *asight.v2*. This file is updated on any controllers in the workspace when the ACE software connects to a given controller.

2. The asight.v2 file contains some of the sample code associated with AdeptSight 2.x queue management, namely:

getinstance	reset_seq	string2instance
nzs2string	set_as_exec_mod	

It is no longer necessary to include those programs in an application being migrated from AdeptSight 2 to ACE Sight.

3. ACE Sight operates in the “AdeptSight Server” mode of AdeptSight 2.x. This will not functionally change applications that use the communications tool queue to access parts.

4. Several of the properties have changed. In total, there are 440 command parameter codes in AdeptSight 2.x. The conversion to ACE Sight affects 17 of these commands. The changes are as follows:

Property Name	Code	Description of the Change
RobotConfiguration	10400	No longer supported.
InverseKinematics	10060	The configuration used in the kinematic calculation is based on the current configuration of the robot. It no longer uses the RobotConfiguration parameter. Inverse kinematics calculations are not supported for the Cobra i-Series robot.
GripperOffset	10100	ACE Sight returns the gripper offset from the gripper offset table associated with a specific robot. For details, see See "GripperOffset".

The following command codes are not directly supported in ACE Sight at this time. Please contact Adept Technical Support for programming options.

Property Name	Code
LoadProject	10300
LoadSequence	10301
LoadColorCalibration	10302
LoadVisionCalibration	10303
LoadRobotCalibration	10304

Property Name	Code
LoadBeltCalibration	10305
LoadCameraSettings	10306
SaveProject	10320
SaveSequence	10321
SaveColorCalibration	10322
SaveVisionCalibration	10323
SaveRobotCalibration	10324
SaveBeltCalibration	10325
SaveCameraSettings	10326

The following command codes are no longer supported.

Property Name	Code
ModelAutomaticLevels	410
ModelOutlineLevel	411
ModelDetailLevel	412
ModelContrastThresholdMode	413
ModelContrastThreshold	414
ModelTrackingInertia	415
ModelFeatureSelection	416
ModelBoundingAreaBottom	417
ModelBoundingAreaTop	418
ModelBoundingAreaLeft	419
ModelBoundingAreaRight	420
ModelOriginPositionX	421
ModelOriginPositionY	422
ModelOriginRotation	423
ModelReferencePointCount	424

Property Name	Code
ModelReferencePointPositionX	425
ModelReferencePointPositionY	426
ModelShadingAreaBottom	427
ModelShadingAreaTop	428
ModelShadingAreaLeft	429
ModelShadingAreaRight	430
ToolUseEntireImage	102
LearnTime	1302
Status	1002
MessageCount	1300
MessageNumber	1301
ModelDatabaseModified	402
ModelCount	404
ModelEnabled	405
CoordinateSystem	1000
OutputOutlineSceneEnabled	21
OutputDetailSceneEnabled	22
OutputInstanceSceneEnabled	23
OutputMode	24
ModelBasedScaleFactorMode	220
ModelBasedMinimumScaleFactor	221
ModelBasedMaximumScaleFactor	222
ModelBasedRotationMode	223
ModelBasedMinimumRotation	224
ModelBasedMaximumRotation	225
ModelEnabled	405

FrameTranslationX (2400), FrameTranslationY (2401), and FrameRotation (2402) should be replaced with the InstanceTranslationX (1315), InstanceTranslationY (1316), and InstanceRotation (1314). The Instance parameter has been expanded to support all tools.

5. Table-mounted camera refinement has changed. In AdeptSight 2.x, a table-mounted camera is treated as a special case, and the results are used to directly offset a target position for placement. In ACE Sight, the table-mounted camera is *not* treated as a special case. If you use a table-mounted camera, command code 1311 returns the world location of the located vision object.

In addition, the instruction InstanceToolOffset (command code 1372) has been added to return the location of the vision object relative to the gripper tip at the time the picture was taken. For details, see See "InstanceToolOffset".

6. Additional properties and corresponding command codes have been added to enable the reading of a robot gripper IO:

Property Name	Code
GripperOutputOpen	5511
GripperOutputClose	5512
GripperOutputRelease	5513
GripperInputOpen	5514
GripperInputClose	5515
GripperOutputExtend	5516
GripperOutputRetract	5517
GripperInputExtend	5518
GripperInputRetract	5519

NOTE: In version 3.1, the parameters for these were modified to the following:

$ip	IP address of the vision server
sequence	Not used. Must be -1.
Tool	The tip number
Index	The robot number
Object	The index of the signal number to access

7. Additional command codes have been added to enable the reading and writing of camera settings:

Property Name	Code
ActiveCalibration	5504
ActiveSettings	5505
VideoExposure	5502
VideoGain	5503

8. A property and corresponding command code has been added to enable the reading of a tool transformation for a given tip on the robot gripper:

Property Name	Code
GripperToolTransform	11000

NOTE: In version 3.1, the parameters were modified to the following:

$ip	IP address of the vision server.
sequence	Not used. Must be -1.
Tool	Index of the tip to access.
Instance	The robot number.
Result	Not used.
Frame	Not used.

9. A property and corresponding command code has been added to enable the reading of the total number of results that have been queued by all communication tools within a given sequence:

Property Name	Code
CommunicationToolResults	2600

10. The following vision tools were added for ACE Sight:

Calculated Arc	Calibration Grid Locator	Inspection Tool
Calculated Frame	Custom Vision Tool	Remote Vision Tool
Calculated Line	Gripper Clearance Tool	
Calculated Point	Image Sampling Tool	

11. ACE Sight can import project files created in AdeptSight 2. ACE Sight will attempt to extract all vision sequences and camera devices. There are some known limitations, which are described below.

• ACE Sight will not import the robot and controller device information.
AdeptSight 2 does not store enough information to automate the creation of these objects in the workspace

• ACE Sight will not properly import uncalibrated camera devices.
AdeptSight 2 does not use a consistent scheme when defining the origin of images associated with camera devices. In AdeptSight 2, the origin of an uncalibrated device is the top left corner of the image. The origin of a calibrated device is the center of the image. In ACE Sight, the coordinates used for all images are the center of the image. When an uncalibrated device is loaded, there is not enough information in the project file to properly translate the uncalibrated data from an AdeptSight 2 device and tool locations to the equivalent device and tool information in ACE Sight 3. In this case, the device and tools will be imported, but the user will need to retrain the tools and device settings.

• The ACE Sight Inspection Tool does not support the same set of operations as the AdeptSight 2 Results Inspection Tool.
An ACE Sight Inspection Tool will be created in place of an AdeptSight 2 Results Inspection Tool, but the filters are not imported.

12. The following properties and corresponding command codes were added to the Inspection tool in ACE Sight:

Property Name	Code
InspectionFilterMeasuredValue	2700
InspectionFilterNominalDeviation	2701
InspectionFilterPassStatus	2702

13. To support the Adept AnyFeeder bulk-part feeder, the ACE Sight V+ keywords VPARAMETER, VRUN, VSTATE, and VWAITI have been modified, and ACE Sight command codes 6000-6018 have been added. For details, see the *Adept AnyFeeder User's Guide*, Rev B or later.

CAUTION: If an ACE Sight vision sequence and an Adept AnyFeeder have the same index value, only the ACE Sight vision sequence will be executed. This prevents any conflict with an existing ACE Sight vision system when adding an Adept AnyFeeder.

14. The following properties and corresponding command codes were added to ACE Sight 3.2:

Property Name	Code
BeltCalibrationNearsideLimit	10003
InstanceVisionOffset	1373

15. The following property and corresponding command code was added to ACE Sight 3.4:

Property Name	Code
BeltLatchCalibrationOffset. For details, see See "BeltLatchCalibrationOffset ".	10010

ACE Sight V+ and MicroV+ Keywords

The following keywords are required for programming ACE Sight applications in MicroV+ or V+.

Click on a keyword to view the corresponding description.

VLOCATION transformation function

Syntax

MicroV+

```
VLOCATION (sequence_id, tool_id, instance_id, result_id, index_id, frame_id)
```

V+

```
VLOCATION ($ip, sequence_id, tool_id, instance_id, result_id, index_id, frame_id)
```

Description

Returns a Cartesian transform result of the execution of the specified vision sequence. The returned value is a transform result: x, y, z, yaw, pitch, roll.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance in the specified result frame. If no result frame is specified, it is the index for all instances returned by the tool.
result_id	Identifier (ID) of the result. Refer to the ACE Sight Properties Quick Reference tables to find the ID for the required result. Typically this value = 1311. For gripper offset location, this value can be set to 1400 and incremented by 1 for each additional gripper offset. The maximum value is 1499. See Example 2.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame for which you want to retrieve the result contained in the specified instance.

Details

The following parameters are optional: *sequence_id, tool_id*, *instance_id*, *index_id*, and *frame_id*. These parameters are 1-based. If no value is provided for these parameters, they default to 1.

For V+ systems, the vision server is the PC on which the ACE Sight vision software is running.

To retrieve specific values

To retrieve global values:	sequence_id = -1, tool_id = -1
To retrieve camera values:	sequence_id = -1, tool_id = cameraIndex
To retrieve camera-relative-to robot values:	sequence_id = -1, tool_id = cameraIndex, index_id = robotIndex
To retrieve sequence values:	sequence_id = sequenceIndex, tool_id = -1

To retrieve Belt Calibration-related values (read only)

Property	sequence_id	tool_id	instance_id	result_id	index_id	frame_id
Frame	-1	cameraIndex	n/a	10000	robotIndex	n/a
UpstreamLimit	-1	cameraIndex	n/a	10001	robotIndex	n/a
DownstreamLimit	-1	cameraIndex	n/a	10002	robotIndex	n/a
NearsideLimit	-1	cameraIndex	n/a	10003	robotIndex	n/a
VisionOrigin	-1	cameraIndex	n/a	10050	robotIndex	n/a

To retrieve Belt Latch Calibration offsets (read only)

Property	sequence_id	tool_id	instance_id	result_id	index_id	frame_id
Latch CalibrationOffset	-1	Reference number, as defined in Keyword Mapping parameter of ACE Sight Latch Calibration (in ACE workspace).	n/a	10010	robotIndex	n/a

Examples

Example 1

In this example, the *1311* result ID indicates using the first gripper offset. This is equivalent to using the *1400* result ID.

```
; Retrieve the location of a found instance
; instance location = 1311
SET location = VLOCATION($ip, 1, 2, 1, 1311)
```

Example 2

```
; set 1st gripper offset location
; 1st gripper offset location = 1400
SET location = VLOCATION ($ip, 1, 2, 1, 1400)
; set 2nd gripper offset location
SET location = VLOCATION ($ip, 1, 2, 1, 1401)
...
; set 6th gripper offset location
SET location = VLOCATION ($ip, 1, 2, 1, 1405)
```

Example 3

```
; Retrieve the location of the Belt frame
; BeltCalibrationFrame index is 10000
VLOCATION ($ip, -1, cameraIndex, , 10000, robotIndex)
; Retrieve the location of the Vision origin
; VisionOrigin index is 10050
VLOCATION ($ip, -1, cameraIndex, , 10050, robotIndex)
```

VPARAMETER program instruction

NOTE: This keyword can also be used to control the Adept AnyFeeder. For details, see the *Adept AnyFeeder User's Guide*, Rev B or later.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, parameter_id, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, parameter_id, index_id, object_id) $ip = value
```

Description

Sets the current value of a vision tool parameter.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
parameter_id	Identifier (ID) of the parameter. Refer to the ACE Sight Properties Quick Reference tables to find the ID for the required parameter.
index_id	Reserved for internal use. Value is always 1.
object_id	Some parameters require an object index to access specific values in an array. Please refer to the details for the individual parameter to understand the meaning and possible usage.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Details

The following parameters are optional: *sequence_id, tool_id*, *parameter_id*, *index_id*, and *object_id*. These parameters are 1-based. If no value is provided for these parameters, they default to 1.

For V+ systems, the vision server is the PC on which the ACE Sight vision software is running.

Example

```
; Set a Locator to find
; a maximum of 4 object instances.
; MaximumInstanceCount = 519
VPARAMETER(1, 2, 519) $ip = 4
```

VPARAMETER real-valued function

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, parameter_id, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, parameter_id, index_id, object_id)
```

Description

Gets the current value of a vision tool parameter.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
parameter_id	Identifier (ID) of the parameter. Refer to the ACE Sight Properties Quick Reference tables to find the ID for the required parameter.
index_id	Reserved for internal use. Value is always 1.
object_id	Some parameters require an object index to access specific values in an array. Please refer to the details for the individual parameter to understand the meaning and possible usage.

Details

The following parameters are optional: *sequence_id, tool_id*, *parameter_id*, *index_id*, and *object_id*. These parameters are 1-based. If no value is provided for these parameters, they default to 1.

To retrieve specific values

To retrieve global values: sequence_id = -1, tool_id = -1

To retrieve camera values: sequence_id = -1, tool_id = cameraIndex

To retrieve sequence values: sequence_id = sequenceIndex, tool_id = -1

To retrieve Belt-Calibration-related values (read only)

Scale (10004)	sequence_id = -1, tool_id = cameraIndex, index_id = robotIndex, object_id = n/a

To retrieve sequence-related values

Mode (10200)	sequence_id = sequenceIndex, tool_id = -1, index_id = n/a, object_id = n/a

Example

```
; Get the scale value for the Belt Calibration
scalevalue = VPARAMETER ($ip, -1, cameraIndex, 10004, robotIndex)
```

VRESULT real-valued function

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, result_id, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, result_id, index_id,
        frame_id)
```

Description

Returns a specified result of a vision tool, or returns the status of a specified tool.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance in the specified result frame. If no result frame is specified, it is the index for all instances returned by the tool.
result_id	Identifier (ID) of the result. Refer to the ACE Sight Properties Quick Reference tables to find the ID for the required result.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame for which you want to retrieve the result contained in the specified instance.

Details

The following parameters are optional: *sequence_id, tool_id*, *instance*, *index_id*, and *frame_id*. These parameters are 1-based. If no value is provided for these parameters, they default to 1.

For V+ systems, the vision server is the PC on which the ACE Sight vision software is running.

When a VRESULT is issued for a specific tool, it checks to see if that tool supports the VRESULT code. If the specified tool does not support the code, VRESULT moves to the parent tool to see if it supports the code. It continues up the chain until it finds a tool that supports the code. If no valid tool is found, an invalid vision result error is generated.

For example, suppose an Arc Finder tool is placed relative to a Blob Analyzer tool. In the application, the Blob Analyzer tool locates many blobs and adds an Arc Finder tool at each instance. If you ask for the blob area associated with an arc finder instance, VRESULT will recognize that the Arc Finder tool does not support that code, so it moves to the parent tool (the Blob Analyzer tool) and finds the blob instance associated with the specified arc result. It validates that the blob result supports the VRESULT code, and so it returns the data.

Some vision tools are considered ***Frame Sources***.The Blob Analyzer and Locator tool are the most commonly used Frame Sources. When these tools execute, it will mark each results as a separate frame or grouping. Any vision tools relative to a Frame Source will associacte each of it's results with the frame it is relative to. In this case, you may want to use the *frame_id* parameter to extract the results.

For example, going back to the Arc Finder tool relative to the Blob Analyzer tool. If the Blob Analyzer locates 5 different results, then the Arc Finder tool will execute 5 different Arc Finder operations, one relative to each result returned by the Blob Analyzer. The Arc Finder will associate each result with a frame number that correlates with the index of the result returned by the Blob Analyzer. So, if you want to get an Arc Fider result associated with the 4th result of the Blob Analyzer, you would reference *index_id* =1 in *frame_id* = 4. You are requesting the first instance in result frame 4. In this situation, you can still access all the Arc Finder results using *frame_id* = -1. But note, some child vision tools may have multiple results within each frame and sometimes might have no results within a frame.

Example

The following illustrates how to retrieve a specific tool result.

```
; Get the number of instances found by a Locator.
; instance count = 1310
instance_count = VRESULT($ip, 1, 2, 1, 1310)
```

VRUN program instruction

NOTE: This keyword can also be used to control the Adept AnyFeeder. For details, see the *Adept AnyFeeder User's Guide*, Rev B or later.

Syntax

Micro V+

```
VRUN sequence_id
```

V+

```
VRUN $ip, sequence_id
```

Description

Initiates the execution of a vision sequence.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

Details

The *sequence_id* parameter is optional. This parameter is 1-based. If no value is provided for this parameter, it defaults to 1.

For V+ systems, the vision server is the PC on which the ACE Sight vision software is running.

Example

```
; Execute the first sequence
VRUN $ip, 1
```

VSTATE real-valued function

NOTE: This keyword can also be used to control the Adept AnyFeeder. For details, see the *Adept AnyFeeder User's Guide*, Rev B or later.

Syntax

MicroV+

```
value = VSTATE (sequence_id)
```

V+

```
value = VSTATE ($ip, sequence_id)
```

Description

Returns the state of the execution of a sequence.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

Details

The *sequence_id* parameter is optional. This parameter is 1-based. If no value is provided, it defaults to 1.

In V+ the vision server is the PC on which the ACE Sight vision software is running.

Return

Return values are different for V+ and MicroV+:

MicroV+	
Value	**Description**
0	Running
1	This value is currently unused.

MicroV+	
Value	**Description**
2	Completed
3	Error

V+	
Value	**Description**
0	Idle
1	Running
2	Paused
3	Done
4	Error
5	Starting

Example

```
; Get the state of the first sequence
value = VSTATE($ip, 1)
```

VTIMEOUT system parameter

Syntax

MicroV+

```
PARAMETER VTIMEOUT = value
```

V+

```
PARAMETER VTIMEOUT = value
```

Description

Sets a timeout value so that an error message is returned if no response is received following a vision command. The timeout value is expressed in seconds (for example, the value 0.15 = 150 ms).

For the MicroV+ system, the default value is 0, which causes an infinite timeout. For the V+ system, the default value is 5 (seconds).

Details

For the MicroV+ system:

- It is important to set a value other than the default value of 0.
- VTIMEOUT = 0 sets the timeout to "infinite". In this case, the MicroV+ system will wait indefinitely for a response from the vision system.

For the V+ system, VTIMEOUT = 0 sets the timeout value to 16 ms (which is the minimum timeout that will be used).

Example

```
; Get error message if there is no response after 200 ms.
PARAMETER VTIMEOUT = 0.20
```

VWAITI program instruction

NOTE: This keyword can also be used to control the Adept AnyFeeder. For details, see the *Adept AnyFeeder User's Guide*, Rev B or later.

Syntax

MicroV+

```
VWAITI (sequence_id) type
```

V+

```
VWAITI (sequence_id) $ip, type
```

Description

Waits until the specified vision sequence reaches the state specified by the *type* parameter. Use a VWAITI call after VRUN. In a V+ conveyor-tracking application, the absence of a specific VWAITI instruction can interfere with the Virtual Camera tool and the Communication tool, and cause a delay in the execution of the application.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
type	**0** Wait for full completion (default). **1** Wait until picture acquisition has completed.

Details

The following parameters are optional: *sequence_id* and *type*. The *sequence_id* parameter is 1-based. If no value is provided for the *sequence_id* parameter, it defaults to 1. If no value is provided for the *type* parameter, it defaults to 0.

In V+, the vision server is the PC on which the ACE Sight vision software is running.

Example

```
; Execute the first sequence
VRUN $ip, 1
```

```
; Wait for full completion of first sequence
VWAITI (1) $ip, 0
```

ACE Sight Properties Reference for V+ and MicroV+

This chapter provides details on all ACE Sight properties and their use in V+ and MicroV+.

- All properties are described in alphabetical order in the following pages.
- To find a property by name, by tool, or by ID, click a link below.

Global Tables

All properties sorted by name or ID number:

Framework Properties

Properties required for configuring standalone vision applications:

Tool Properties

Properties that apply to the selected vision tool:

Image Processing Tool Properties

Image Sampling Tool Properties

Image Sharpness Tool Properties

Inspection Tool Properties

Line Finder Properties

Locator Model Tool Properties (**NOTE:** This tool cannot be accessed by ACE Sight)

Locator Tool Properties

Overlap Tool Properties

Pattern Locator Properties

Point Finder Properties

Remote Vision Tool Properties

Virtual Camera Tool Properties

Abort

VPARAMETER 5501

Stops the execution of the specified Virtual Camera tool. This property is write-only.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5501, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 5501, index_id, object_id) $ip = value
```

Type

Boolean

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5501: the value used to reference this property.
index_id	N/A
object_id	N/A

ActiveCalibration

VPARAMETER 5504

Reads and writes the index of the active calibration relative to a camera.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5504, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5504, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5504, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5504, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of calibrations associated with the tool.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5504: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ActiveSettings

VPARAMETER 5505

Reads and writes the index of the active settings relative to a camera.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5505, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5505, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5505, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5505, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of calibrations associated with the tool.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5505: the value used to reference this property.
index_id	Robot number to select.
object_id	Index of the tool tip to access.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederVRunCommand

VPARAMETER 6000

Identifies the command that is run when a VRUN is issued to the AnyFeeder.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6000, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6000, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6000, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6000, index_id, object_id)
```

Type

Real variable.

Range

A valid AnyFeeder command or sequence index. The valid AnyFeeder commands are:

Feed Forward	1
Feed Backward	2
Feed Flip Forward	3
Feed Flip Backward	4
Flip	5
Dispense	6
Purge	7
Heavy Dispense	8
Stop	15
Init	16
Error Reset	30

Firmware Restart	31
Backlight On	100
Backlight Off	101

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6000: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederDispenseIterations

VPARAMETER 6012

The number of iterations used in the dispense operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6012, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6012, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6012, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6012, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6012: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedBackwardIterations

VPARAMETER 6015

The number of iteration for the feedd backwards operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6015, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6015: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedBackwardSpeed

VPARAMETER 6005

The speed used for the feed backwards operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6005, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6005, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6005, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6005, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6005: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedFlipBackwardIterations

VPARAMETER 6015

The number of itterations used for the feed flip backwards operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6015, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6015, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6015: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedFlipBackwardSpeed

VPARAMETER 6008

The speed used for the feed flip backwards operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6008, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6008, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6008, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6008, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6008: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedFlipForwardIterations

VPARAMETER 6017

The number of iterations used for the feed flip forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6017, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6017, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6017, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6017, index_id, object_id)
```

Type

Real variable.

RangeRange

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6017: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedFlipForwardSpeed

VPARAMETER 6007

The speed used for the feed flip forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6007, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6007, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6007, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6007, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6007: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedForwardIterations

VPARAMETER 6011

The number of iterations used for the feed forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6011, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6011, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6011, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6011, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6011: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFeedForwardSpeed

VPARAMETER 6001

The speed used for the feed forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6001, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6001, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6001, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6001, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6001: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFlipIterations

VPARAMETER 6013

The number of iterations used for the flip operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6013, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6013, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6013, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, 6013, 6000, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6013: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederFlipSpeed

VPARAMETER
6003

The speed used for the flip operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6003, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6003, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6003, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6003, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6003: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederHeavyDispenseIterations

VPARAMETER 6016

The number of iterations used for the heavy dispense operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6016, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6016, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6016, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6016, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 63

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6016: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederHeavyDispenseSpeed

VPARAMETER 6006

The speed used for the heavy dispense operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6006, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6006: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederHeavyDispenseSpeed

VPARAMETER 6006

The speed used for the heavy dispense operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6006, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6006, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6006: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

AnyFeederPurgeSpeed

VPARAMETER 6004

The speed used for the purge operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 6004, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 6004, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 6004, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 6004, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	6004: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ArcMustBeTotallyEnclosed

VPARAMETER 5141

When set to True, the start and end points of the arc must be located on the radial bounding sides of the Search Area. When set to False, the found arc can enter and/or exit the Search Area at the inner or outer annular bounds of the Search Area.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5141, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5141, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5141, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5141, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Start and end points of the arc must be located on the sides of the bounding area.
0	Start and end points of the arc can be anywhere inside or outside of the bounding area.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5141: the value used to reference this property.
index_id	N/A
object_id	N/A

ArithmeticClippingMode

VPARAMETER
5360

Clipping mode applied by an arithmetic operation.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5360, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5360, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5360, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5360, index_id, object_id)
```

Remarks

hsClippingNormal mode forces the destination pixel value to a value from 0 to 255 for unsigned 8-bit images, to a value from -327678 to 32767 for signed 16 bits images and so on. Values that are less than the specified minimum value are set to the minimum value. Values greater than the specified maximum value are set to the maximum value.

hsClippingAbsolute mode takes the absolute value of the result and clips it using the same algorithm as for the hsClippingNormal mode.

Type

Long

Range

Value	Image Processing Clipping Mode	Description
0	hsClippingNormal	Normal clipping method is used.
1	hsClippingAbsolute	Absolute clipping method is used.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5360: the value used to reference this property.
index_id	N/A
object_id	N/A

ArithmeticConstant

VPARAMETER
5361

Constant applied by an arithmetic operation when no valid operand image is specified.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5361, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5361, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5361, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5361, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 256

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5361: the value used to reference this property.
index_id	N/A
object_id	N/A

ArithmeticScale

VPARAMETER
5362

Scaling factor applied by an arithmetic operation.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5362, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5362, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5362, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5362, index_id, object_id)
```

Type

Double

Range

Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5362: the value used to reference this property.
index_id	N/A
object_id	N/A

AssignmentConstant

VPARAMETER
5365

Constant applied by an assignment operation.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5365, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5365, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5365, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5365, index_id, object_id)
```

Type

Long

Range

Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5365: the value used to reference this property.
index_id	N/A
object_id	N/A

AssignmentHeight

VPARAMETER
5366

Constant value that defines the height of the output image. This property is read-only.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5366, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5366, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5366, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5366, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 2048

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5366: the value used to reference this property.
index_id	N/A
object_id	N/A

AssignmentWidth

VPARAMETER
5367

Constant value that defines the width of the output image. This property is read-only.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5367, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5367, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5367, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5367, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 2048

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5367: the value used to reference this property.
index_id	N/A
object_id	N/A

AutoCoarsenessSelectionEnabled

VPARAMETER
5421

When is set to True, the value of SearchCoarseness is automatically determined by the Pattern Locator process when the pattern is learned.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5421, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5421, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5421, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5421, index_id, object_id)
```

Type

Long

Range

Value	Description
1	The Coarseness levels are automatically determined and set by the tool.
0	The Coarseness levels are set by the user.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5421: the value used to reference this property.

index_id	N/A
object_id	N/A

AutomaticCandidateCountEnabled

VPARAMETER
5301

When set to True, the number of candidate measurement points is automatically determined according to the dimension of the tool region of interest. When set to False, the number of candidate measurement points is set manually through the CandidatePointsCount property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5301, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5301, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5301, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5301, index_id, object_id)
```

Type

Long

Range

Value	Description
1	The number of candidate measurement points is set automatically.
0	The number of candidate measurement points is set manually.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5301: the value used to reference this property.

index_id	N/A
object_id	N/A

AverageContrast

VRESULT
1801

Average contrast, expressed in greylevel values, between light and dark pixels on either side of the found entity (point, line, or arc). This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1801, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1801, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0.

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1801: the value used to reference this property.
index_id	N/A
frame_id	Frame that contains the entity for which you want the result.

BeltCalibrationDownstreamLimit

VLOCATION
10002

The downstream limit of the belt, which is defined during the Belt Calibration. Expressed as a transform. This property is read-only.

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10002, index_id, frame_id)
```

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
          SmartController.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	N/A
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10002: the value used to reference this property.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

BeltCalibrationFrame

VLOCATION
10000

The belt frame of reference, which is defined during the Belt Calibration. Expressed as a transform. This property is read-only.

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10000, index_id, frame_id)
```

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
           SmartController.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10000: the value used to reference this property.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

BeltCalibrationNearsideLimit

VLOCATION
10003

The nearside limit of the belt, which is defined during the Belt Calibration. Expressed as a transform. This property is read-only.

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10003, index_id, frame_id)
```

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
        SmartController.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	N/A
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10003: the value used to reference this property.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

BeltCalibrationScale

VPARAMETER
10004

The scale factor between encoder counts and millimeters, which is defined during the Belt Calibration. This is the number of millimeters that the belt advances for each encoder count. This property is read-only.

Syntax

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 10004, index_id, object_id)
```

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
SmartController.
```

Type

Double

Parameters

sequence_id	Must be set to -1.
tool_id	The camera number referenced in the keyword mapping of the camera calibration object.
ID	10004: the value used to reference this property.
index_id	N/A
object_id	The robot number associated with the calibration.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

BeltCalibrationUpstreamLimit

VLOCATION
10001

The upstream limit of the belt, which is defined during the Belt Calibration. Expressed as a transform. This property is read-only.

Syntax

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
          SmartController.
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10001, index_id, frame_id)
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	N/A
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10001: the value used to reference this property.
index	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

BeltLatchCalibrationOffset

VLOCATION
10010

The robot-to-latch calibration offset, which is defined during the Latch Calibration. Expressed as a transform. This property is read-only.

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10010, index_id, frame_id)
```

MicroV+

```
Not applicable. Conveyor-tracking is supported only in V+ running on the Adept
        SmartController.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10010: the value used to reference this property.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

BilinearInterpolationEnabled

VPARAMETER
120

Specifies if bilinear interpolation is used to sample the input image. By default, bilinear interpolation is enabled because it ensures subpixel accuracy.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 120, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 120, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 120, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 120, index_id, object_id)
```

Remarks

Bilinear interpolation is crucial for obtaining accurate results with inspection tools. When a tool is positioned in frame-based mode, the tool region of interest is rarely aligned with the pixel grid, which results in jagged edges on edges of objects. The bilinear interpolation function smooths out the jaggedness within the sampled image by attributing to each pixel a value interpolated from values of neighboring pixels, which provides a more true-to-life representation of contours, as illustrated in the following figure.

Uninterpolated sampling may provide a small increase in speed but will provide less accurate results.

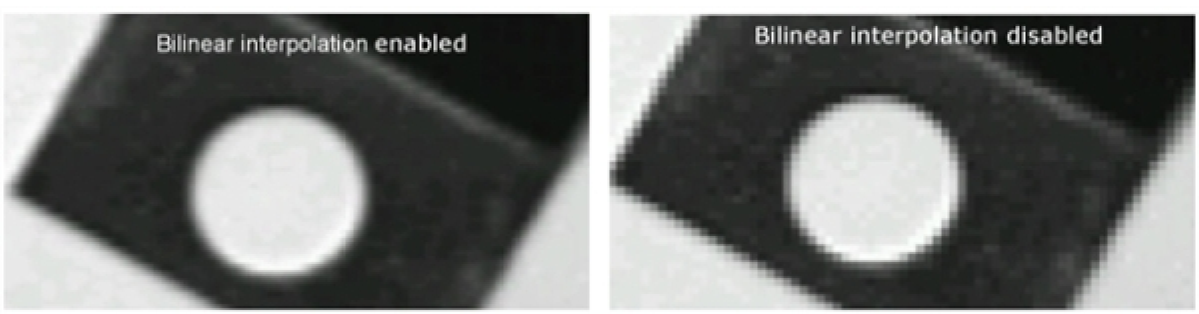


Figure: Effect of Bilinear Interpolation

Type

Boolean

Range

Value	Description
1	Bilinear interpolation is enabled. Recommended default setting.
0	Bilinear interpolation is disabled.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	120: the value used to reference this property.
index_id	N/A
object_id	N/A

BlobArea

VRESULT
1611

Area of the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1611, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1611, index_id, frame_id)
```

Type

Double

Range

Minimum: MinimumBlobArea

Maximum: MaximumBlobArea

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1611: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxBottom

VRESULT
1648

The bottommost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system. This value is returned with respect to the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1648, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1648, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1648: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxCenterX

VRESULT
1624

X-coordinate of the center of the bounding box aligned with the Tool coordinate system. This value is returned with respect to the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1624, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1624, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1624: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxCenterY

VRESULT
1625

Y-coordinate of the center of the bounding box aligned with the Tool coordinate system. This value is returned with respect to the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1625, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1625, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1625: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxHeight

VRESULT
1626

Height of the bounding box with respect to the Y-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1626, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1626, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1626: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxLeft

VRESULT
1645

The leftmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system. This value is returned with respect to the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1645, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1645, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1645: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxRight

VRESULT
1646

The rightmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1646, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1646, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1646: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxRotation

VRESULT
1649

Rotation of the bounding box with respect to the X-axis of the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1649, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1649, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 360

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1649: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxTop

VRESULT
1647

The topmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system. This value is returned with respect to the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1647, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1647, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1647: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobBoundingBoxWidth

VRESULT
1627

Width of the bounding box with respect to the X-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1627, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1627, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1627: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobChainCode

VRESULT
1656

Direction, in Tool coordinates, of a given boundary element in the chain code. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1656, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1656, index_id, frame_id)
```

Type

Long

Range

Value	Name	Description
0	hsDirectionRight	Right direction
1	hsDirectionTop	Top direction
2	hsDirectionLeft	Left direction
3	hsDirectionBottom	Bottom direction

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1656: the value used to reference this property.

index_id	Index of the selected boundary element.
frame_id	Frame containing the blob for which you want the result.

BlobChainCodeDeltaX

VRESULT
1659

Horizontal length, in Tool coordinates, of a boundary element in the chain code. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1659, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1659, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1659: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobChainCodeDeltaY

VRESULT
1660

Vertical length, in Tool coordinates, of a boundary element in the chain code. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1660, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1660, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1660: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobChainCodeLength

VRESULT
1655

Number of boundary elements in the chain code of the blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1655, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1655, index_id, frame_id)
```

Type

Long

Range

Minimum: Greater than 4

Maximum: unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1655: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobChainCodeStartX

VRESULT
1657

X-position, in Tool coordinates, of the first pixel in the chain code. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1657, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1657, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1657: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobChainCodeStartY

VRESULT
1658

Y-position, in Tool coordinates, of the first pixel in the chain code. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1658, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1658, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1658: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobConvexPerimeter

VRESULT
1614

Convex perimeter of the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1614, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1614, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0.0

Maximum: unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1614: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobCount

VRESULT
1610

Number of blobs detected by the tool. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1610, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1610, index_id, frame_id)
```

Type

Long

Range

Minimum:

Maximum: 65534

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1610: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobElongation

VRESULT
1616

Ratio of the moment of inertia about the blob's minor axis (BlobInertiaMaximum) to the moment of inertia about the blob's major axis (BlobInertiaMinimum). This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1616, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1616, index_id, frame_id)
```

Remarks

No units.

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1616: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobExtentBottom

VRESULT
1653

Distance along the Y-axis between the blob's center of mass and the bottom side of the bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1653, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1653, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1653: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobExtentLeft

VRESULT
1650

Distance along the Y-axis between the blob's center of mass and the bottom side of the bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1650, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1650, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1650: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobExtentRight

VRESULT
1651

Distance along the X-axis between the blob's center of mass and the right side of the bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1651, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1651, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1651: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobExtentTop

VRESULT
1652

Distance along the Y-axis between the blob's center of mass and the top side of the bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1652, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1652, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1652: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobGreyLevelMaximum

VRESULT
1622

Highest greylevel value of the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1622, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1622, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1622: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobGreyLevelMean

VRESULT
1618

Mean greylevel value in the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1618, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1618, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1618: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobGreyLevelMinimum

VRESULT
1621

Lowest greylevel value in the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1621, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1621, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1621: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobGreyLevelRange

VRESULT
1619

Range of the greylevel values in the selected blob. The range is calculated as [BlobGreyLevelMaximum - BlobGreyLevelMinimum + 1]. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1619, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1619, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1619: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobGreyLevelStdDev

VRESULT
1620

Standard deviation of the greylevel values in the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1620, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1620, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1620: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobHoleCount

VRESULT
1654

The number of holes found in the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1654, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1654, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1654: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobInertiaMaximum

VRESULT
1633

Moment of inertia about the minor axis, which corresponds to the highest moment of inertia. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1633, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1633, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than .0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1633: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobInertiaMinimum

VRESULT
1632

Moment of inertia about the major axis, which corresponds to the lowest moment of inertia. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1632, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1632, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1632: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobInertiaXAxis

VRESULT
1634

Moment of inertia about the X-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1634, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1634, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1634: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobInertiaYAxis

VRESULT
1635

Moment of inertia about the Y-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1635, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1635, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0.

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1635: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxBottom

VRESULT
1639

The bottommost coordinate of the bounding box with respect to the X-axis (major axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1639, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1639, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1639: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxCenterX

VRESULT
1628

X-coordinate of the center of the bounding box with respect to the X-axis (major axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1628, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1628, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1628: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxCenterY

VRESULT
1629

Y-coordinate of the center of the bounding box with respect to the Y-axis (minor axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1629, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1629, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1629: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxHeight

VRESULT
1630

Height of the bounding box with respect to the Y-axis (minor axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1630, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1630, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1630: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxLeft

VRESULT
1636

The leftmost coordinate of the bounding box aligned with respect to the X-axis (major axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1636, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1636, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1636: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxRight

VRESULT
1637

The rightmost coordinate of the bounding box aligned with the X-axis (major axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1637, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1637, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1637: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxRotation

VRESULT
1640

Rotation of the intrinsic bounding box with respect to the X-axis of the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1640, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1640, index_id, frame_id)
```

Type

Double

Range

Minimum: -180

Maximum: 180

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1640: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxTop

VRESULT
1638

The topmost coordinate of the bounding box aligned with the Y-axis (minor axis) of the principal axes. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1638, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1638, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1638: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicBoundingBoxWidth

VRESULT
1631

Width of the bounding box with respect to the X-axis of the Tool coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1631, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1631, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1631: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicExtentBottom

VRESULT
1644

Distance along the minor axis between the blob's center of mass and the bottom side of the intrinsic bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1644, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1644, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1644: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicExtentLeft

VRESULT
1641

Distance along the major axis between the blob's center of mass and the left side of the intrinsic bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1641, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1641, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1641: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicExtentRight

VRESULT
1642

Distance along the major axis between the blob's center of mass and the right side of the intrinsic bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1642, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1642, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1642: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobIntrinsicExtentTop

VRESULT
1643

Distance along the major axis between the blob's center of mass and the top side of the intrinsic bounding box. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1643, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1643, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1643: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobPositionX

VRESULT
1612

X-coordinate of the center of mass of a given blob in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1612, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1612, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1612: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobPositionY

VRESULT
1613

Y-coordinate of the center of mass of a given blob in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1613, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1613, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1613: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobPrincipalAxesRotation

VRESULT
1617

Angle of axis of the smallest moment of inertia with respect to the X-axis of the selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1617, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1617, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1617: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobRawPerimeter

VRESULT
1615

Raw perimeter of the selected blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1615, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1615, index_id, frame_id)
```

Type

Double

Range

Minimum: Greater than 0.

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1615: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the blob for which you want the result.

BlobRoundness

VRESULT
1623

Degree of similarity between the blob and a circle. The roundness is 1 for a perfectly-circular blob. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1623, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1623, index_id, frame_id)
```

Remarks

No units.

Type

Double

Range

Minimum: Greater than 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	1623: the value used to reference this property.

index_id	N/A
frame_id	Frame containing the blob for which you want the result.

CalibratedUnitsEnabled

VPARAMETER
103

Returns 1 if the image the virtual camera image the tool is operating on has an active calibration applied. If no calibration is in effect, a 0 is returned. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 103, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 103, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Dimensions are expressed in millimeters. (Default)
0	Dimensions are expressed in pixel units.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	103: the value used to reference this property.
index_id	N/A
object_id	N/A

CandidatePointsCount

VPARAMETER
5300

Sets the number of candidate locations where the tool tries to evaluate the sharpness. When the tool is executed, it scans the region of interest and identifies a number of candidate locations (equal to CandidatePointsCount) where the local standard deviation is the highest. The local sharpness is then evaluated at each candidate location that has a local standard deviation above the StandardDeviationThreshold property.

The number of locations where the sharpness is actually measured is returned by the MeasurementPointsCount property. When the AutomaticCandidateCountEnabled property is True, the number of candidate measurement points is automatically determined according to the size of the region of interest and CandidatePointsCount.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5300, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5300, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5300, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5300, index_id, object_id)
```

Type

Long

Range

Minimum: Greater than 0.

Maximum: 32767

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5300: the value used to reference this property.
index_id	N/A
object_id	N/A

ChainCodeResultsEnabled

VPARAMETER
1607

Enables the computation of the blob chain code properties: BlobChainCode, BlobChainCodeDeltaX, BlobChainCodeDeltaY, BlobChainCodeLength, BlobChainCodeStartX and BlobChainCodeStartY

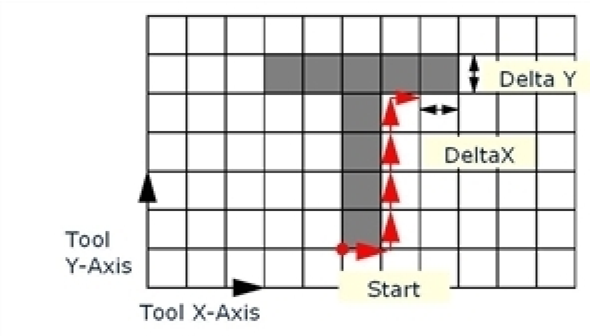


Figure: Illustration of Chain Code Results

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1607, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1607, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1607, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1607, index_id, object_id)
```

Type

Boolean

Range

Index	Description
1	Chain Code Results are output by the tool.
0	Chain Code Results are not output.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1607: the value used to reference this property.
index_id	N/A
object_id	N/A

ColorFilterCount

VPARAMETER
5700

Returns the number of filters that are defined for the Color Matching tool. This property is read-only.

Syntax

Micro V+

```
value = VPARAMETER (sequence_id, tool_id, 5700, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5700, index_id, object_id)
```

Remarks

ColorFilterCount reports the number of filters that are defined in the tool and that appear in the Filters list in the interface. This value is not affected by the number of filter results in an image.

Type

Long

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5700: the value used to reference this property.
index_id	N/A
object_id	N/A

ColorFilterMatchPixelCount

VRESULT
2502

Counts the number of pixels that match the conditions set by the filter. This result is output for each filter, starting at Filter 0. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2502, filter_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2502, filter_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: ImagePixelCount

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	2502: the value used to reference this property.
filter_id	Index of the filter for which you want the result. First Filter is 0.
frame_id	Frame for which you want the results.

ColorFilterMatchQuality

VRESULT
2501

Calculates the percentage of pixels matched to the specified filter. This value is equal to the number of matched pixels (**Filter (n) Match Pixel Count**), divided by the total number of pixels in the region of interest (**Image Pixel Count**). This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2501, filter_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2501, filter_id, frame_id)
```

Type

Long

Range

Minimum: Greater than 0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the blob for which you want the result.
ID	2501: the value used to reference this property.
filter_id	Index of the filter for which you want the result. First Filter is 0.
frame_id	Frame for which you want the results.

CommunicationToolResults

VRESULT
2600

Returns the total number of results that have been queued by all communication tools within a given sequence. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2600, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2600, index_id, frame_id)
```

Type

Integer

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Should always be set to -1.
instance_id	Not used
ID	2600: the value used to reference this property.
index_id	Not used
frame_id	Not used

ConformityTolerance

VPARAMETER
556

Maximum local deviation between the expected model contours of an instance and the contours actually detected in the input image. It corresponds to the maximum distance by which a matched contour can deviate from either side of its expected position in the model. This property can only be set when UseDefaultConformityTolerance is set to False. Otherwise, it is read-only.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 556, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 556, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 556, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 556, index_id, object_id)
```

Type

Double

Remarks

This property can be set to any positive value if ConformityToleranceRangeEnabled is set to False.

Type

Double

Range

Minimum: MinimumConformityTolerance

Maximum: MaximumConformityTolerance

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	556: the value used to reference this property.
index_id	N/A
object_id	N/A

ConformityToleranceRangeEnabled

VPARAMETER
553

When ConformityToleranceRangeEnabled is set to True, the allowable range of values for ConformityTolerance is set by the read-only MinimumConformityTolerance and MaximumConformityTolerance properties. When set to False, ConformityTolerance can be set to any positive value.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 553, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 553, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 553, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 553, index_id, object_id)
```

Remarks

Disabling the conformity tolerance range can be useful for finding deformable objects, which requires a high conformity tolerance value for a better match.

Type

Boolean

Range

Value	Description
0	ConformityToleranceRange is enabled.
1	ConformityToleranceRange is disabled.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	553: the value used to reference this property.
index_id	N/A
object_id	N/A

Connectivity

VPARAMETER
5120

Defines a minimum number of connected edges required to generate a point hypothesis from a a specific found edge, which satisfies the search constraints.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5120, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5120, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5120, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5120, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 20

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5120: the value used to reference this property.
index_id	N/A
object_id	N/A

Constraints

VPARAMETER
5220

Defines the edge detection constraints of an Arc Locator tool or an Edge Locator tool. Constraints can be set for position and/or magnitude and are used to score edges.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5220, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5220, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5220, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5220, index_id, object_id)
```

Type

Long

Range

Value	Constraints NAme	Description
0	hsNone	No constraint.
1	hsPosition	Position constraint.
2	hsMagnitude	Magnitude constraint.
3	hsAllConstraints	Magnitude and position

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5220: the value used to reference this property.
index_id	N/A
object_id	N/A

ContrastPolarity

VPARAMETER
522

Selects the type of polarity accepted for object recognition. Contrast polarity identifies the direction of change in greylevel values between an object and its surrounding area. Polarity is always defined with respect to the initial polarity in the image on which the model was created.

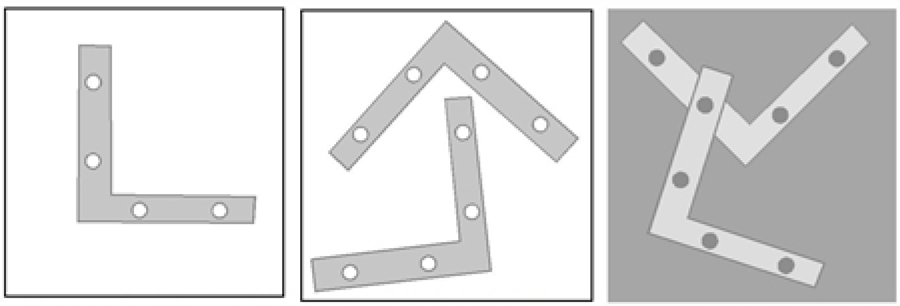


Figure: Contrast Polarity

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 522, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 522, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 522, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 522, index_id, object_id)
```

Type

Long

Range

Value	hsContrastPolarity	Description
0	hsContrastPolarityNormal	The Locator accepts only instances having the same

Value	hsContrastPolarity	Description
		polarity as that of the model and does not recognize local changes in polarity.
1	hsContrastPolarityReverse	The Locator accepts only instances having the inverse polarity as that of the model and does not recognize local changes in polarity.
2	hsContrastPolarityNormalAndReverse	The Locator accepts only instances having a polarity that is either the same or the inverse of the model's polarity but does not recognize local changes in polarity.
3	hsContrastPolarityDontCare	Accepts any polarity for the object, INCLUDING local changes in polarity.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	522: the value used to reference this property.
index_id	N/A
object_id	N/A

ContrastThreshold

VPARAMETER
303

Defines the minimum contrast needed for an edge to be detected in the input image and used for arc computation. This threshold is expressed in terms of a step in greylevel values. Except when ContrastThresholdMode is set to hsContrastThresholdFixedValue, the property is read-only.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 303, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 303, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 303, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 303, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 255

Remark(s)

By default, the tool selects a ContrastThresholdMode based on image content to provide flexibility to variations in image lighting conditions and contrast. Adaptive threshold modes are generally recommended. A fixed-value contrast threshold should only be used when adaptive values do not provide satisfactory results.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	303: the value used to reference this property.
index_id	N/A
object_id	N/A

Related Properties

ContrastThresholdMode

ContrastThresholdMode

VPARAMETER
302

Selects the method used to compute the threshold used for detecting edges in the input image.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 302, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 302, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 302, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 302, index_id, object_id)
```

Type

Long

Remarks

By default, the tool selects a **ContrastThresholdMode** based on image content to provide flexibility to variations in image lighting conditions and contrast. Adaptive threshold modes are generally recommended. A fixed-value contrast threshold should only be used when adaptive values do not provide satisfactory results.

Range

The valid range for this property is as follows:

Value	Contrast Threshold Mode Name	Description
0	hsContrastThresholdAdaptiveLowSensitivity	Uses a low sensitivity adaptive threshold for detecting edges. Adaptive Low Sensitivity reduces the amount of noisy edges but may also cause significant edges to be undetected.
1	hsContrastThresholdAdaptiveNormalSensitivity	Uses a normal sensitivity adaptive threshold for detecting edges.

Value	Contrast Threshold Mode Name	Description
2	hsContrastThresholdAdaptiveHighSensitivity	Uses a high sensitivity adaptive threshold for detecting edges. Adaptive High Sensitivity can help detect weak-contrast edges but also increases the amount of noisy edges.
3	hsContrastThresholdFixedValue	Uses a fixed value threshold for detecting edges.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	302: the value used to reference this property.
index_id	N/A
object_id	N/A

DefaultConformityTolerance

VPARAMETER
552

Default value for ConformityTolerance computed by the Locator by analyzing the calibration, the contour detection parameters, and the search parameters. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 552, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 552, index_id, object_id)
```

Remarks

This default value is used for ConformityTolerance when UseDefaultConformityTolerance is set to True.

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	552: the value used to reference this property.
index_id	N/A
object_id	N/A

Related Properties

MaximumConformityTolerance
MinimumConformityToleranceInstanceLocationGripperOffsetMinimum

DetailLevel

VPARAMETER
301

The coarseness of the contours at the Detail level. This property can only be set when ParametersBasedOn is set to hsParametersCustom. Otherwise, it is read-only.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 301, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 301, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 301, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 301, index_id, object_id)
```

Remarks

For most applications, the ParametersBasedOn property should be set to hsParametersAllModels. Custom contour detection should only be used when the default values do not work correctly.

Type

Long

Range

Minimum: 1

Maximum: 16

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	301: the value used to reference this property.
index_id	N/A
object_id	N/A

Edge1Constraints

VPARAMETER
5221

Defines the detection constraints for the first edge of the selected pair. Constraints can be set for position and/or magnitude and are used to score edges.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5221, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5221, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5221, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5221, index_id, object_id)
```

Type

Long

Range

Value	Constraint Name	Description
0	hsNone	No constraint
1	hsPosition	Position constraint
2	hsMagnitude	Magnitude constraint
3	hsAllConstraints	Magnitude and position constraints.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5221: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1]
object_id	N/A

Edge1Magnitude

VRESULT
1940

Magnitude of the first edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1940, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1940, index_id, frame_id)
```

Type

Double

Range

Minimum: -255

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1940: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1MagnitudeConstraint

VPARAMETER 5227

Indexed property used to set the magnitude constraint function. Two points are used: Base and Top.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5227, index_id, constraint_id) = value
value = VPARAMETER (sequence_id, tool_id, 5227, index_id, constraint_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5227, index_id, constraint_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5227, index_id, constraint_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5227: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
constraint_id	One of the two points of the magnitude constraint function (hsMagnitudeConstraintIndex) 1: Base point 2: Top point

Edge1MagnitudeScore

VRESULT
1942

Magnitude score of the first edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1942, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1942, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1942: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1PolarityMode

VPARAMETER
5211

Selection criterion for the first edge of the selected pair. The greyscale transition of the edge must respect the polarity set by this property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5211, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5211, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5211, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5211, index_id, object_id)
```

Type

Long

Range

Value	Polarity Mode Name	Description
0	hsDarkToLight	The greylevel value must go from dark to light when crossing an edge.
1	hsLightToDark	The greylevel value must go from light to dark when crossing an edge.
2	hsEitherPolarity	The change in greylevel value is not a criterion for locating an edge.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5211: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge1PositionConstraint

VPARAMETER
5224

Indexed property used to set the position constraint function of the first edge of the selected pair. Four points are used: Base Left, Top Left, Top Right, Base Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5224, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5224, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5224, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5224, index_id, object_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5224: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge1PositionScore

VRESULT
1944

Position score of the first edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1944, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1944, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1944: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1PositionX

VRESULT
1946

X-coordinate of the center of the first edge of the selected pair in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1946, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1946, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1946: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1PositionY

VRESULT
1947

Y-coordinate of the center of the first edge of the selected pair in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1947, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1947, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1947: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1Radius

VRESULT
1954

Radius of the first edge of the selected pair. ToolPositionX and ToolPositionY are at the center of the circular arc described by the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1954, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1954, index_id, frame_id)
```

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1954: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1Rotation

VRESULT
1950

Rotation of the first edge of the selected pair in the vision coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1950, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1950, index_id, frame_id)
```

Remarks

The rotation is defined as the angle between the X-axis of the vision coordinate system and the selected edge.

Type

Double

Range

Minimum: -180

Maximum: 180

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1950: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].

frame_id	Index of the frame containing the edge pair.

Edge1Score

VRESULT
1952

Minimum score needed to accept an edge as the first edge of the selected pair. The score is computed according to the constraints set by the Edge1Constraints property. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1952, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1952, index_id, frame_id)
```

Remarks

The rotation is defined as the angle between the X-axis of the vision coordinate system and the selected edge.

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1952: the value used to reference this property.

index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge1ScoreThreshold

VPARAMETER
5241

Minimum score needed to accept an edge as the first edge of the selected pair. The score of the first edge is returned by the Edge1Score property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5241, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5241, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5241, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5241, index_id, object_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5241: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	Index of the frame containing the edge pair.

Edge2Constraints

VPARAMETER
5222

Defines the detection constraints for the second edge of the selected pair. Constraints can be set for position and/or magnitude and are used to score edges.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5222, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5222, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5222, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5222, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
0	hsNone	No constraint
1	hsPosition	Position constraint
2	hsMagnitude	Magnitude constraint
3	hsAllConstraints	Magnitude and position constraints.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5222: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge2Magnitude

VRESULT
1941

Magnitude of the second edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1941, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1941, index_id, frame_id)
```

Type

Double

Range

Minimum: -255

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1941: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2MagnitudeConstraint

VPARAMETER
5228

Indexed property used to set the magnitude constraint function of the second edge of the selected pair. Two points are used: Base and Top.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5228, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5228, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5228, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5228, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5228: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge2MagnitudeScore

VRESULT
1943

Magnitude score of the second edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1943, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1943, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1943: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2PolarityMode

VPARAMETER
5212

Selection criterion for the second edge of the selected pair. The greyscale transition of the edge must respect the polarity set by this property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5212, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5212, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5212, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5212, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
0	hsDarkToLight	The greylevel value must go from dark to light when crossing an edge.
1	hsLightToDark	The greylevel value must go from light to dark when crossing an edge.
2	hsEitherPolarity	The change in greylevel value change is not a criterion for locating an edge.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5212: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge2PositionConstraint

VPARAMETER
5225

Indexed property used to set the position constraint function of the second edge of the selected pair. Four points are used: Base Left, Top Left, Top Right, Base Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5225, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5225, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5225, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5225, index_id, object_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5225: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	N/A

Edge2PositionScore

VRESULT
1945

Position score of the second edge of the selected pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1945, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1945, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1945: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2PositionX

VRESULT
1948

X-coordinate of the center of the second edge of the selected pair in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1948, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1948, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1948: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2PositionY

VRESULT
1949

Y-coordinate of the center of the second edge of the selected pair in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1949, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1949, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1949: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2Radius

VRESULT
1955

Radius of the second edge of the selected pair. ToolPositionX and ToolPositionY ar at center of the circular arc described by the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1955, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1955, index_id, frame_id)
```

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1955: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2Rotation

VRESULT
1951

Rotation of the second edge of the selected pair in the vision coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1951, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1951, index_id, frame_id)
```

Remarks

The rotation is defined as the angle between the X-axis of the vision coordinate system and the selected edge.

Type

Double

Range

Minimum: -180

Maximum: 180

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1951: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].

frame_id	Index of the frame containing the edge pair.

Edge2Score

VRESULT
1953

Minimum score to accept an edge as the second edge of the selected pair The score is computed according to the constraints set by the Edge2Constraints property. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1953, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1953, index_id, frame_id)
```

Remarks

The rotation is defined as the angle between the X-axis of the vision coordinate system and the selected edge.

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1953: the value used to reference this property.

index_id	Index of the edge pair. Range [1, PairCount -1].
frame_id	Index of the frame containing the edge pair.

Edge2ScoreThreshold

VPARAMETER
5242

Minimum score to accept an edge as the second edge of the selected pair. The score of the second edge is returned by the Edge2Score property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5242, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5242, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5242, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5242, index_id, object_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5242: the value used to reference this property.
index_id	Index of the edge pair. Range [1, PairCount -1].
object_id	Index of the frame containing the edge pair.

EdgeCount

VRESULT
1900

Number of edges detected by the tool. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1900, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1900, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	N/A
ID	1900: the value used to reference this property.
index_id	N/A
frame_id	N/A

EdgeFilterHalfWidth

VPARAMETER
5203

Half-width of the convolution filter used to compute the edge magnitude curve from which actual edges are detected. The filter approximates the first derivative of the projection curve. The half width of the filter should be set in order to match the width of the edge in the projection curve (the extent of the greyscale transition expressed in number of pixels).

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5203, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5203, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5203, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5203, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 25

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5203: the value used to reference this property.
index_id	N/A
object_id	N/A

EdgeMagnitude

VRESULT
1901

Magnitude of the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1901, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1901, index_id, frame_id)
```

Type

Long

Range

Minimum: -255

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1901: the value used to reference this property.
index_id	Index of the edge.
frame_id	Index of the frame containing the edge.

EdgeMagnitudeScore

VRESULT
1902

Magnitude score of the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1902, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1902, index_id, frame_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1902: the value used to reference this property.
index_id	Index of the edge.
frame_id	Index of the frame containing the edge.

EdgeMagnitudeThreshold

VPARAMETER
5201

Magnitude threshold is used to find edges on the magnitude curve. In order to locate edges, a subpixel, peak-detection algorithm is applied on the region of every minimum or maximum of the curve that exceeds this threshold.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5201, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5201, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5201, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5201, index_id, object_id)
```

Type

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5201: the value used to reference this property.
index_id	N/A
object_id	N/A

EdgePolarityMode

VPARAMETER
5210

Edge-selection criterion. The greyscale transition of the edge must respect the polarity set by this property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5210, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5210, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5210, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5210, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
0	hsDarkToLight	The greylevel value must go from dark to light when crossing an edge.
1	hsLightToDark	The greylevel value must go from light to dark when crossing an edge.
2	hsEitherPolarity	The change in greylevel value change is not a criterion for locating an edge.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5210: the value used to reference this property.
index_id	N/A
object_id	N/A

EdgePositionScore

VRESULT
1903

Position score of the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1903, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1903, index_id, frame_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1903: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgePositionX

VRESULT
1904

X-coordinate of the center of the selected edge in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1904, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1904, index_id, frame_id)
```

Type

Long

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1904: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgePositionY

VRESULT
1905

Y-coordinate of the center of the selected edge in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1905, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1905, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1905: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgeRadius

VRESULT
1908

Radius of the selected edge. ToolPositionX and ToolPositionY designate the center of the circular arc described by the selected edge. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1908, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1908, index_id, frame_id)
```

Type

Range

Minimum: -180

Maximum: 180

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1908: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgeRotation

VRESULT
1906

Rotation of the selected edge with respect to the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1906, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1906, index_id, frame_id)
```

Type

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1906: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgeScore

VRESULT
1907

Score of the selected edge. The score is computed according the constraints set by the Constraints property. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1907, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1907, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1907: the value used to reference this property.
index_id	Index of the edge for which you want the results.
frame_id	Index of the frame that contains the selected edge.

EdgeSortResultsEnabled

VPARAMETER 5243

Property that specifies if edges are sorted in descending order of score values.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5243, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5243, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5243, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5243, index_id, object_id)
```

Type

Boolean

Range

Value	Description
0	The edges are sorted in descending order of score values.
1	The edges are not sorted.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5243: the value used to reference this property.
index_id	N/A
object_id	N/A

ElapsedTime

VRESULT
1001

Total time elapsed (in milliseconds) during the last execution of the Locator tool. This time includes the time for the learn process, the time for the search process, and the overhead required to create and output the results structures. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1001, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1001, index_id, frame_id)
```

Remarks

This property returns the total elapsed time; it is not the used CPU time.

Type

Double

Range

Minimum: 0.0

Maximum: unlimited

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	N/A
ID	1001: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance.

	Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ExtrinsicInertiaResultsEnabled

VPARAMETER
1604

Enables the computation of the following blob properties: BlobInertiaXAxis, BlobInertiaYAxis and BlobPrincipalAxesRotation.

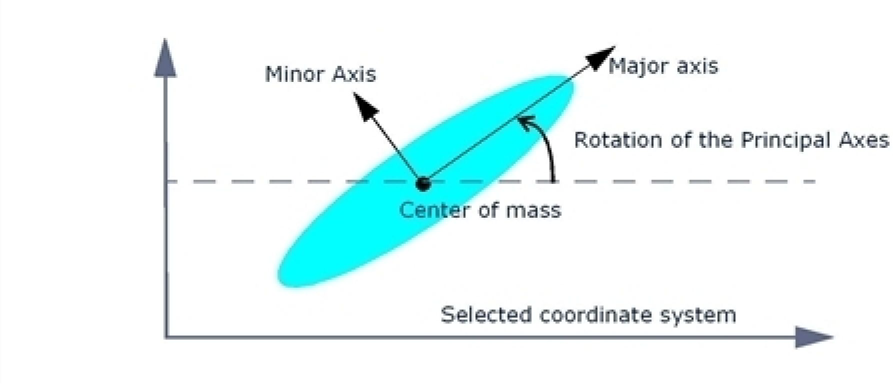


Figure: Illustration of Extrinsic Inertia Results

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1604, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1604, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1604, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1604, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The extrinsic inertia properties will be computed
0	No extrinsic inertia properties will be computed

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1604: the value used to reference this property.
index_id	N/A
object_id	N/A

FilterCount

VPARAMETER
5601

Number of filters applied by the tool. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5601, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5601, index_id, object_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5601: the value used to reference this property.
index_id	N/A
object_id	N/A

FilterHalfWidth

VPARAMETER
5202

Half-width of the convolution filter used by the tool to compute an edge-magnitude curve from which edges are detected. This value should be set to a value approximately equivalent to the width of the edge, in pixels, as it appears in the image.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5202, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5202, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5202, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5202, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 25

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5202: the value used to reference this property.
index_id	N/A
object_id	N/A

FilterHueTolerance

VPARAMETER
5716

For the selected filter, the value of the tolerance allowed for the Hue value defined by FilterHueValue. The FilterHueTolerance value is distributed equally above and below the FilterHueValue. For example, if FilterLuminanceValue = 200 and FilterHueTolerance = 20, the filter will accept pixels with a range of hue values = [190,200].

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5716, filter_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5716, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5716, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5716, filter_id, object_id)
```

Remarks

When **FilterHueTolerance** = 1, no tolerance (variation) in luminance is accepted. The filter will only accept pixels with a luminance value equal to FilterHueValue.

Type

Long

Range

Minimum: 1

Maximum: 128

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5716: the value used to reference this property.
filter_id	Index of the filter to which the value applies First Filter is 0.
object_id	N/A

FilterHueValue

VPARAMETER
5713

Value of the Hue component, in the HSL colorspace, for the selected filter. This value may be modified if any changes are made to the RGB values of the filter. Hue is the quality of color that is perceived as the color itself. It is commonly expressed by the color name, for example: red, green, yellow. Hue is determined by the perceived dominant wavelength, or the central tendency of combined wavelengths, within the visible spectrum.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5713, filter_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5713, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5713, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5713, filter_id, object_id)
```

Remarks

The value of a filter can be configured either by its HSL values or its RGB values. The Tolerance in a color filter can only be expressed in HSL values.

HSL values are defined by properties: FilterHueValue, FilterLuminanceValue, and FilterSaturationValue.

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5713: the value used to reference this property.
filter_id	Index of the filter to which this value applies.
object_id	N/A

FilteringClippingMode

VPARAMETER
5370

Sets the clipping mode applied by a filtering operation. Typically, the hsClippingAbsolute mode is used for filter operations.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5370, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5370, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5370, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5370, index_id, object_id)
```

Remarks

hsClippingNormal mode forces the destination pixel value to a value from 0 to 255 for unsigned 8-bit images, to a value from -327678 to 32767 for signed 16 bits images, and so on. Values that are less than the specified minimum value are set to the minimum value. Values greater than the specified maximum value are set to the maximum value.

hsClippingAbsolute mode takes the absolute value of the result and clips it using the same algorithm used for hsClippingNormal mode.

Range

Value	Image Processing Clipping Mode	Description
0	hsClippingNormal	Normal clipping method is used.
1	hsClippingAbsolute	Absolute clipping method is used.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5370: the value used to reference this property.
index_id	N/A
object_id	N/A

FilteringKernelSize

VPARAMETER
5371

Kernel size applied by a fixed (predefined) filtering operation.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5371, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5371, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5371, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5371, index_id, object_id)
```

Type

Long

Range

Valid sizes are 3,5,7

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5371: the value used to reference this property.
index_id	N/A
object_id	N/A

FilteringScale

VPARAMETER
5372

Scaling factor applied by a filtering operation. After the operation has been applied, the value of each pixel is multiplied by the FilteringScale value.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5372, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5372, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5372, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5372, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5372: the value used to reference this property.
index_id	N/A
object_id	N/A

FilterLuminanceTolerance

VPARAMETER
5718

Value of the tolerance allowed for the Luminance value, defined by FilterLuminanceValue, for the selected filter. The FilterLuminanceTolerance value is distributed equally above and below the FilterLuminanceValue. For example, if FilterLuminanceValue = 200 and FilterLuminanceTolerance = 20, the filter will accept pixels within a range of luminance values = [190,200].

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5718, filter_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5718, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5718, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5718, filter_id, object_id)
```

Remarks

When FilterLuminanceTolerance = 1, no tolerance (variation) in luminance is accepted. The filter will only accept pixels with a luminance value equal to FilterLuminanceValue.

Type

Long

Range

Minimum: 1

Maximum: 128

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5718: the value used to reference this property.
filter_id	Index of the filter to which this value applies.
object_id	N/A

FilterLuminanceValue

VPARAMETER
5715

Value of the Luminance component, in the HSL colorspace, for the selected filter. This value may be modified if any changes are made to the RGB values of the filter. Luminance is perceived as the brightness of the color or the amount of white contained in the color. When FilterLuminanceValue = 0, the color is completely black (RGB= 0,0,0). When FilterLuminanceValue = 255, the color is almost completely white.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5715, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5715, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5715, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5715, filter_id, object_id)
```

Remarks

The value of a filter can be configured either by its HSL values or its RGB values. The Tolerance in a color filter can only be expressed in HSL values.

HSL values are defined by properties: FilterHueValue, FilterLuminanceValue, and FilterSaturationValue.

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5715: the value used to reference this property.
filter_id	Index of the filter to which this value applies.
object_id	N/A

FilterSaturationTolerance

VPARAMETER
5717

Value of the tolerance allowed for the saturation value, defined by FilterSaturationValue, for the selected filter. The FilterSaturationTolerance value is distributed equally above and below the FilterSaturationValue.
For example, if FilterSaturationValue = 200 and FilterSaturationTolerance = 20, the filter will accept pixels with a range of saturation values = [190,200].

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5717, filter_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5717, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5717, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5717, filter_id, object_id)
```

Remarks

When FilterSaturationTolerance = 1, no tolerance (variation) in saturation is accepted. The filter will only accept pixels with a saturation value equal to FilterSaturationValue.

Type

Long

Range

Minimum: 1

Maximum: 128

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5717: the value used to reference this property.
filter_id	Index of the filter to which this value applies.
object_id	N/A

FilterSaturationValue

VPARAMETER
5714

Value of the Saturation component, in the HSL colorspace, for the selected filter. This value may be modified if any changes are made to the RGB values of the filter. Saturation is perceived as the amount of purity of the color or of the amount of grey in a color. When FilterSaturationValue = 0, the color appears as mid-grey (RGB = 112,126,126). When FilterSaturationValue = 255, the color is said to be saturated.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5714, filter_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5714, filter_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5714, filter_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5714, filter_id, object_id)
```

Remarks

The value of a filter can be configured either by its HSL values or its RGB values. The Tolerance in a color filter can only be expressed in HSL values.

HSL values are defined by properties: FilterHueValue, FilterLuminanceValue, and FilterSaturationValue.

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5714: the value used to reference this property.
filter_id	Index of the filter to which this value applies.
object_id	N/A

FitMode

VPARAMETER 5140

Specifies the mode used by the tool to calculate and return values for the found arc.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5140, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5140, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5140, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5140, index_id, object_id)
```

Type

Long

Range

Value	hsFitMode	Description
0	hsBoth	The Arc Finder calculates and returns both the arc center and arc radius.
1	hsRadius	The arc radius is calculated, the arc center returned is the value of the tool center.
2	hsCenter	The arc center is calculated; the radius returned is the value of the tool radius

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5140: the value used to reference this property.
index_id	N/A
object_id	N/A

FitQuality

VRESULT
1803

Normalized average error between the calculated arc or line entity and the actual edges matched to the found entity. Fit quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that the average error is 0. Conversely, a value of 0 means that the average matched error is equal to Conformity Tolerance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1803, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1803, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1803: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederVRunCommand

VPARAMETER 7000

Identifies the command that is run when a VRUN is issued to the Flexibowl Feeder.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7000, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7000, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7000, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7000, index_id, object_id)
```

Type

Real variable.

Range

A valid Flexibowl Feeder command or sequence index. The valid Flexibowl Feeder commands are:

Blow	1
Forward	2
Shake	3
Flip 1	4
Flip 2	5
Flip Blow	6
Forward Blow	7
Forward Flip Blow	8
Light On	15
Light Off	16

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7000: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederBlowTime

VPARAMETER 7012

The length of time (in ms) for the blow operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7012, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7012, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7012, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7012, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: Unbounded

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7012: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederFlipCount

VPARAMETER 7005

The number of iterations used in the flip operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7005, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7005, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7005, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7005, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: Unbounded

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7005: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederFlipDelay

VPARAMETER 7006

The amount of time (in ms) to delay in the flip operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7006, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7006, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7006, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7006, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: Unbounded

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7006: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederForwardAcceleration

VPARAMETER 7001

The acceleration for the forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7001, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7001, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7001, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7001, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 10

Maximum: 10000

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7001: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederForwardAngle

VPARAMETER
7003

The angle used in the forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7003, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7003, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7003, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7003, index_id, object_id)
```

Type

Real variable.

Range

Minimum: -360

Maximum: 360

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7003: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederForwardDeceleration

VPARAMETER 7002

The deceleration for the forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7002, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7002, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7002, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7002, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 10

Maximum: 10000

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7002: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederForwardSpeed

VPARAMETER 7004

The speed for the forward operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7004, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7004, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7004, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7004, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 130

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7004: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederShakeAcceleration

VPARAMETER 7007

The acceleration for the shake operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7007, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7007, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7007, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7007, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 10

Maximum: 10000

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7007: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederShakeAngle

VPARAMETER 7009

The angle for the shake operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7009, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7009, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7009, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7009, index_id, object_id)
```

Type

Real variable.

Range

Minimum: -360

Maximum: 360

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7009: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederShakeCount

VPARAMETER 7011

The number of iterations used in the shake operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7011, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7011, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7011, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7011, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: Unbounded

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7011: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederShakeDeceleration

VPARAMETER 7008

The deceleration for the shake operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7008, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7008, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7008, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7008, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 10

Maximum: 10000

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7008: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FlexibowlFeederShakeSpeed

VPARAMETER 7010

The speed for the shake operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 7010, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 7010, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 7010, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 7010, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 1

Maximum: 130

Parameters

sequence_id	Index associated with the feeder as defined in the feeder editor.
tool_id	N/A
ID	7010: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Found

VRESULT
1800

Specifies if an entity was found. If True, at least one entity (point, line or arc) was found in the current image. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1800, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1800, index_id, frame_id)
```

Type

Long

Range

Value	State	Description
0	False	No entity was found.
1	True	An entity was found.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1800: the value used to reference this property.
index_id	N/A
frame_id	N/A

FrameCount

VRESULT
2410

Uses the frame index to return the number of results relative to the specified frame. This property is read-only.

NOTE: This ACE Sight property is interchangeable with InstanceCount.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2410, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2410, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	N/A
ID	2410: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Related Topics

InstanceCount
ResultCountInstanceCount

FrameIntrinsicBoundingBox

VRESULT
2420

Sets the coordinates of the instrinsic bounding box that defines a frame. The intrinsic bounding box is the smallest box that can enclose the frame. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool, instance_id, 2420, bounding_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool, instance_id, 2420, bounding_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2420: the value used to reference this property.
bounding_id	1 to 8: Index of the X/Y-coordinates that define corners of the intrinsic bounding box: 1: X-coordinate of the corner 2: Y-coordinate of the corner 3: X-coordinate of the corner 4: Y-coordinate of the corner 5: X-coordinate of the corner

	6: Y-coordinate of the corner 7: X-coordinate of the corner 8: Y-coordinate of the corner
frame_id	Index of the frame for which you want to retrieve the result contained in the specified instance.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

FrameRotation

VRESULT 2402

Rotation of the specified output frame. It does not include a tool offset or camera calibration offset. This property is read-only.

NOTE: InstanceRotation is the preferred property. Therefore, you should update your code to use that property.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2402, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2402, index_id, frame_id)
```

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2402: the value used to reference this property.
index_id	Index of the frame for which you want to set the mode.
frame_id	N/A

FrameTranslationX

VRESULT
2400

X-coordinate of the origin of the specified output frame. It does not include a tool offset or camera calibration offset. This property is read-only.

NOTE: InstanceTranslationX is the preferred property. Therefore, you should update your code to use that property.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2400, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2400, index_id, frame_id)
```

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2400: the value used to reference this property.
index_id	Index of the frame for which you want to set the mode.
frame_id	N/A

FrameTranslationY

VRESULT
2401

Y-coordinate of the origin of the specified output frame. It does not include a tool offset or camera calibration offset. This property is read-only.

NOTE: InstanceTranslationY is the preferred property. Therefore, you should update your code to use that property.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2401, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2401, index_id, frame_id)
```

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2401: the value used to reference this property.
index_id	Index of the frame for which you want to set the mode.
frame_id	N/A

GreylevelRange

VRESULT
1508

Range of greylevel values of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. The range is equal to [MaximumGreylevelValue - MinimumGreylevelValue + 1]. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1508, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1508, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 256

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1508: the value used to reference this property.
index_id	The index of the frame for which you want to set the mode.
frame_id	N/A

GreyLevelResultsEnabled

VPARAMETER
1608

Enables the computation of the following blob greylevel properties: BlobGreyLevelMaximum, BlobGreyLevelMean, BlobGreyLevelMaximum, BlobGreyLevelMinimum, BlobGreyLevelRange, and BlobGreyLevelStdDev.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1608, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1608, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1608, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1608, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The greylevel blob properties will be computed
0	No greylevel blob properties will be computed

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1608: the value used to reference this property.

index_id	N/A
object_id	N/A

GripperInputClose

VPARAMETER
5515

Returns the close input signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5515, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5515, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5515: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperInputExtend

VPARAMETER 5518

Returns the extend input signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5518, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5518, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5518: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperInputOpen

VPARAMETER 5514

Returns the open input signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5514, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5514, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5514: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperInputRetract

VPARAMETER
5519

Returns the retract input signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5519, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5519, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5519: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperOffset

VLOCATION
10100

Allows an application program to extract the gripper offsets for the tips associated with a robot in the workspace. The "instance" number is used to identify the robot number to access. The result index is used to specify the tip to return.

Syntax

MicroV+

```
VLOCATION (sequence_id, tool_id, instance_id, 10100, index_id, frame_id)
```

V+

```
VLOCATION ($ip, sequence_id, tool_id, instance_id, 10100, index_id, frame_id)
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	Should always be set to -1.
instance_id	The robot number to access.
ID	10100: the value used to reference this property.
index_id	Index of the tip to return.
frame_id	N/A

GripperOutputClose

VPARAMETER
5512

Returns the close output signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5512, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5512, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5512: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperOutputExtend

VPARAMETER 5516

Returns the extend output signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5516, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5516, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5516: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperOutputOpen

VPARAMETER
5511

Returns the open output signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5511, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5511, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5511: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperOutputRelease

VPARAMETER
5513

Returns the release signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5513, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5513, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5513: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperOutputRetract

VPARAMETER
5517

Returns the retract output signal for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5517, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5517, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The tip number.
ID	5517: the value used to reference this property.
index_id	The robot number.
object_id	The index of the signal number to access.

GripperPayload

VPARAMETER
5550

Allows an application program to read the payload associated with a gripper associated with a robot in the workspace. The "instance" number is used to identify the robot number to access.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 5550, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 5550, index_id, object_id)
```

Type

Real variable.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	
ID	5550: the value used to reference this property.
index_id	The robot number.
object_id	

GripperToolTransform

VLOCATION
11000

Returns the tool transformation for a given tip on the gripper. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 11000, result_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 11000, result_id, frame_id)
```

Type

Transformation.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	Index of the tip number to access.
instance_id	The robot number.
ID	11000: the value used to reference this property.
result_id	Not used.
frame_id	Not used.

Histogram

VRESULT
1511

Histogram of greylevel values of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. The histogram comprises 256 bins. One histogram bin is associated with each of the 256 possible greylevel values. It contains the number of pixels with the corresponding greylevel value in the region of interest. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1511, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1511, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1511: the value used to reference this property.
index_id	N/A
frame_id	N/A

HistogramPixelCount

VRESULT
1512

Total number of pixels in the histogram. The number of pixels in the histogram is equal to ImagePixelCount minus the pixels excluded from the Histogram by any threshold or tail functions set by the properties: ThresholdBlack, ThresholdWhite, TailWhite, or TailBlack. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1512, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1512, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1512: the value used to reference this property.
index_id	N/A
frame_id	N/A

HistogramThreshold

VPARAMETER 5385

Threshold value applied by a histogram thresholding operation.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5385, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5385, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5385, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5385, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5385: the value used to reference this property.
index_id	N/A
object_id	Index of the frame containing the edge pair.

HoleFillingEnabled

VPARAMETER
5002

Enables the filling of the holes in each blob.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5002, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5002, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5002, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5002, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	All holes will be filled.
0	No hole will be filled.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5002: the value used to reference this property.
index_id	N/A
object_id	N/A

ImageHeight

VRESULT
1021

Height, in pixels, of the tool region of interest. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1021, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1021, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1021: the value used to reference this property.
index_id	N/A
frame_id	N/A

ImageOriginBelt

VLOCATION
10053

Origin of the image frame of reference. Expressed as a transform relative to the robot frame of reference. This property is read-only.

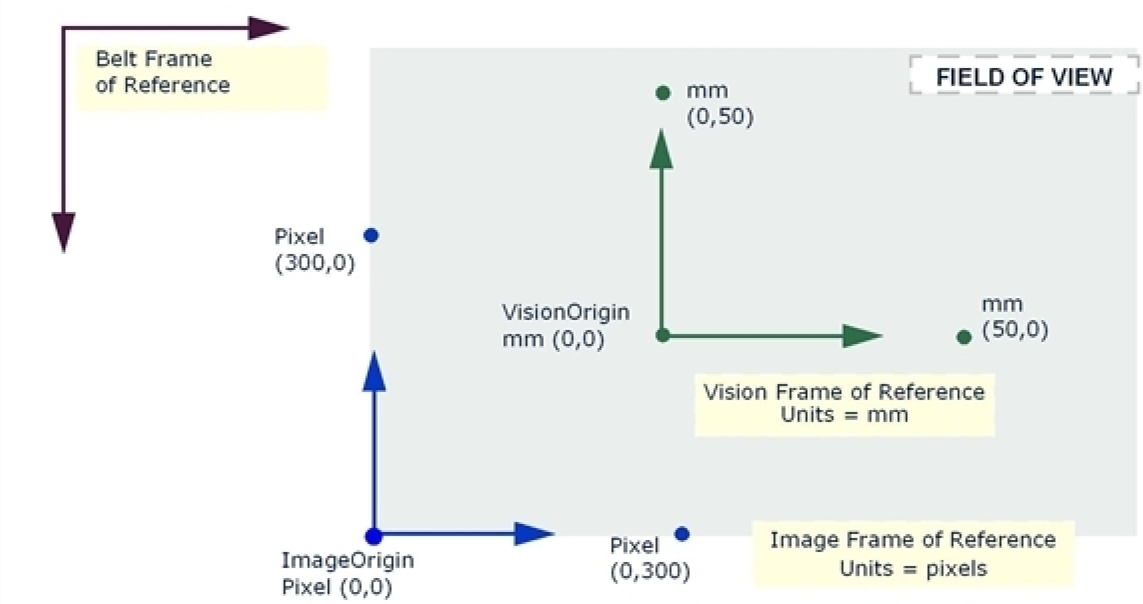


Figure: Illustration of ImageOrigin and VisionOrigin Properties

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10053, index_id, frame_id)
```

MicroV+

```
value = Not applicable. Conveyor tracking is supported only in V+.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10053: the value used to reference this property.
index	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

Related Properties

ImageOriginRobot
VisionOriginBelt
VisionOriginRobot

ImageOriginRobot

VLOCATION
10051

Origin of the image frame of reference. Expressed as a transform relative to the robot frame of reference. This property is read-only.

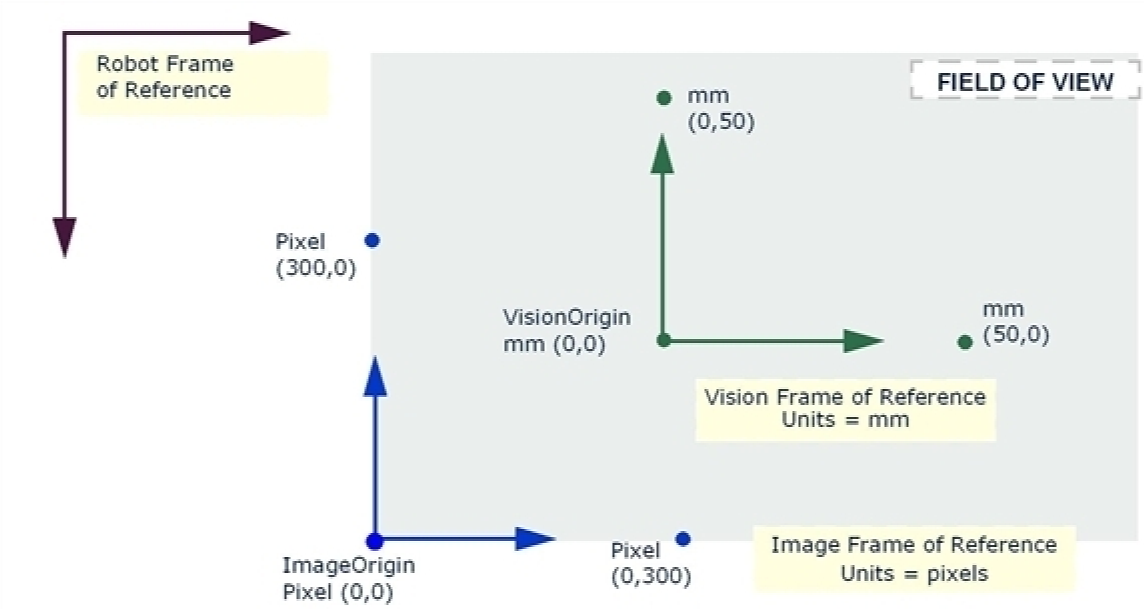


Figure: Illustration of ImageOrigin and VisionOrigin Properties

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 10051, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10051, index_id, frame_id)
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10051: the value used to reference this property.
index	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

Related Properties

ImageOriginBelt
VisionOriginBelt
VisionOriginRobot

ImagePixelCount

VRESULT
1513

Number of pixels in the tool region of interest. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1513, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1513, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1513: the value used to reference this property.
index_id	N/A
frame_id	N/A

ImageSubsampling

VPARAMETER

5324

Factor used to subsample the greyscale image in the tool region of interest. With a subsampling factor of 1, the greyscale image is not subsampled. With a subsampling factor of 2, the greyscale image is subsampled in tiles of 2x2 pixels. With a subsampling factor of 3 the greyscale image is subsampled in tiles of 3x3 pixels, and so on.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5324, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5324, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5324, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5324, index_id, object_id)
```

Remarks

Color Matching Tool

Increasing the subsampling level reduces the number of pixels and the quantity information analyzed by the tool. Increasing the Image Subsampling may reduce the execution time but affects the accuracy of color matching results.

Image Histogram

Using a higher subsampling factor speeds up the generation of the histogram but slightly reduces the accuracy of the statistics computed from the histogram. The pixel properties computed by the Image Histogram tool are normalized with respect to the subsampling factor (HistogramPixelCount, ImageHeight, ImagePixelCount and ImageWidth). Therefore, the total number of pixels in the histogram should remain the same at any subsampling factor. Note that there might be slight differences in the values of these properties when either of the width or the height of the region of interest is not a multiple of the subsampling factor used.

Type

Long

Range

1 (no subsampling),2, 3, 4, 5, 6, 7, 8

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5324: the value used to reference this property.
index_id	N/A
object_id	N/A

ImageWidth

VRESULT
1020

Width, in pixels, of the tool region of interest. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1020, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1020, index_id, frame_id)
```

Type

Long

Range

Greater than or equal to 0.

Remarks

Instance is relative for the Image Histogram tool. Note that it is:

- Not relative for the Color Matching tool.
- Not implemented for virtual cameras.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1020: the value used to reference this property.

index_id	N/A
frame_id	N/A

InspectionFilterMeasuredValue

VRESULT 2700

Returns the measured value of the specified filter for the instance. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2700, index_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2700, index_id)
```

Type

Long

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2700: the value used to reference this property.
index_id	Index of the filter for which you want the result.

InspectionFilterPassStatus

VRESULT 2702

Returns the pass status of the specified filter for the instance. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2702, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2702, index_id, frame_id)
```

Type

Long

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	2702: the value used to reference this property.
index_id	Index of the filter for which you want the result.
frame_id	Index of the category you wish to access. Range: 1, Category Count + 1 Where Category Count + 1 = Unassigned.

InstanceClearQuality

VRESULT
1319

Measure of the unencumbered area surrounding the specified object instance. Clear quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that the instance is completely free of obstacles. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1319, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1319, index_id, frame_id)
```

Type

Double

Remarks

In MicroV+/V+, the frame_id parameter is required.

Range

Minimum: 0.0

Maximum: 1.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1319: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance.

	Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceCount

VRESULT
1310

Uses the frame index to return the number of results relative to the specified frame. This property is read-only.

NOTE: This ACE Sight property is interchangeable with FrameCount.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1310, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1310, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Long

Range

Greater than or equal to 0.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	N/A
ID	1310: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance.

	Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Related Topics

FrameCount
ResultCount

InstanceFitQuality

VRESULT
1317

Normalized average error between the matched model contours of the selected object instance and the actual contours detected in the input image. Fit quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that the average error is 0. Conversely, a value of 0 means that the average matched error is equal to ConformityTolerance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1317, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1317, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1317: the value used to reference this property.
index_id	N/A

frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceIntrinsicBoundingBox

VRESULT
1330

Returns the coordinates of the instrinsic bounding box that defines an instance. The intrinsic bounding box is the smallest box that can enclose the instance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1330, bounding_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1330, bounding_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1330: the value used to reference this property.
bounding_id	1 to 8: Index of the XY-coordinates that define corners of the intrinsic bounding box: 1: X-coordinate of the corner 2: Y-coordinate of the corner 3: X-coordinate of the corner 4: Y-coordinate of the corner 5: X-coordinate of the corner

	6: Y-coordinate of the corner 7: X-coordinate of the corner 8: Y-coordinate of the corner
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceLocation

VLOCATION
1311

Returns the location of the selected instance in the frame of reference of the specified robot. If a gripper offset has been assigned to the instance, it is automatically applied to the location. If no robot-to-vision calibration has been applied, InstanceLocation returns the location in the vision frame of reference. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1311, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1311, index_id, frame_id)
```

Remarks

If there is a single gripper offset, **InstanceLocation** (1311) is the same as InstanceLocationGripperOffsetMinimum (1400). If there are multiple gripper offsets that can be applied to the instance, you should use InstanceLocationGripperOffsetMinimum = 1400 for the location with the first gripper offset, InstanceLocationGripperOffsetMinimum = 1401 for the location with the second gripper offset, and so on, for additional gripper offsets.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which the location is required.
ID	1311: the value used to reference this property.
index_id	Index of the robot.
frame_id	Index of the frame in which the instance is found. Typically this is '0' (i.e. the Locator is not frame-based).

Related Properties

InstanceRobotLocation

InstanceLocationGripperOffsetMaximum

InstanceLocationGripperOffsetMinimum

InstanceLocationGripperOffsetMaximum

VLOCATION
1499

Returns the maximum number of gripper offsets. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1499, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1499, index_id, frame_id)
```

Type

Location

Remarks

If there is a single gripper offset, InstanceLocation (1311) is the same as InstanceLocationGripperOffsetMinimum (1400). If there are multiple gripper offsets that can be applied to the instance you should use InstanceLocationGripperOffsetMinimum = 1400 for the location with the first gripper offset, InstanceLocationGripperOffsetMinimum = 1401 for the location with the second gripper offset, and so forth, for additional gripper offsets.

Range

Minimum: Greater than or equal to InstanceLocationGripperOffsetMinimum

Maximum: 100

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1499: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Related Properties

InstanceLocation

InstanceLocationGripperOffsetMinimum

InstanceLocationGripperOffsetMinimum

VLOCATION
1400

Returns the minimum number of gripper offsets. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1400, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1400, index_id, frame_id)
```

Type

Location

Remarks

If there is a single gripper offset, InstanceLocation (1311) is the same as InstanceLocationGripperOffsetMinimum (1400). If there are multiple gripper offsets that can be applied to the instance, you should use InstanceLocationGripperOffsetMinimum = 1400 for the location with the first gripper offset, InstanceLocationGripperOffsetMinimum = 1401 for the location with the second gripper offset, and so on, for additional gripper offsets.

Range

Minimum: Greater than or equal to 0

Maximum: Greater than or equal to InstanceLocationGripperOffsetMaximum

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1400: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Related Properties

InstanceLocation

InstanceLocationGripperOffsetMaximum

InstanceMatchQuality

VRESULT
1318

Returns a value representing the percent of matched model contours for the selected object instance. Match quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that 100% of the model contours were successfully matched to the actual contours detected in the input image. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1318, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1318, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1318: the value used to reference this property.
index_id	N/A

frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceModel

VRESULT
1312

Returns the index of the model associated with the selected object instance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1312, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1312, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Long

Range

Minimum: 0

Maximum: Number of models - 1

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1312: the value used to reference this property.
index_id	N/A

frame	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceOrdering

VPARAMETER
530

Order in which the instances are processed and output.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 530, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 530, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 530, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 530, index_id, object_id)
```

Remarks

With the hsDistanceImage and hsDistanceWorld modes, the reference coordinate used to compute the distance is set with the InstanceOrderingReferenceX and InstanceOrderingReferenceY properties.

Type

Long

Range

Value	Mode Name	Description
0	hsEvidence	Instances are processed and output according to their hypothesis strength, beginning with the strongest hypothesis.
1	hsLeftToRight	Instances are processed and output in the order they appear in the search area, from left to right.
2	hsRightToLeft	Instances are processed and output in the order they appear in the search area, from right to left.
3	hsTopToBottom	Instances are processed and output in the order they appear in the search area, from top to bottom.

Value	Mode Name	Description
4	hsBottomToTop	Instances are processed and output in the order they appear in the search area, from bottom to top.
5	hsQuality	All the instances are first processed and then they are output according to their Quality, beginning with the highest quality.
6	hsDistanceImage	Instances are processed and output according to their distance from a reference image coordinate, beginning with the closest.
7	hsDistanceWorld	Instances are processed and output according to their distance from a reference world coordinate, beginning with the closest.
8	hsShadingConsistency	Instances are processed and output according to their shading consistency with respect to the model, beginning with the strongest hypothesis.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	530: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceOrderingReferenceX

VPARAMETER
531

Reference X-coordinate used to compute the distance when the hsDistanceImage or hsDistanceWorld ordering mode is enabled through the InstanceOrdering property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 531, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 531, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 531, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 531, index_id, object_id)
```

Type

Double

Range

Not applicable.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	531: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceOrderingReferenceY

VPARAMETER
532

Reference Y-coordinate used to compute the distance when the hsDistanceImage or hsDistanceWorld ordering mode is enabled through the InstanceOrdering property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 532, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 532, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 532, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 532, index_id, object_id)
```

Type

Double

Range

Not applicable.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	532: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceRobotLocation

VLOCATION
1371

Returns the location of the selected instance in the frame of reference for the specified robot. No offset transformations are applied to the location. If a gripper offset has been assigned to the instance, it is ignored. If no vision-to-robot calibration has been applied, the system returns an error. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1371, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1371, index_id, frame_id)
```

Type

Location

Remarks

This differs from InstanceLocation, which applies any calculated offset and returns the vision frame of reference coordinates if there is no robot-to-vision calibration.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which the location is required.
ID	1371: the value used to reference this property.
index_id	Index of the robot.
frame_id	Index of the frame in which the instance is found. Typically this is '0' (i.e. the Locator is not frame-based).

Related Properties

InstanceLocation

InstanceLocationGripperOffsetMaximum

InstanceLocationGripperOffsetMinimum

InstanceRotation

VRESULT
1314

Angle of rotation of the Object coordinate system of the selected object instance. It does not include a tool offset or camera-calibration offset. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1314, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1314, index_id, frame_id)
```

Remarks

When the NominalRotationEnabled property is True, the rotation of the object instance is always equal to NominalRotation.

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1314: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceScaleFactor

VRESULT
1313

Scale factor of the selected object instance based on its size relative to the associated model. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1313, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1313, index_id, frame_id)
```

Remarks

When the NominalScaleFactorEnabled property is True, the scale factor of the object instance is always equal to NominalScaleFactor.

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Minimum: MinimumScaleFactor or NominalScaleFactor

Maximum: MaximumScaleFactor or NominalScaleFactor

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.

ID	1313: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceSymmetry

VRESULT
1320

Index of the object instance that is symmetrical to the selected object instance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1320, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1320, index_id, frame_id)
```

Remarks

If OutputSymmetricInstances is set to False, InstanceSymmetry is always equal to the instance's index.

In MicroV+/V+, the frame_id parameter is required.

Type

Long

Range

Minimum: 0

Maximum: InstanceCount -1

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.

ID	1320: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceTime

VRESULT
1322

Time, in milliseconds, needed to recognize and locate the selected object instance. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1322, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1322, index_id, frame_id)
```

Remarks

The time needed to locate the first object instance is usually longer because it includes all low-level image preprocessing.

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.

ID	1322: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceToolOffset

VLOCATION
1372

Returns the location of the selected instance relative to the position of the robot at the last picture position. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1372, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1372, index_id, frame_id)
```

Remarks

The time needed to locate the first object instance is usually longer because it includes all low-level image preprocessing.

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.

ID	1372: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceTranslationX

VRESULT
1315

X-translation of the Object coordinate system for the selected object instance. It does not include a tool offset or camera-calibration offset. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1315, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1315, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1315: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceTranslationY

VRESULT
1316

Y-translation of the Object coordinate system for the selected object instance. It does not include a tool offset or camera-calibration offset. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1316, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1316, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1316: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

InstanceVisible

VRESULT
1321

Returns the percentage of the instance bounding box that was found in the image. If the entire instance bounding box is in the field of view, the percentage is 100; if it is partially outside the field of view, the percentage is less than 100. This property is read-only.

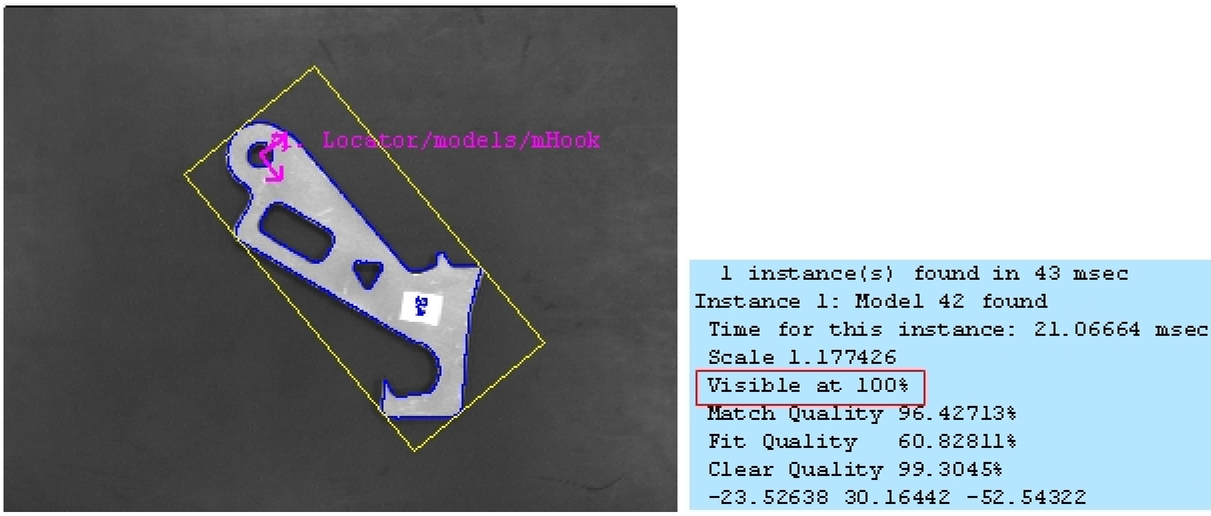


Figure: Instance Visible at 100%

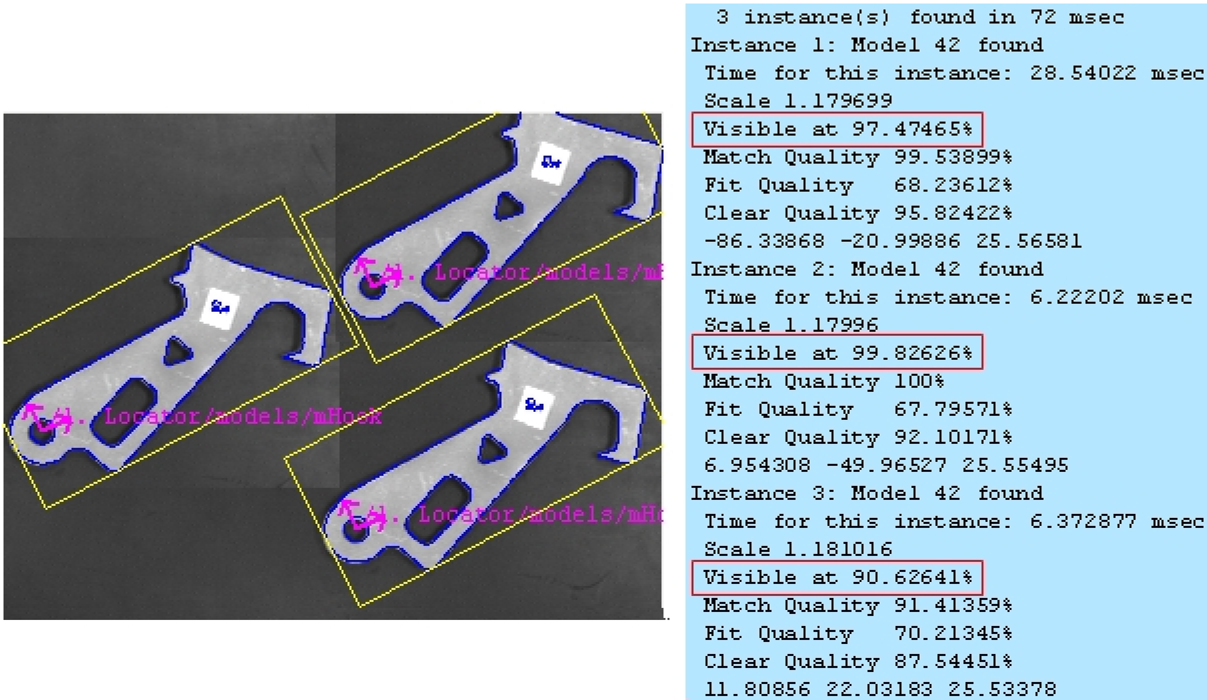


Figure: Instance Visible at less than 100%

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1321, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1321, index_id, frame_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1321: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

InstanceVisionOffset

VLOCATION
1373

Returns the vision coordinates of a located instance. This property is read-only.

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 1373, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 1373, index_id, frame_id)
```

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1373: the value used to reference this property.
index_id	Index of the robot.
frame_id	Index of the frame that contains the specified instance.

InterpolatePositionMode

VPARAMETER
5122

Sets the mode used by the Point Finder tool to compute a point hypothesis

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5122, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5122, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5122, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5122, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
0	hsCorner	The tool will compute a hypothesis that fits a corner point to interpolated lines from connected edges.
1	hsIntersection	The tool will compute a hypothesis that is an intersection between the search axis and connected edges of an interpolated line.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5122: the value used to reference this property.
index_id	N/A
object_id	N/A

InterpolatePositionModeEnabled

VPARAMETER 5123

When InterpolatePositionModeEnabled is set to True, the Point Finder tool uses the value set by the InterpolatePositionMode property to compute a point hypothesis. Otherwise, point hypothesis coordinates are taken directly from a specific found edge that satisfies search constraints.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5123, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5123, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5123, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5123, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
1		The Point Finder tool uses the value set by the InterpolatePositionMode property to compute a point hypothesis
0		The Point Finder tool calculates point hypothesis directly from a specific found edge that satisfies search constraints.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5123: the value used to reference this property.
index_id	N/A
object_id	N/A

IntrinsicBoxResultsEnabled

VPARAMETER
1605

Enables the computation of the following intrinsic bounding boxes and intrinsic extents: BlobIntrinsicBoundingBoxBottom, BlobIntrinsicBoundingBoxCenterX, BlobIntrinsicBoundingBoxCenterY, BlobIntrinsicBoundingBoxHeight, BlobIntrinsicBoundingBoxLeft, BlobIntrinsicBoundingBoxRight, BlobIntrinsicBoundingBoxRotation, BlobIntrinsicBoundingBoxTop, BlobIntrinsicBoundingBoxWidth, BlobIntrinsicExtentBottom, BlobIntrinsicExtentLeft, BlobIntrinsicExtentRight and BlobIntrinsicExtentTop.

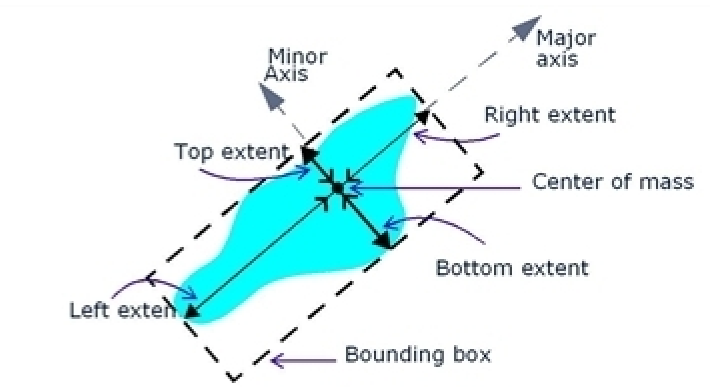


Figure: Illustration of Intrinsic Box Results

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1605, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1605, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1605, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1605, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The intrinsic box properties will be computed
0	No intrinsic box properties will be computed

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1605: the value used to reference this property.
index_id	N/A
object_id	N/A

InverseKinematics

VLOCATION
10060

For a robot with a tool-mounted or an arm-mounted camera, InverseKinematics retrieves the location to which to move the robot so the camera sees a specific point in the workspace (robot frame of reference) at a specific point in the image (image frame of reference). The X/Y-coordinates of the point in the workspace are defined by RobotXPosition and RobotYPosition, and the X/Y-coordinates of the point in the image are defined by VisionXPosition and VisionYPosition.

If the camera is arm-mounted, the configuration used in the kinematic calculation is based on the current arm configuration of the robot If the camera is tool-mounted, there are an infinite number of solutions for positioning the robot. Therefore, using the VisionRotation property, you must specify the angle of rotation between the vision X-axis and the robot X-axis.

NOTE: Inverse kinematics calculations for the Cobra i-series (i600 and i800) robots are not supported.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, instance_id, 10060, index_id, frame_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, instance_id, 10060, index_id, frame_id)
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Should always be set to -1.
tool_id	The camera number, as defined in Keyword Mapping parameter of the ACE Sight Camera Calibration (in the ACE workspace).
instance_id	Not used.

ID	10060: the value used to reference this property.
index	The robot number, as defined in Keyword Mapping parameter of the ACE Sight Camera Calibration (in the ACE workspace).
frame_id	Not used.

Example

This example illustrates the use of the following properties: InverseKinematics, RobotXPosition, RobotYPosition, VisionXPosition, VisionYPosition, and VisionRotation.

```
.PROGRAM demo()

        ; This program will move the robot so that a given point in the
        ; robot frame of reference can be seen in a given point in the vision
        ; Coordinate system (Calibrated)

        ; This defines the IP address of the PC
            $ip = "192.168.0.223"

        ; This defines the point in the robot coordinate system that should
        ; be visible in the camera
        robot_x = 300
        robot_y = 0

        ; This is the point where the robot point should be seen in the
        ; camera coordinate system. These units are mm (Calibrated Image).
        ; When they are set to (0,0), it means the center of the image.
        ; Vision_rot only applies for a ToolMountedCamera
        vision_x = 0
        vision_y = 0
        vision_rot = 0

        ; Tell ACE Sight what are the chosen values
        ; for configuration and vision points.
        VPARAMETER(-1, 1, 10401, 1) $ip = vision_x
        VPARAMETER(-1, 1, 10402, 1) $ip = vision_y
        VPARAMETER(-1, 1, 10403, 1) $ip = vision_rot

        WHILE TRUE DO

            ; Tell ACE Sight what are the chosen values for robot point.
            VPARAMETER(-1, 1, 10404, 1) $ip = robot_x
            VPARAMETER(-1, 1, 10405, 1) $ip = robot_y

            ; Ask ACE Sight where to move the robot in order to make
            ; robot point seen in vision point
            SET loc = VLOCATION($ip, -1, 1, , 10060, 1)

            ; Move to the position
            MOVES loc
```

```
            BREAK
        END
.END
```

Related Properties

RobotXPosition
RobotYPosition
VisionXPosition
VisionYPosition
VisionRotation

KernelSize

VPARAMETER
5304

Sets the size of the kernel of the operator for the sharpness process. The default setting of 5 (for a 5X5 kernel) is generally sufficient for most cases. This property is read-only.

Syntax

Micro V+

```
VRESULT (sequence_id, tool_id, instance_id, 5304, index_id, frame_id) = value
value = VRESULT (sequence_id, tool_id, instance_id, 5304, index_id, frame_id)
```

V+

```
VRESULT (sequence_id, tool_id, instance_id, 5304, index_id, frame_id) $ip = value
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 5304, index_id, frame_id)
```

Type

Long

Range

Minimum: 2

Maximum: 16

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pair for which you want the result.
ID	5304: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the pair.

LastOperation

VRESULT
2200

Operation applied by the Image Processing tool at the last iteration. This property is read-only.

Type

Long

Range

Value	Name	Description
0	hsArithmeticAddition	Operand value (constant or Operand Image pixel) is added to the corresponding pixel in the input image.
1	hsArithmeticSubtraction	Operand value (constant or Operand Image pixel) is subtracted from the corresponding pixel in the input image.
2	hsArithmeticMultiplication	The input image pixel value is multiplied by the Operand value (constant or corresponding Operand Image pixel).
3	hsArithmeticDivision	The input image pixel value is divided by the Operand value (constant or corresponding Operand image pixel). The result is scaled and clipped, and finally written to the output image.
4	hsArithmeticLightest	The Operand value (constant or Operand Image pixel) and corresponding pixel in the input image are compared to find the maximal value.
5	hsArithmeticDarkest	The Operand value (constant or Operand Image pixel) and corresponding pixel in the input image are compared to find the minimal value.
6	hsAssignmentInitialization	All the pixels of the output image are set to a specific constant value. The height and width of the output image must be specified.
7	hsAssignmentCopy	Each input image pixel is copied to the corresponding output image pixel.
8	hsAssignmentInversion	The input image pixel value is inverted and the result is copied to the corresponding output image pixel.

Value	Name	Description
9	hsLogicalAnd	AND operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
10	hsLogicalNAnd	NAND operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
11	hsLogicalOr	OR operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
12	hsLogicalXOr	XOR operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
13	hsLogicalNOr	NOR operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
14	hsFilteringCustom	Applies a Custom filter.
15	hsFilteringAverage	Applies an Average filter.
16	hsFilteringLaplacian	Applies a Laplacian filter.
17	hsFilteringHorizontalSobel	Applies a Horizontal Sobel filter.
18	hsFilteringVerticalSobel	Applies a Vertical Sobel filter.
19	hsFilteringSharpen	Applies a Sharpen filter.
20	hsFilteringSharpenLow	Applies a SharpenLow filter.
21	hsFilteringHorizontalPrewitt	Applies a Horizontal Prewitt filter.
22	hsFilteringVerticalPrewitt	Applies a Vertical Prewitt filter.
23	hsFilteringGaussian	Applies Gaussian filter.
24	hsFilteringHighPass	Applies High Pass filter.
25	hsFilteringMedian	Applies a Median filter.
26	hsMorphologicalDilate	Sets each pixel in the output image as the largest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size.

Value	Name	Description
27	hsMorphologicalErode	Sets each pixel in the output image as the smallest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size.
28	hsMorphologicalClose	Has the effect of removing small dark particles and holes within objects.
29	hsMorphologicalOpen	Has the effect of removing peaks from an image, leaving only the image background.
30	hsHistogramEqualization	Equalization operation enhances the Input Image by flattening the histogram of the Input Image.
31	hsHistogramStretching	Stretches (increases) the contrast in an image by applying a simple piecewise linear intensity transformation based on the histogram of the Input Image.
32	hsHistogramLightThreshold	Changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the minimum acceptable value. Otherwise, it is set to the maximum presentable value.
33	hsHistogramDarkThreshold	Changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the maximum presentable value. Otherwise, it is set to the minimum acceptable value.
34	hsTransformFFT	Converts and outputs a frequency description of the input image by applying a Fast Fourier Transform (FFT).
35	hsTransformDCT	Converts and outputs a frequency description of the input image by applying a Discrete Cosine Transform (DCT).Parameters

LastOutputType

VRESULT
2201

Type of the image output by the Image Processing tool at the last iteration. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2201, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2201, index_id, frame_id)
```

Type

Long

Range

Value	Name	Description
1	hsType8Bits	Unsigned 8-bit image.
10	hsType16Bits	Signed 16-bit image.
7	hsType32Bits	Signed 32-bit image

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance in the specified result frame. If no result frame is specified, it is the index for all instances returned by the tool.
ID	2201: the value used to reference this property.

index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]

LogicalConstant

VPARAMETER
5380

Constant applied by a logical operation when no valid operand image is specified.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5380, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5380, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5380, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5380, index_id, object_id)
```

Type

Long

Range

Minimum: -32768

Maximum: 32767

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5380: the value used to reference this property.
index_id	N/A
object_id	N/A

MagnitudeConstraint

VPARAMETER
5226

Indexed property used to set the magnitude-constraint function for edge detection. Two points are used: Base and Top.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5226, index_id, constraint_id) = value
value = VPARAMETER (sequence_id, tool_id, 5226, index_id, constraint_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5226, index_id, constraint_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5226, index_id, constraint_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5226: the value used to reference this property.
index_id	N/A
constraint_id	One of the two points of the magnitude constraint function (hsMagnitudeConstraintIndex)

	1: Base point 2: Top point

MagnitudeThreshold

VPARAMETER
5200

Magnitude threshold sets the threshold used to find edges on the magnitude curve. In order to locate edges, a subpixel, peak-detection algorithm is applied on the region of every minimum or maximum of the curve that exceeds this threshold.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5200, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5200, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5200, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5200, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5200: the value used to reference this property.
index_id	N/A
object_id	N/A

MatchCount

VRESULT
2100

Number of matched patterns found. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2100, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2100, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pattern instance for which you want the result.
ID	2100: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame containing the pattern instance for which you want the result.

MatchPositionX

VRESULT
2102

X-coordinate of a matched pattern in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2102, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2102, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pattern instance for which you want the result.
ID	2102: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame containing the pattern instance for which you want the result.

MatchPositionY

VRESULT
2103

Y-coordinate of a matched pattern in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2103, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2103, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pattern instance for which you want the result.
ID	2103: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame containing the pattern instance for which you want the result.

MatchQuality

VRESULT
1802

Percentage of edges actually matched to the found entity (point, arc, or line). MatchQuality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that edges were matched for every point along the found entity. Similarly, a value of 0.2 means edges were matched to 20% of the points along the found entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1802, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1802, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1802: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MatchRotation

VRESULT
2104

Rotation of a matched pattern in the currently-selected coordinate system. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2104, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2104, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pattern instance for which you want the result.
ID	2104: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame containing the pattern instance for which you want the result.

MatchStrength

VPARAMETER
2101

Strength of the match matrix for the selected matched pattern. Match value ranges from 0 to 1, with 1 being the best quality. A value of 1 means that 100% of the reference pattern was successfully matched to the found pattern instance. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 2101, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 2101, index_id, object_id)
```

Type

Double

Range

Minimum: MatchThreshold

Maximum: 1.0

Parameter

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pattern instance for which you want the result.
ID	2101: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame containing the pattern instance for which you want the result.

MatchThreshold

VPARAMETER
5420

Sets the minimum match strength required for a pattern to be recognized as valid. A perfect match value is 1.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5420, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5420, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5420, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5420, index_id, object_id)
```

Remarks

In MicroV+/V+, the frame_id parameter is required.

Type

Double

Range

Minimum: 0.0 (weak match)

Maximum: 1.0 (strong match)

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5420: the value used to reference this property.

index_id	N/A
object_id	N/A

MaximumAngleDeviation

VPARAMETER
5102

Maximum angular deviation allowed for a detected edge to be used to generate an entity hypothesis.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5102, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5102, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5102, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5102, index_id, object_id)
```

Remarks

For an arc entity, the deviation is calculated between the tangent angle of the arc at points where the edge is matched to the arc. For a line entity, the Line Finder accepts a 20 degree deviation (default). However, the tool uses the defined MaximumAngleDeviation value to test the hypothesis and refine the pose of the found line.

Type

Double

Range

Minimum: 0 degrees

Maximum: 20 degrees

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5102: the value used to reference this property.
index_id	N/A
object_id	N/A

MaximumBlobArea

VPARAMETER
5001

Maximum area for a blob. This validation criterion is used to filter out unwanted blobs from the results.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5001, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5001, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5001, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5001, index_id, object_id)
```

Type

Double

Range

0 or greater.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5001: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MaximumGreylevelValue

VRESULT
1507

Highest greylevel value of all pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1507, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1507, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1507: the value used to reference this property.
index_id	N/A
frame_id	N/A

MaximumInstanceCount

VPARAMETER
519

Maximum number of object instances that are searched for in the input greyscale Image. All of the object instances respecting the search constraints are output, up to a maximum of **MaximumInstanceCount**. They are ordered according to the InstanceOrdering property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 519, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 519, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 519, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 519, index_id, object_id)
```

Remarks

This property is applicable only if the MaximumInstanceCountEnabled property is set to True.

Type

Double

Range

Minimum: 1

Maximum: 2000

Parameter

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	519: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MaximumInstanceCountEnabled

VPARAMETER
518

When True, limits the search to the number of instances set by the MaximumInstanceCount property

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 518, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 518, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 518, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 518, index_id, object_id)
```

Type

Long

Range

Value	Description
1	Search is limited to number of instances specified by MaximumInstanceCount.
0	Search is not limited to a set number of instances.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	518: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MaximumRotation

VPARAMETER
517

Maximum angle of rotation allowed for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 517, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 517, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 517, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 517, index_id, object_id)
```

Remarks

This property is applicable only if the NominalRotationEnabled property is set to False. When MaximumRotation is lower than MinimumRotation, the search range is equivalent to MinimumRotation to (MaximumRotation + 360 degrees).

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	517: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MaximumScaleFactor

VPARAMETER 513

Maximum scale factor allowed for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 513, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 513, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 513, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 513, index_id, object_id)
```

Remarks

This property is applicable only if the NominalRotation property is set to False.

Type

Double

Range

Minimum: 0.1

Maximum: 10.0

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	513: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Mean

VRESULT
1500

Mean of the greylevel distribution of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1500, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1500, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1500: the value used to reference this property.
index_id	N/A
object_id	N/A

MeasurementPointsCount

VRESULT
2002

The number of points where the local sharpness is evaluated. When the Image Sharpness tool is executed, it scans the region of interest and identifies a number of candidate locations (equal to CandidatePointsCount) where the local standard deviation is the highest. The local sharpness is then evaluated at each of the candidate locations that has a local standard deviation above StandardDeviationThreshold. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 2002, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 2002, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: CandidatePointsCount

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	2002: the value used to reference this property.
index_id	N/A
object_id	N/A

Median

VRESULT 1501

Median of the greylevel distribution of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1501, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1501, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1501: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumArcPercentage

VPARAMETER
5142

Minimum percentage of arc contours that need to be matched for an arc hypothesis to be considered as valid.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5142, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5142, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5142, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5142, index_id, object_id)
```

Type

Double

Range

Minimum: 1

Maximum: 100.0

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5142: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumBlobArea

VPARAMETER
5000

Minimum area for a blob. This validation criterion is used to filter out unwanted blobs from the results.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5000, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5000, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5000, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5000, index_id, object_id)
```

Type

Double

Range

0 or greater.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5000: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

MinimumClearPercentage

VPARAMETER
559

When MinimumClearPercentageEnabled is set to True, MinimumClearPercentage sets the minimum percentage of the model bounding-box area that must be free of obstacles to consider an object instance as valid.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 559, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 559, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 559, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 559, index_id, object_id)
```

Type

Double

Range

Minimum: 0.1 or greater.

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	559: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumClearPercentageEnabled

VPARAMETER
558

When set to True, the MinimumClearPercentage constraint is applied to the search process.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 558, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 558, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 558, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 558, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The MinimumClearPercentage constraint is enabled and applied to the Search process.
0	The MinimumClearPercentage constraint is not enabled.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	558: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumGreylevelValue

VRESULT
1506

Lowest greylevel value of all pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1506, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1506, index_id, frame_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1506: the value used to reference this property.
index_id	N/A
frame_id	N/A

MinimumLinePercentage

VPARAMETER
5130

Minimum percentage of line contours that need to be matched for a line hypothesis to be considered as valid.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5130, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5130, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5130, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5130, index_id, object_id)
```

Type

Double

Range

Minimum: 0.1

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5130: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumModelPercentage

VPARAMETER
557

Minimum percentage of model contours that need to be matched in the input image in order to consider the object instance as valid.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 557, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 557, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 557, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 557, index_id, object_id)
```

Type

Double

Range

Minimum: 0.1 or greater.

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	557: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumRequiredFeatures

VPARAMETER
560

Minimum percentage of required features that must be recognized in order to consider the object instance as valid.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 560, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 560, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 560, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 560, index_id, object_id)
```

Type

Double

Range

Minimum: 0.1 or greater.

Maximum: 100.0

Remark(s)

The minimum percentage of required features is expressed in terms of the number of required features in a model without considering the amount of contour each required feature represents in the model. For example, if the model contains 3 required features and MinimumRequiredFeatures is set to 50%, an instance of the object will be considered valid as long as 2 out of 3 required features are recognized.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	560: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumRotation

VPARAMETER
516

Minimum angle of rotation allowed for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 516, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 516, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 516, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 516, index_id, object_id)
```

Remarks

This property is applicable only if the NominalRotationEnabled property is set to False. When MaximumRotation is lower than MinimumRotation, the search range is equivalent to MinimumRotation to (MaximumRotation + 360 degrees).

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	516: the value used to reference this property.
index_id	N/A
object_id	N/A

MinimumScaleFactor

VPARAMETER
512

Minimum scale factor allowed for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 512, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 512, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 512, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 512, index_id, object_id)
```

Remarks

This property is applicable only if the NominalScaleFactor property is set to False.

Type

Double

Range

Minimum: 0.1

Maximum: 10.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	512: the value used to reference this property.
index_id	N/A
object_id	N/A

Mode

VRESULT
1504

Mode of the greylevel distribution of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. The mode is the greylevel value which corresponds to the histogram bin with the highest number of pixels. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1504, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1504, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1504: the value used to reference this property.
index_id	N/A
object_id	N/A

ModelDisambiguationEnabled

VPARAMETER
403

When set to True (default), the Locator applies disambiguation to discriminate between similar models and similar hypotheses of a single object. When set to False, the Locator does not apply disambiguation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 403, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 403, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 403, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 403, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Locator applies disambiguation.
0	Locator does not apply disambiguation.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	403: the value used to reference this property.

index_id	N/A
object_id	N/A

ModePixelCount

VRESULT
1505

Number of pixels in the histogram bin which corresponds to the Mode of the greylevel distribution of all pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. The mode is the greylevel value which corresponds to the histogram bin with the highest number of pixels. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1505, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1505, index_id, frame_id)
```

Type

Double

Range

Greater than or equal to 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1505: the value used to reference this property.
index_id	N/A
object_id	N/A

MorphologicalNeighborhoodSize

VPARAMETER
5390

Neighborhood size applied by a morphological operation.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5390, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5390, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5390, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5390, index_id, object_id)
```

Type

Long

Range

Fixed value: 3

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5390: the value used to reference this property.
index_id	N/A
object_id	N/A

NominalRotation

VPARAMETER 515

Required angle of rotation for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 515, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 515, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 515, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 515, index_id, object_id)
```

Remarks

This property is applicable only if the NominalRotationEnabled property is set to True.

Type

Double

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	515: the value used to reference this property.
index_id	N/A
object_id	N/A

NominalRotationEnabled

VPARAMETER
514

Specifies whether the rotation of a recognized instance must fall within the range set by MinimumRotation and MaximumRotation or be equal to the nominal value set by the NominalRotation property. When NominalRotationEnabled is set to True, the nominal value is applied. Otherwise, the range is used.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 514, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 514, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 514, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 514, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Locator searches for instances that meet NominalRotation constraint
0	Locator searches for instances within range set by MinimumRotation and MaximumRotation.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	514: the value used to reference this property.
index_id	N/A
object_id	N/A

NominalScaleFactor

VPARAMETER
511

Required scale factor for an object instance to be recognized.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 511, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 511, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 511, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 511, index_id, object_id)
```

Remarks

This property is applicable only if the NominalScaleFactorEnabled property is set to True.

Type

Long

Range

Minimum: 0.1

Maximum: 10.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	511: the value used to reference this property.
index_id	N/A
object_id	N/A

NominalScaleFactorEnabled

VPARAMETER
510

Specifies whether the scale factor of a recognized instance must fall within the range set by MinimumScaleFactor and MinimumScaleFactor or be equal to the nominal value set by the NominalScaleFactor property. When NominalScaleFactorEnabled is set to True, the nominal value is applied. Otherwise, the range is used.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 510, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 510, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 510, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 510, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Locator searches for instances that meet NominalScaleFactor constraint.
0	Locator searches for instances within range set by MinimumScaleFactor and MaximumScaleFactor.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	510: the value used to reference this property.
index_id	N/A
object_id	N/A

Operation

VPARAMETER
5355

Operation applied by the Image Processing tool.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5355, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5355, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5355, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5355, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
0	hsArithmeticAddition	Operand value (constant or Operand Image pixel) is added to the corresponding pixel in the input image.
1	hsArithmeticSubtraction	Operand value (constant or Operand Image pixel) is subtracted from the corresponding pixel in the input image.
2	hsArithmeticMultiplication	The input image pixel value is multiplied by the Operand value (constant or corresponding Operand Image pixel).
3	hsArithmeticDivision	The input image pixel value is divided by the Operand value (constant or corresponding Operand image pixel). The result is scaled and clipped, and finally written to the output image.
4	hsArithmeticLightest	The Operand value (constant or Operand Image pixel) and corresponding pixel in the input image are compared to find the maximal value.
5	hsArithmeticDarkest	The Operand value (constant or Operand Image pixel) and corresponding pixel in the input image are compared to find

Value	Name	Description
		the minimal value.
6	hsAssignmentInitialization	All the pixels of the output image are set to a specific constant value. The height and width of the output image must be specified.
7	hsAssignmentCopy	Each input image pixel is copied to the corresponding output image pixel.
8	hsAssignmentInversion	The input image pixel value is inverted and the result is copied to the corresponding output image pixel.
9	hsLogicalAnd	AND operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
10	hsLogicalNAnd	NAND operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
11	hsLogicalOr	OR operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
12	hsLogicalXOr	XOR operation is applied to the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
13	hsLogicalNOr	NOR operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image.
14	hsFilteringCustom	Applies a Custom filter.
15	hsFilteringAverage	Applies an Average filter.
16	hsFilteringLaplacian	Applies a Laplacian filter.
17	hsFilteringHorizontalSobel	Applies a Horizontal Sobel filter.
18	hsFilteringVerticalSobel	Applies a Vertical Sobel filter.
19	hsFilteringSharpen	Applies a Sharpen filter.
20	hsFilteringSharpenLow	Applies a SharpenLow filter.
21	hsFilteringHorizontalPrewitt	Applies a Horizontal Prewitt filter.

Value	Name	Description
22	hsFilteringVerticalPrewitt	Applies a Vertical Prewitt filter.
23	hsFilteringGaussian	Applies Gaussian filter.
24	hsFilteringHighPass	Applies High Pass filter.
25	hsFilteringMedian	Applies a Median filter.
26	hsMorphologicalDilate	Sets each pixel in the output image as the largest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size.
27	hsMorphologicalErode	Sets each pixel in the output image as the smallest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size.
28	hsMorphologicalClose	Has the effect of removing small dark particles and holes within objects.
29	hsMorphologicalOpen	Has the effect of removing peaks from an image, leaving only the image background.
30	hsHistogramEqualization	Equalization operation enhances the Input Image by flattening the histogram of the Input Image
31	hsHistogramStretching	Stretches (increases) the contrast in an image by applying a simple, piecewise, linear-intensity transformation based on the histogram of the Input Image.
32	hsHistogramLightThreshold	Changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the minimum acceptable value. Otherwise, it is set to the maximum presentable value.
33	hsHistogramDarkThreshold	Changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the maximum-presentable value. Otherwise, it is set to the minimum-acceptable value.
34	hsTransformFFT	Converts and outputs a frequency description of the input image by applying a Fast Fourier Transform (FFT).
35	hsTransformDCT	Converts and outputs a frequency description of the input image by applying a Discrete Cosine Transform (DCT).Parameters

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5355: the value used to reference this property.
index_id	N/A
object_id	N/A

Operator

VPARAMETER
5600

Logical operator applied by the Results Inspection tool.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5600, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5600, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5600, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5600, index_id, object_id)
```

Type

Long

Range

0 or 1

Range

Value	State	Description
1	AND	AND operator is applied.
0	OR	OR operator is applied.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5600: the value used to reference this property.
index_id	N/A
object_id	N/A

OutlineLevel

VPARAMETER
300

The coarseness of the contours at the Outline level. This property can only be set when ParametersBasedOn is set to hsParametersCustom. Otherwise, it is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 300, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 300, index_id, object_id)
```

Remarks

For most applications, the ParametersBasedOn property should be set to hsParametersAllModels. Custom contour detection should only be used when the default values do not work correctly.

Type

Long

Range

Minimum: 1

Maximum: 16

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	300: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputArcAngle

VRESULT
1841

Angle of the specified arc entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1841, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1841, index_id, frame_id)
```

Type

Double

Range

Minimum: -180

Maximum: 180 degrees

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1841: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputArcCenterPointX

VRESULT
1846

X-coordinate of the center point of the specified arc entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1846, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1846, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1846: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputArcCenterPointY

VRESULT
1847

The Y-coordinate of the center point of the specified arc entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1847, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1847, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1847: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputArcRadius

VRESULT
1840

The radius of the specified arc entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1840, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1840, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1840: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputLineAngle

VRESULT
1820

Angle of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1820, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1820, index_id, frame_id)
```

Type

Double

Range

Minimum: -180

Maximum: 180

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1820: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineEndPointX

VRESULT
1823

X-coordinate of the end point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1823, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1823, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1823: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineEndPointY

VRESULT
1824

Y-coordinate of the end point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1824, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1824, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1824: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineStartPointX

VRESULT
1821

X-coordinate of the start point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1821, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1821, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1821: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineStartPointY

VRESULT
1822

Y-coordinate of the start point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1822, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1822, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1822: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineVectorPointX

VRESULT
1825

X-coordinate of the vector point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1825, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1825, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1825: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputLineVectorPointY

VRESULT
1826

Y-coordinate of the vector point of the specified line entity. This property is read-only.

Syntax

MicroV+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1826, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1826, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1826: the value used to reference this property.
index_id	N/A
frame_id	Index of the frame that contains the specified instance. Range: [1, ResultCount -1]
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OutputPointX

VRESULT 1810

X-coordinate of the specified point entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1810, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1810, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1810: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputPointY

VRESULT
1811

Y-coordinate of the specified point entity. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1811, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1811, index_id, frame_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1811: the value used to reference this property.
index_id	N/A
frame_id	N/A

OutputSymmetricInstances

VPARAMETER
520

When set to True, all the symmetric poses of the object instance are output. If False, only the single best-quality symmetric pose of the object instance is output.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 520, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 520, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 520, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 520, index_id, object_id)
```

Remarks

See also InstanceSymmetry.

Type

Boolean

Range

Value	Description
1	Locator outputs all symmetrical poses of an instance.
0	Locator outputs only single best pose of an instance.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	520: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

OverrideType

VPARAMETER
5351

Output image type when the OverrideTypeEnabled property is set to True. By default, the Image Processing Tool outputs all resulting images as unsigned 8-bit images.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5351, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5351, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5351, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5351, index_id, object_id)
```

Type

Long

Range

Value	Name	Description
1	hsType8Bits	Unsigned 8-bit image.
10	hsType16Bits	Signed 16-bit image.
7	hsType32Bits	Signed 32-bit image

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5351: the value used to reference this property.
index_id	N/A
object_id	N/A

OverrideTypeEnabled

VPARAMETER 5350

Enables or disables the OverrideType property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5350, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5350, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5350, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5350, index_id, object_id)
```

Type

Long

Range

Value	State	Description
1	Enabled	The output image type is set based on the setting of OverrideType [5351].
0	Disabled	The output image type is automatically set to the same type as the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5350: the value used to reference this property.
index_id	N/A
object_id	N/A

PairCount

VPARAMETER
1920

PairCount indicates the number of pairs that have been configured for the tool. This property is read-only.

Syntax

Micro V+

```
value = VPARAMETER (sequence_id, tool_id, 1920, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 1920, index_id, object_id)
```

Remarks

If an edge pair is not found, results for that edge pair appear as zero. However, the PairCount property is not affected. To get the number of pairs found by the tool use the ResultCount property.

Type

Long

Range

Minimum: 0

Maximum: Unlimited

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1920: the value used to reference this property.

index_id	N/A
frame_id	N/A

PairPositionX

VRESULT
1921

X-coordinate of the center of the selected pair. The position of a pair is defined as the middle of the line segment drawn from the X/Y-coordinates of the first and second edges of the pair. This property is read-only.

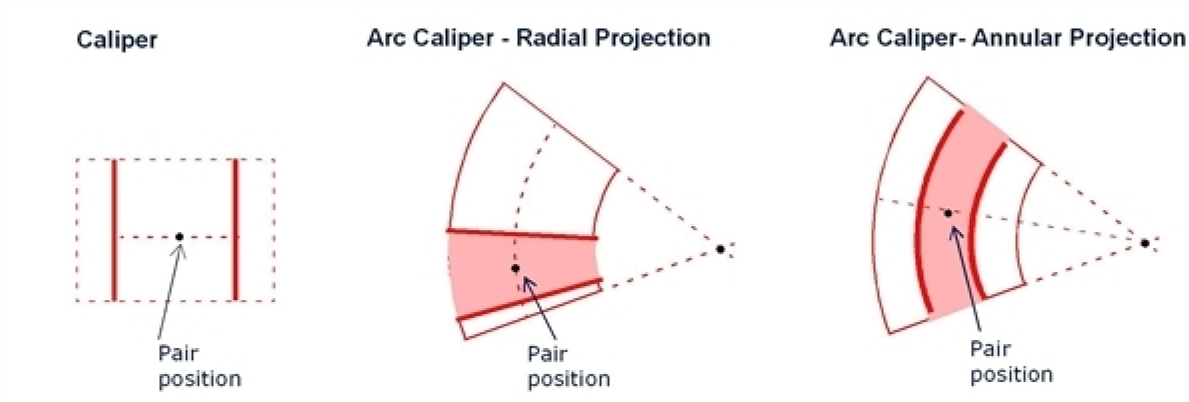


Figure: Position of Arc Caliper and Caliper Pairs

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1921, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1921, index_id, frame_id)
```

Type

Long

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pair for which you want the result.
ID	1921: the value used to reference this property.
index_id	N/A
frame_id	N/A

PairPositionY

VRESULT
1922

Y-coordinate of the center of the selected pair. The position of a pair is defined as the middle of the line segment drawn from the X/Y-coordinates of the first and second edges of the pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1922, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1922, index_id, frame_id)
```

Type

Double

Range

Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pair for which you want the result.
ID	1922: the value used to reference this property.
index_id	N/A
frame_id	N/A

PairRotation

VRESULT
1923

Angle of rotation of the selected pair in the currently-selected coordinate system. The rotation of a given pair is always the same as the rotation of its first and second edges. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1923, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1923, index_id, frame_id)
```

Type

Long

Range

Minimum: -180

Maximum: 180

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pair for which you want the result.
ID	1923: the value used to reference this property.
index_id	N/A
frame_id	N/A

PairScore

VRESULT
1924

Score of the selected pair. The score of the pair is equal to the mean score of the two edges (Edge1Score and Edge2Score) that form the pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1924, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1924, index_id, frame_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the result.
ID	1924: the value used to reference this property.
index_id	N/A
frame_id	N/A

PairSize

VRESULT
1925

Size of the selected pair. The size of the pair is equal to the mean size of the two edges (Edge1Score and Edge2Score) that form the pair. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1925, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1925, index_id, frame_id)
```

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the pair for which you want the result.
ID	1925: the value used to reference this property.
index_id	N/A
frame_id	Frame containing the pair.

ParametersBasedOn

VPARAMETER
304

Sets how the contour detection parameters are configured.

- When set to hsContourParametersAllModels, the contour detection parameters are optimized by analyzing the parameters used to build all the models.
- When set to hsContourParametersCustom, the contour detection parameters are set manually.
- When set to a value greater than hsContourParametersCustom, the contour detection parameters of a specific model are used.

The contour detection parameters on which this property has an effect are DetailLevel, OutlineLevel, ContrastThresholdMode, and ContrastThreshold

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 304, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 304, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 304, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 304, index_id, object_id)
```

Remarks

For most applications, the ParametersBasedOn property should be set to hsParametersAllModels. Custom contour detection should only be used when the default values do not work correctly.

Type

Long

Range

Value	Detection Mode	Description
-2	hsContourParametersAllModels	The contour detection parameters are optimized by analyzing the parameters used to build all the models.

Value	Detection Mode	Description
-1	hsContourParametersCustom	The contour detection parameters are set manually.
Integer from 1 - 10	*Integer* specifying the index of a model	The contour detection parameters of the specified model are used.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	304: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

PatternHeight

VPARAMETER
5403

Height of the region of interest of the Pattern. This is the sample pattern for which the Pattern Locator searches.

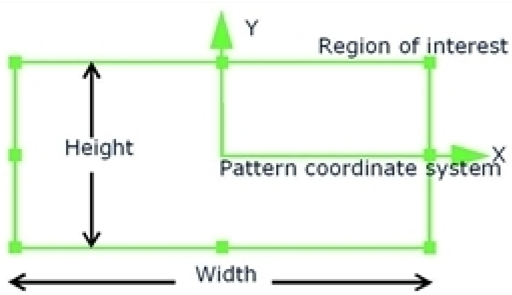


Figure: Illustration of Pattern Height and Width

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5403, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5403, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5403, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5403, index_id, object_id)
```

Type

Long

Range

Greater than or equal to three pixels. Minimum size is 3x3 pixels.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5403: the value used to reference this property.
index_id	N/A
object_id	N/A

PatternPositionX

VPARAMETER
5400

X-coordinate of the center of the pattern region of interest. This is the sample pattern for which the Pattern Locator searches.

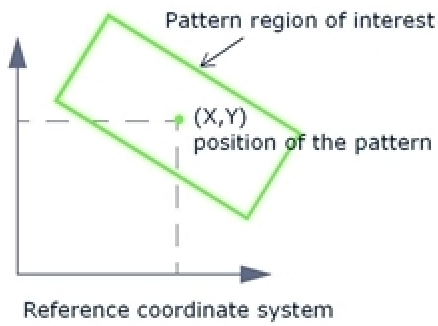


Figure: Illustration of the Pattern location

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5400, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5400, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5400, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5400, index_id, object_id)
```

Type

Long

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5400: the value used to reference this property.
index_id	N/A
object_id	N/A

PatternPositionY

VPARAMETER
5401

Y-coordinate of the center of the pattern region of interest. This is the sample pattern for which the Pattern Locator searches.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5401, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5401, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5401, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5401, index_id, object_id)
```

Type

Long

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5401: the value used to reference this property.
index_id	N/A
object_id	N/A

PatternRotation

VPARAMETER
5404

Angle of rotation of the pattern region of interest. This is the sample pattern for which the Pattern Locator searches.

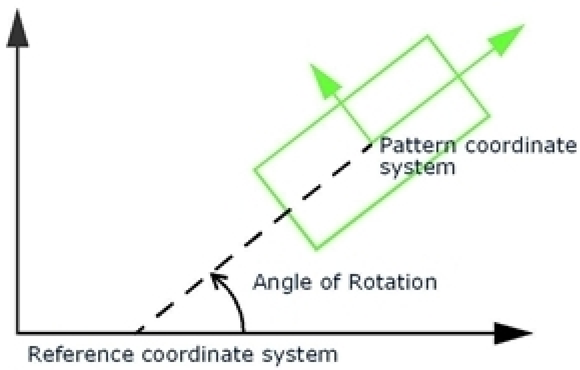


Figure: Illustration of Pattern Rotation

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5404, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5404, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5404, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5404, index_id, object_id)
```

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5404: the value used to reference this property.

index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

PatternWidth

VPARAMETER
5402

Width of the pattern region of interest.This is the sample pattern for which the Pattern Locator searches.

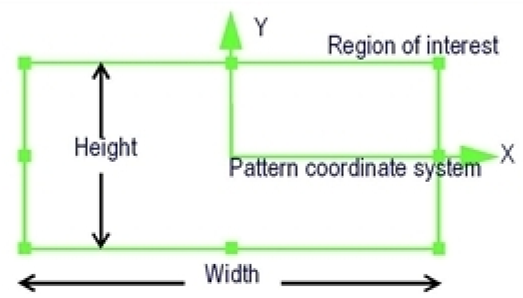


Figure: Illustration of Pattern Height and Width

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5402, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5402, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5402, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5402, index_id, object_id)
```

Type

Long

Range

Greater than or equal to three pixels. Minimum size is 3x3 pixels.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5402: the value used to reference this property.
index_id	N/A
object_id	N/A

PerimeterResultsEnabled

VPARAMETER
1602

Enables the computation of the following blob properties: BlobRawPerimeter, BlobConvexPerimeter and BlobRoundness.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1602, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1602, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1602, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1602, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The perimeter properties will be computed.
0	No perimeter properties will be computed.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1602: the value used to reference this property.

index_id	N/A
object_id	N/A

PolarityMode

VPARAMETER
5100

Selects the type of polarity accepted for finding an entity. Polarity identifies the change in greylevel values from the tool center (inside) towards the outside.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5100, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5100, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5100, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5100, index_id, object_id)
```

Type

Long

Range

Value	Mode	Description
0	hsDarkToLight	The tool searches only for arc instances occurring at a dark-to-light transition in greylevel values.
1	hsLightToDark	The tool searches only for arc instances occurring at a light-to-dark transition in greylevel values.
2	hsEither	The tool searches only for arc instances occurring either at a light-to-dark or dark-to-light transition in greylevel values.
3	hsDontCare	The tool searches only for arc instances occurring at any transition in greylevel values including reversals in contrast along the arc, for example, on an unevenly colored background.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5100: the value used to reference this property.
index_id	N/A
object_id	N/A

PositionConstraint

VPARAMETER
5223

Indexed property used to set the position-constraint function for edge detection. Four points are used: Base Left, Top Left, Top Right, Base Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5223, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5223, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5223, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5223, index_id, object_id)
```

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5223: the value used to reference this property.
index_id	N/A
object_id	N/A

PositioningLevel

VPARAMETER 561

Configurable effort level of the instance positioning process. The value is expressed as a percentage. The minimal allowable value is 10. Lower values will provide coarser positioning and lower execution time. Conversely, a value of 100 will provide the highest accuracy for the positioning of object instances.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 561, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 561, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 561, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 561, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 10

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	561: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ProjectionMode

VPARAMETER
140

Projection mode used by the tool to detect edges.

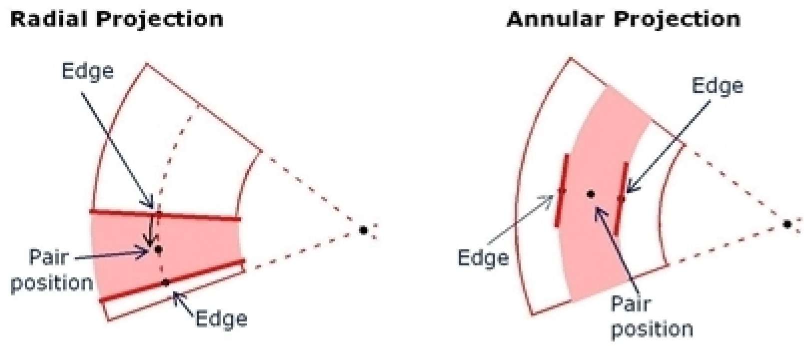


Figure: Projection Modes used by Arc Caliper and Arc Edge Caliper

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 140, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 140, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 140, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 140, index_id, object_id)
```

Type

0 or 1

Range

Value	Projection Mode	Description
0	hsProjectionAnnular	Annular projection is used to find edges that are aligned with the median annulus, such as arcs on concentric circles.
1	hsProjectionRadial	Radial projection is used to find edges aligned along radial projections, similar to the spokes of a wheel.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	140: the value used to reference this property.
index_id	N/A
object_id	N/A

RecipeManagerActiveRecipe

VPARAMETER
8001

Returns the currently selected recipe associated with the Recipe Manager.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 8001, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 8001, index_id, object_id)
```

Type

Real variable.

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8001: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecipeManagerRecipeCount

VPARAMETER
8000

Returns the number of available recipes associated with the Recipe Manager.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 8000, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 8000, index_id, object_id)
```

Type

Real variable.

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8000: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecipeManagerRecipeSelection

VPARAMETER 8002

Identifies the new recipe that will be selected when a VRUN is issued against the Recipe Manager.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 8002, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 8002, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 8002, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 8002, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of available recipes - 1

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8002: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecipeManagerRecipeDelete

VPARAMETER 8003

Deletes a specified recipe from a recipe manager object in the workspace.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 8003, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 8003, index_id, object_id) $ip = value
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of available recipes - 1

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8003: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecipeManagerLoadFile

VPARAMETER 8004

Load a recipe from a file into the specified recipe manager. When this command is invoked, the variable $as.filename[0] will be read to identify the file to load. A sample showing how this is used is located in the ***ASIGHT.V2*** module in the program ***as.load.recipe***.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 8004, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 8004, index_id, object_id) $ip = value
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of available recipes - 1

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8004: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecipeManagerSaveFile

VPARAMETER
8005

Saves a recipe into a file from the specified recipe manager. When this command is invoked, the variable $as.filename[0] will be read to identify the file to save into. A sample showing how this is used is located in the ***ASIGHT.V2*** module in the program ***as.save.recipe***.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 8005, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 8005, index_id, object_id) $ip = value
```

Type

Real variable.

Range

Minimum: 0

Maximum: The number of available recipes - 1

Parameters

sequence_id	Index associated with the recipe manager as defined in the recipe manager editor.
tool_id	N/A
ID	8004: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

RecognitionLevel

VPARAMETER
550

Configurable effort level of the search process. A value of 0 will lead to a faster search that may miss instances that are partly occluded. Conversely, a value of 10 is useful for finding partly occluded objects in cluttered or noisy images, or for models made up of small features at the Outline Level.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 550, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 550, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 550, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 550, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 10

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	550: the value used to reference this property.
index_id	N/A
object_id	N/A

Reset

VPARAMETER 5500

Resets the data currently stored for the tool. This property is write-only.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5500, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 5500, index_id, object_id) $ip = value
```

Type

Long

Range

Not applicable

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5500: the value used to reference this property.
index_id	N/A
object_id	N/A

ResultCount

VRESULT
1010

Returns the total number of results found by the tool in all frames of reference. If you want the number of results within a specified frame of reference, see FrameCount. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1900, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1900, index_id, frame_id)
```

Type

Long

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	N/A
ID	1900: the value used to reference this property.
index_id	N/A
frame_id	N/A

Related Topics

FrameCount
InstanceCount

RobotXPosition

VPARAMETER
10404

X-coordinate (in millimeters) of a location in the robot frame of reference, which is required for the InverseKinematics property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10404, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10404, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10404, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10404, index_id, object_id)
```

Type

Long

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	10404: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Example

See the InverseKinematics for an example of this property and related properties

Related Properties

RobotYPosition
VisionXPosition

VisionYPosition
VisionRotation
InverseKinematics

RobotYPosition

VPARAMETER
10405

Y-coordinate (in millimeters) of a location in the robot frame of reference, which is required for the InverseKinematics property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10405, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10405, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10405, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10405, index_id, object_id)
```

Type

Long

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	10405: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Example

See the InverseKinematics for an example of this property and related properties

Related Properties
RobotXPosition
VisionXPosition

VisionYPosition
VisionRotation
InverseKinematics

SamplingStepCustom

VPARAMETER
124

When SamplingStepCustomEnabled is set to True, this property defines the sampling step used to sample the region of interest from the input image. WhenSamplingStepCustomEnabled is set to False, the default sampling step is used.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 124, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 124, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 124, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 124, index_id, object_id)
```

Remark

A custom sampling step is usually not recommended.

Type

Single

Range

Minimum: Greater than zero.

Maximum: Boundaries of the input image.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	124: the value used to reference this property.

index_id	N/A
object_id	N/A

Related Properties

SamplingStepCustomEnabled_

SamplingStepCustomEnabled

VPARAMETER
121

When enabled, the tool uses the user-defined sampling step (SamplingStepCustom) instead of the optimal (default) sampling step to sample the region of interest from the input image.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 121, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 121, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 121, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 121, index_id, object_id)
```

Remark

A custom sampling step is usually not recommended.

Type

Boolean

Range

Value	Description
0	The tool uses the default sampling step.
1	The default sampling step is overridden by SamplingStepCustom.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.

tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	121: the value used to reference this property.
index_id	N/A
object_id	N/A

Related Properties

SamplingStepCustom

SaveImage

VPARAMETER 10327

Saves the current image to file. Various file formats are available, including the Adept hig file format. The hig format saves the calibration information in the image file. Files with this format can be reused in ACE Sight applications through an Emulation device. This property is write-only.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10327, index_id, object_id) = value
```

V+

```
VPARAMETER (sequence_id, tool_id, 10327, index_id, object_id) $ip = value
```

Type

Long

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	10327: the value used to reference this property.
index_id	N/A
object_id	N/A

Details

This functionality can be accessed using the ACE API program "as.save.image", as follows:

```
.PROGRAM as.save.image($filename, $ip, seq.idx, tool_id.idx, status)
```

For more details on as.save.image, see the ACE Sight section of the V+ Module chapter in the *ACE Reference Guide*, which can be accessed from the Help menu in the ACE software.

ScoreThreshold

VPARAMETER
5240

Minimum score to accept an edge. The score of an edge is returned by the EdgeScore property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5240, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5240, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5240, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5240, index_id, object_id)
```

Remarks

Type

Double

Range

Minimum: 0.0

Maximum: 1.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5240: the value used to reference this property.
index_id	N/A
object_id	Index of the frame containing the edge pair.

SearchBasedOnOutlineLevelOnly

VPARAMETER
521

When set to True, the Locator positions object instances using Outline-level models. This mode can be used to improve the speed when only a coarse positioning of object instances is required.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 521, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 521, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 521, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 521, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Only Outline-level models are used to position instances
0	Both Outline- and Detail-level models are used to position instances.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	521: the value used to reference this property.
index_id	N/A
object_id	N/A

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

SearchCoarseness

VPARAMETER
5430

Subsampling level used to find pattern-match hypotheses. High values provide a coarser search and lower execution time than lower values. If AutoCoarsenessSelectionEnabled is set to True, this property is read-only.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5430, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5430, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5430, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5430, index_id, object_id)
```

Type

Long

Range

[1,2,4,8,16,32]

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5430: the value used to reference this property.
index_id	N/A
object_id	N/A

SearchMode

VPARAMETER
5101

Specifies the method used by a Finder tool to select a hypothesis.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5101, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5101, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5101, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5101, index_id, object_id)
```

Type

Long

Range

The range depends on the type of entity. For arcs (Arc Finder Tool), the range is:

Value		Description
0	hsBestArc	Selects the best arc according to hypothesis strength.
1	hsArcClosestToGuideline	Selects the arc hypothesis closest to the Guideline.
2	hsArcClosestToInside	Selects the arc hypothesis closest to the inside of the tool Search Area. (closest to the tool center).
3	hsArcClosestToOutside	Selects the arc hypothesis closest to the outside of the tool Search Area. (furthest from the tool center)

For lines (Line Finder Tool), the range is:

Value		Description
0	hsBestLine	Selects the best line according to hypothesis

Value		Description
		strength.
1	hsLineClosestToGuideline	Selects the line hypothesis closest to the Guideline.
2	hsLineWithMaximumNegativeXOffset	Selects the line hypothesis closest to the Search Area bound that is at maximum negative X-offset.
3	hsLineWithMaximumPositiveXOffset	Selects the line hypothesis closest to the Search Area bound that is at maximum positive X-offset.

For points (Point Finder Tool), the range is:

Value		Description
1	hsPointClosestToGuideline	Selects the point hypothesis closest to the Guideline.
2	hsPointWithMaximumNegativeXOffset	Selects the point hypothesis closest to the Search Area bound that is at maximum negative X-offset.
3	hsPointWithMaximumPositiveXOffset	Selects the point hypothesis closest to the Search Area bound that is at maximum positive X-offset

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5101: the value used to reference this property.
index_id	N/A
object_id	N/A

SegmentationDark

VPARAMETER
5005

Indexed property used to access the Dark Segmentation function. Two points are available, from left to right: Top and Bottom.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5005, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5005, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5005, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5005, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5005: the value used to reference this property.
constraint_id	Point index for the Dark Segmentation function. (hsSegmentationDarkPoint) 1: DarkTop point

	2: DarkBottom point
object_id	N/A

SegmentationDynamicDark

VPARAMETER
5009

Indexed property used to access the Dynamic Dark Segmentation function. Two points are available, from left to right: Top and Bottom.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5009, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5009, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5009, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5009, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5009: the value used to reference this property.
constraint_id	Point index for the Dynamic Dark Segmentation function. (hsSegmentationDarkPoint) 1: DarkTop point

	2: DarkBottom point
object_id	N/A

SegmentationDynamicInside

VPARAMETER
5010

Indexed property used to access the Dynamic Inside Segmentation function. Four points are available, from left to right: Bottom Left, Top Left, Top Right and Bottom Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5010, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5010, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5010, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5010, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5010: the value used to reference this property.
constraint_id	Point index for the Dynamic Inside Segmentation. (hsSegmentationInsidePoint) 0: hsInsideBottomLeft point

	1: hsInsideTopLeft point 2: hsInsideTopRight point 3: hsInsideBottomRight point
object_id	N/A

SegmentationDynamicLight

VPARAMETER
5008

Indexed property used to access the Dynamic Light Segmentation function. Two points are available (from left to right): Bottom and Top.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5008, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5008, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5008, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5008, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5008: the value used to reference this property.
constraint_id	Point index for the Dynamic Light Segmentation function. (hsSegmentationLightPoint) 1: hsLightBottom point

	2: hsLightTop point
object_id	N/A

SegmentationDynamicOutside

VPARAMETER
5011

Indexed property used to access the Dynamic Outside Segmentation function. Four points are available, from left to right: Top Left, Bottom Left, Bottom Right and Top Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5011, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5011, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5011, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5011, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0.0

Maximum: 100.0

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5011: the value used to reference this property.
constraint_id	Point index for the Dynamic Outside Segmentation function. (hsSegmentationOutsidePoint) 0: hsOutsideTopLeft point

	1: hsOutsideBottomLeft point 2: hsOutsideBottomRight point 3: hsOutsideTopRight point
object_id	N/A

SegmentationInside

VPARAMETER
5006

Indexed property used to access the Inside Segmentation function. Four points are available: from left to right, Bottom Left, Top Left, Top Right and Bottom Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5006, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5006, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5006, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5006, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5006: the value used to reference this property.
constraint_id	Point index for the Inside Segmentation. (hsSegmentationInsidePoint) 0: hsInsideBottomLeft point

	1: hsInsideTopLeft point 2: hsInsideTopRight point 3: hsInsideBottomRight point
object_id	N/A

SegmentationLight

VPARAMETER
5004

Indexed property used to access the Light Segmentation function. Two points are available, from left to right: Bottom and Top.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5004, index_id, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5004, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5004, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5004, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5004: the value used to reference this property.
constraint_id	Point index for the Light Segmentation function. (hsSegmentationLightPoint) 1: hsLightBottom point

	2: hsLightTop point
object_id	N/A

SegmentationMode

VPARAMETER 5003

Segmentation mode used by the tool to segment the input image.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5003, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5003, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5003, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5003, index_id, object_id)
```

Type

Long

Range

Value	Segmentation Mode
0	hsLight
1	hsDark
2	hsInside
3	hsOutside
4	hsDynamicLight
5	hsDynamicDark
6	hsDynamicInside
7	hsDynamicOutside
8	HSL Inside
9	HSL Outside

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5003: the value used to reference this property.
index_id	N/A
object_id	N/A

SegmentationOutside

VPARAMETER
5007

Indexed property used to access the Outside Segmentation function. Four points are available, from left to right: Top Left, Bottom Left, Bottom Right and Top Right.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5007, constraint_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5007, constraint_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5007, constraint_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5007, constraint_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5007: the value used to reference this property.
constraint_id	Point index for the Outside Segmentation function. (hsSegmentationOutsidePoint) 0: hsOutsideTopLeft point

	1: hsOutsideBottomLeft point 2: hsOutsideBottomRight point 3: hsOutsideTopRight point
object_id	N/A

SequenceExecutionMode

VPARAMETER
10200

Sets the mode for execution of the sequence. When VRUN is called the sequence is run if single execution mode is selected (0) or in continuously if continuous mode is selected.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10200, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10200, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10200, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10200, index_id, object_id)
```

Type

Long

Range

Value	Execution Mode	Description
0	Single execution	Executes the vision sequence once.
1	Continuous mode	Executes the vision sequence continuously, until the execution is stopped by a program instruction.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Should always be set to -1.
ID	10200: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

SequenceToolCount

VPARAMETER
10201

Returns the number of tools in the specified ACE Sightr sequence.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 10201, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 10201, index_id, object_id)
```

Type

Long

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Should always be set to -1.
ID	10201 : the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Sharpness

VRESULT 2000

Average sharpness computed in the region of interest. When the Image Sharpness tool is executed, it scans the region of interest and identifies a number of candidate locations (equal to CandidatePointsCount) where the local standard deviation is the highest. The local sharpness is then evaluated at each of the candidate locations that has a local standard deviation above StandardDeviationThreshold. The tool then computes the average of all the local sharpness values, which were computed at every measurement point, and returns it through the Sharpness property. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 2000, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 2000, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 1000

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	2000: the value used to reference this property.
index_id	N/A
object_id	N/A

SharpnessPeak

VRESULT
2001

Maximum Sharpness value computed by the tool. This property is read-only.

Syntax

MicroV+

```
value = VPARAMETER (sequence_id, tool_id, 2001, index_id, object_id)
```

V+

```
value = VPARAMETER ($ip, sequence_id, tool_id, 2001, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 1000

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	2001: the value used to reference this property.
index_id	N/A
object_id	N/A

ShowResultsGraphics

VPARAMETER
150

When enabled, vision results are displayed in the image display control. When disabled, vision results are not displayed.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 150, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 150, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 150, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 150, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Enabled: Vision results are displayed in the image display control.
0	Disabled: Vision results are not displayed.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	150: the value used to reference this property.

index_id	N/A
object_id	N/A

SortBlobsBy

VPARAMETER
1601

Sorting mode used by the tool to sort the found blobs.

Remark(s)

Default Value: 0:hsArea

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1601, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1601, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1601, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1601, index_id, object_id)
```

Type

Long

Range

Value	Sorting Mode
0	hsArea
1	hsBoundingBoxBottom
2	hsBoundingBoxCenterX
3	hsBoundingBoxCenterY
4	hsBoundingBoxHeight
5	hsBoundingBoxLeft
6	hsBoundingBoxRight
7	hsBoundingBoxRotation
8	hsBoundingBoxTop

Value	Sorting Mode
9	hsBoundingBoxWidth
10	hsChainCodeDeltaX
11	hsChainCodeDeltaY
12	hsChainCodeLength
13	hsChainCodeStartX
14	hsChainCodeStartY
15	hsConvexPerimeter
16	hsElongation
17	hsExtentBottom
18	hsExtentLeft
19	hsExtentRight
20	hsExtentTop
21	hsGreyLevelMaximum
22	hsGreyLevelMean
23	hsGreyLevelMinimum
24	hsGreyLevelRange
25	hsGreyLevelStdDev
26	hsHoleCount
27	hsInertiaMaximum
28	hsInertiaMinimum
29	hsInertiaXAxis
30	hsInertiaYAxis
31	hsIntrinsicBoundingBoxBottom
32	hsIntrinsicBoundingBoxCenterX
33	hsIntrinsicBoundingBoxCenterY
34	hsIntrinsicBoundingBoxHeight
35	hsIntrinsicBoundingBoxLeft
36	hsIntrinsicBoundingBoxRight

Value	Sorting Mode
37	hsIntrinsicBoundingBoxRotation
38	hsIntrinsicBoundingBoxTop
39	hsIntrinsicBoundingBoxWidth
40	hsIntrinsicExtentBottom
41	hsIntrinsicExtentLeft
42	hsIntrinsicExtentRight
43	hsIntrinsicExtentTop
44	hsPositionX
45	hsPositionY
46	hsPrincipalAxesRotation
47	hsRawPerimeter
48	hsRoundness

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1601: the value used to reference this property.
index_id	N/A
object_id	N/A

SortResultsEnabled

VPARAMETER
1600

Specifies if the found blobs are sorted in descending order using the sorting mode set by the SortBlobsBy property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1600, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1600, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1600, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1600, index_id, object_id)
```

Type

Boolean

Range

Value	Description
0	Blobs are not sorted.
1	Blobs are sorted in descending order using the sorting mode set by SortBlobsBy.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1600: the value used to reference this property.
index_id	N/A
object_id	N/A

StandardDeviation

VRESULT
1503

Standard deviation of the greylevel distribution of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1503, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1503, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1503: the value used to reference this property.
index_id	N/A
object_id	N/A

StandardDeviationThreshold

VPARAMETER
5302

Threshold used to validate candidate locations before computing their local sharpness. When the tool is executed, it scans the region of interest and identifies a number of candidate locations (equal to CandidatePointsCount) where the local standard deviation is the highest. The local sharpness is then evaluated at each of the candidate location that has a local standard deviation above StandardDeviationThreshold.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 5302, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5302, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5302, index_id, object_id) $ip = value value =
        VPARAMETER ($ip, sequence_id, tool_id, 5302, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5302: the value used to reference this property.
index_id	N/A
object_id	N/A

SubsamplingLevel

VPARAMETER
5110

Subsampling level used to detect edges that are used by the tool to generate hypotheses. High values provide a coarser search and lower execution time than lower values.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5110, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5110, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5110, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5110, index_id, object_id)
```

Type

Long

Range

Minimum: 1

Maximum: 8

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5110: the value used to reference this property.
index_id	N/A
object_id	N/A

TailBlack

VPARAMETER
5323

Amount of pixels to ignore at the dark end of the greylevel distribution in the tool region of interest. TailBlack is expressed as a percentage of the total number of pixels in the histogram before tails are removed. After its creation, the histogram is scanned starting from bin 0. The bins at the dark end of the histogram are then cleared until the amount of pixels defined by TailBlack is reached.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5323, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5323, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5323, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5323, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 100

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5323: the value used to reference this property.
index_id	N/A
object_id	N/A

TailBlackGreylevelValue

VRESULT
1509

Represents the darkest greylevel value that remains in the histogram after the tail is removed. Used in conjunction with TailBlack, which is used to ignore pixels at the dark end of the greylevel distribution.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 1509, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1509, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1509, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1509, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1509: the value used to reference this property.
index_id	N/A
object_id	N/A

TailWhite

VPARAMETER
5322

Amount of pixels to ignore at the bright end of the greylevel distribution in the tool region of interest. TailWhite is expressed as a percentage of the total number of pixels in the histogram before tails are removed. After its creation, the histogram is scanned starting from bin 255. The bins at the bright end of the histogram are then cleared until the amount of pixels defined by TailWhite is reached.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5322, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5322, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5322, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5322, index_id, object_id)
```

Type

Double

Range

Minimum: 0

Maximum: 100

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5322: the value used to reference this property.
index_id	N/A
object_id	N/A

TailWhiteGreylevelValue

VRESULT
1510

Represents the brightest greylevel value that remains in the histogram after the tail is removed. Used in conjunction with TailWhite, which is used to ignore pixels at the bright end of the greylevel distribution.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 1510, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1510, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1510, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1510, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1510: the value used to reference this property.
index_id	N/A
object_id	N/A

ThresholdBlack

VPARAMETER
5320

Darkest greylevel value to consider when building the histogram. Greylevel values below ThresholdBlack are ignored during the histogram creation process. When a threshold is used and the tool is also configured to remove a percentage of pixels at the dark tail of the histogram (see the TailBlack property), the tail-removal process begins to scan the histogram at the bin corresponding to ThresholdBlack instead of starting at bin 0.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5320, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5320, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5320, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5320, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5320: the value used to reference this property.
index_id	N/A
object_id	N/A

ThresholdWhite

VPARAMETER
5321

Brightest greylevel value to consider when building the histogram. Greylevel values above ThresholdWhite are ignored during the histogram creation process. When a threshold is used and the tool is also configured to remove a percentage of pixels at the bright tail of the histogram (see the TailWhite property), the tail removal process begins to scan the histogram at the bin corresponding to ThresholdWhite instead of starting at bin 255.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5321, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5321, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5321, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5321, index_id, object_id)
```

Type

Long

Range

Minimum: 0

Maximum: 255

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5321: the value used to reference this property.
index_id	N/A
object_id	N/A

Timeout

VPARAMETER 501

Time (in milliseconds) after which the Locator tool aborts its search process. This timeout period does not include the model-learning phase. When the timeout is reached and TimeoutEnabled is set to True, the instances recognized up to the timeout are output by the Locator and the search is aborted.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 501, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 501, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 501, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 501, index_id, object_id)
```

Remarks

Due to internal and operating system latencies, it may take the Locator a few milliseconds more than the time specified by Timeout to abort. This property is applied only when TimeoutEnabled is set to True.

Type

Long

Range

Minimum: 1 ms

Maximum: 60,000 ms

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	501: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

TimeoutEnabled

VPARAMETER
500

Specifies if the timeout period set by the Timeout property will be used to limit the search time.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 500, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 500, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 500, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 500, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	Search time is limited by the Timeout property.
0	Search time is not limited.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	500: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolGuidelineOffset

VPARAMETER 130

The radial offset of the Guideline marker from the center of the tool search area.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 130, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 130, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 130, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 130, index_id, object_id)
```

Type

Long

Range

Maximum: +0.5*ToolThickness

Maximum: -0.5*ToolThickness

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	130: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolHeight

VPARAMETER
111

Height of the tool region of interest.

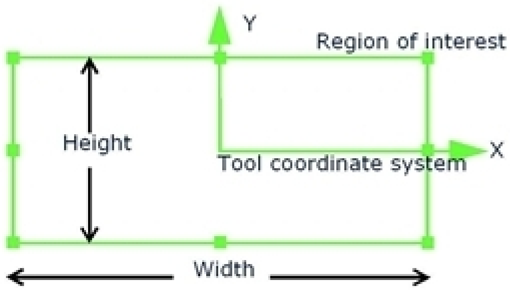


Figure: Illustration of Tool Height and Width

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 111, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 111, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 111, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 111, index_id, object_id)
```

Type

Double

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	111: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolOpening

VPARAMETER 137

Angle between the two bounding radii of the tool sector.

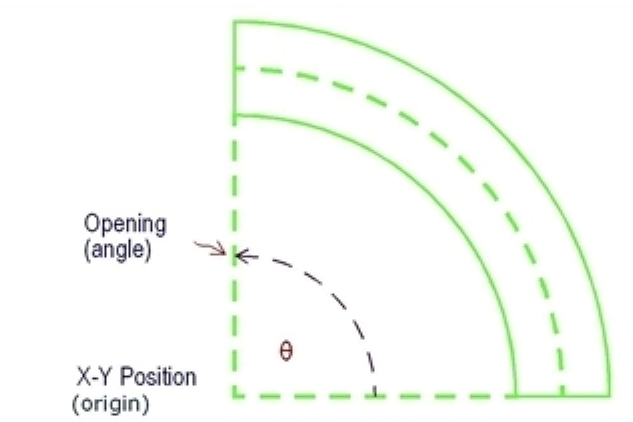


Figure: Illustration of the ToolOpening Property

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 137, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 137, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 137, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 137, index_id, object_id)
```

Type

Long

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	137: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolPositionX

VPARAMETER
100

X-coordinate of the center of the tool region of interest.

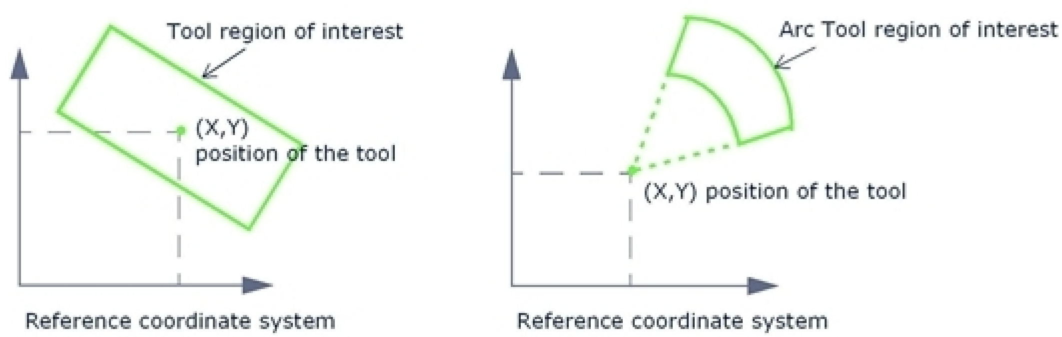


Figure: Illustration of the Tool position

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 100, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 100, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 100, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 100, index_id, object_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP

	address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	100: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolPositionY

VPARAMETER 101

Y-coordinate of the center of the tool region of interest.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 101, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 101, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 101, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 101, index_id, object_id)
```

Type

Double

Range

Unbounded

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	101: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolRadius

VPARAMETER
135

The radius of the tool corresponds to the radius of the median annulus of the tool sector.

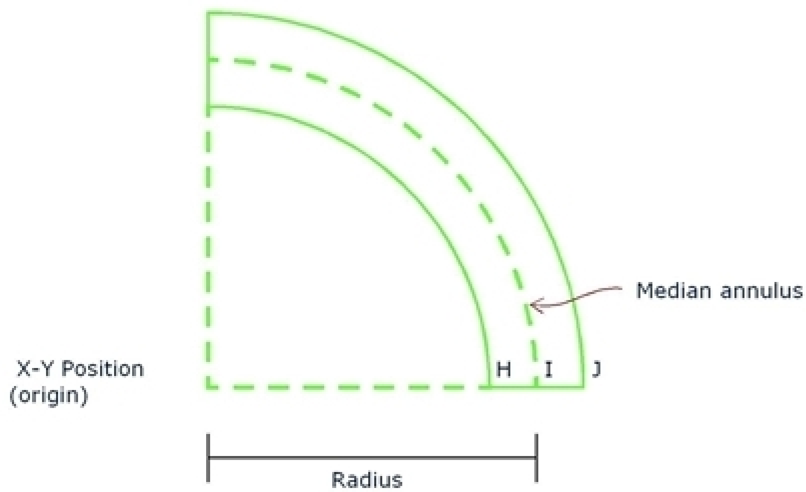


Figure: Illustration of the ToolRadius Property

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 135, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 135, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 135, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 135, index_id, object_id)
```

Type

Long

Range

Greater than 0.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	135: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolRotation

VPARAMETER 112

Angle of rotation of the tool region of interest.

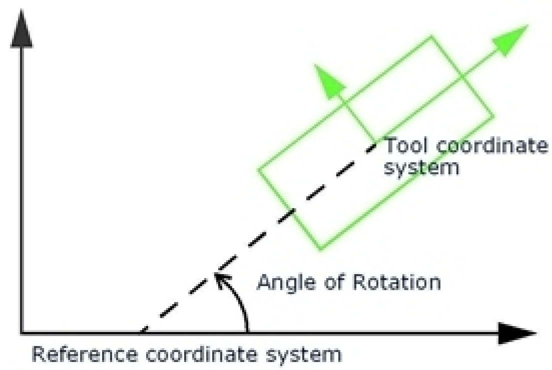


Figure: Illustration of Tool Rotation

MicroV+

```
VPARAMETER (sequence_id, tool_id, 112, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 112, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 112, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 112, index_id, object_id)
```

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	112: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ToolSkew

VPARAMETER 113

Skew angle of the tool region of interest. The angle of skew is defined as the angle between the Y-axis of an orthogonal Tool coordinate system and the Y-axis of the skewed Tool coordinate system.
The Edge Locator tool and the Caliper tool search for edges that are parallel to the X-axis of the tool. Skewing allows positioning of the tool to match the inclination of features within an object.

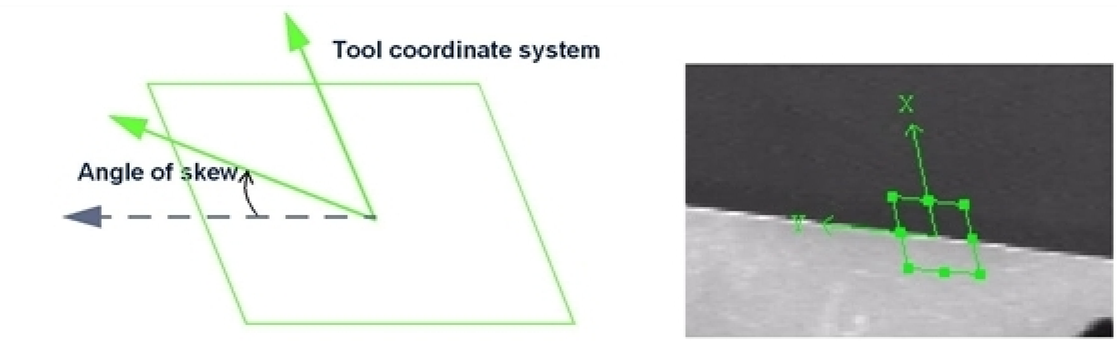


Figure: Illustration of the tool skew and image of use on an object

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 113, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 113, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 113, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 113, index_id, object_id)
```

Type

Double

Range

Minimum: -90 degrees.

Maximum: +90 degrees.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	113: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

ToolThickness

VPARAMETER
136

Distance between the two bounding annuli of the tool region of interest.

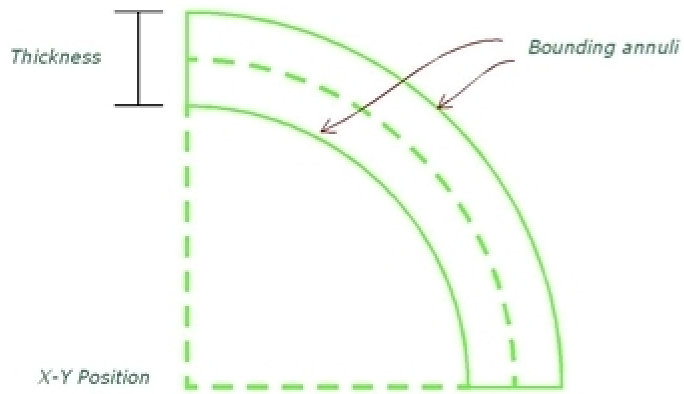


Figure: Illustration of the ToolThickness Property

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 136, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 136, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 136, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 136, index_id, object_id)
```

Type

Long

Range

1 or greater.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	136: the value used to reference this property.
index_id	N/A
object_id	N/A

ToolWidth

VPARAMETER 110

Width of the tool region of interest.

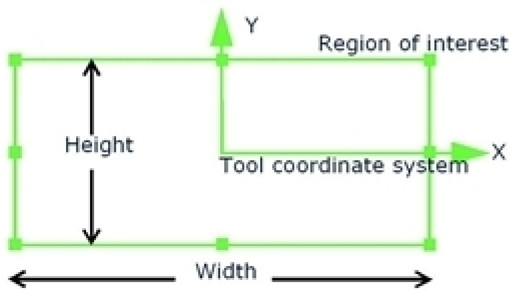


Figure: Illustration of Tool Height and Width

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 110, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 110, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 110, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 110, index_id, object_id)
```

Type

Double

Range

Greater than 0.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	110: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

TopologicalResultsEnabled

VPARAMETER 1609

Enables the computation of the BlobHoleCount property.

Syntax

Micro V+

```
VPARAMETER (sequence_id, tool_id, 1609, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 1609, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 1609, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 1609, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	The topological properties will be computed.
0	No topological properties will be computed.

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1609: the value used to reference this property.
index_id	N/A
object_id	N/A

TransformFlags

VPARAMETER 5395

Sets the flag used by a transform operation, either FFT or DCT.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5395, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5395, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5395, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5395, index_id, object_id)
```

Type

Long

Range

Value	Transform Flag Type	Description
0	TransformFlag2DLinear	Transform results output in linear 2D format
1	hsTransformFlag2DLogarithmic	Transform results in logarithmic scale
2	hsTransformFlag1DLinear	Transform results in 1D linear format
3	hsTransformFlagHistogram	Transform results as histogram

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	5395: the value used to reference this property.
index_id	N/A
object_id	N/A

UseDefaultConformityTolerance

VPARAMETER
551

Specifies whether the default conformity tolerance returned by the DefaultConformityTolerance is used instead of the user-defined conformity tolerance set by the ConformityTolerance property. When set to True, the default conformity tolerance is used.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 551, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 551, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 551, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 551, index_id, object_id)
```

Type

Boolean

Range

Value	Description
1	DefaultConformityToleranceRange is enabled.
0	DefaultConformityToleranceRange is disabled.

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	551: the value used to reference this property.
index_id	N/A

object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Variance

VRESULT
1502

Variance of the greylevel distribution of the pixels in the tool region of interest that are included in the final histogram. Pixels removed from the histogram by tails or thresholds are not included in this calculation. This property is read-only.

Syntax

Micro V+

```
value = VRESULT (sequence_id, tool_id, instance_id, 1502, index_id, frame_id)
```

V+

```
value = VRESULT ($ip, sequence_id, tool_id, instance_id, 1502, index_id, frame_id)
```

Type

Double

Range

Minimum: 0

Maximum: 65535

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	1502: the value used to reference this property.
index_id	N/A
object_id	N/A

VideoExposure

VPARAMETER
5502

Reads and writes the exposure setting for the active settings object relative to a camera.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5502, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5502, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5502, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5502, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 32767

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5502: the value used to reference this property.
index_id	Robot number to select.
object_id	Index of the tool tip to access.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

VideoGain

VPARAMETER
5503

Reads and writes the gain setting for the active settings object relative to a camera.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 5503, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 5503, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 5503, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 5503, index_id, object_id)
```

Type

Real variable.

Range

Minimum: 0

Maximum: 32767

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	5503: the value used to reference this property.
index_id	Robot number to select.
object_id	Index of the tool tip to access.
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

VisionOriginBelt

VLOCATION
10052

Origin of the vision frame of reference, which was defined during the calibration. It is expressed as a transform relative to the Belt frame of reference. This property is read-only.

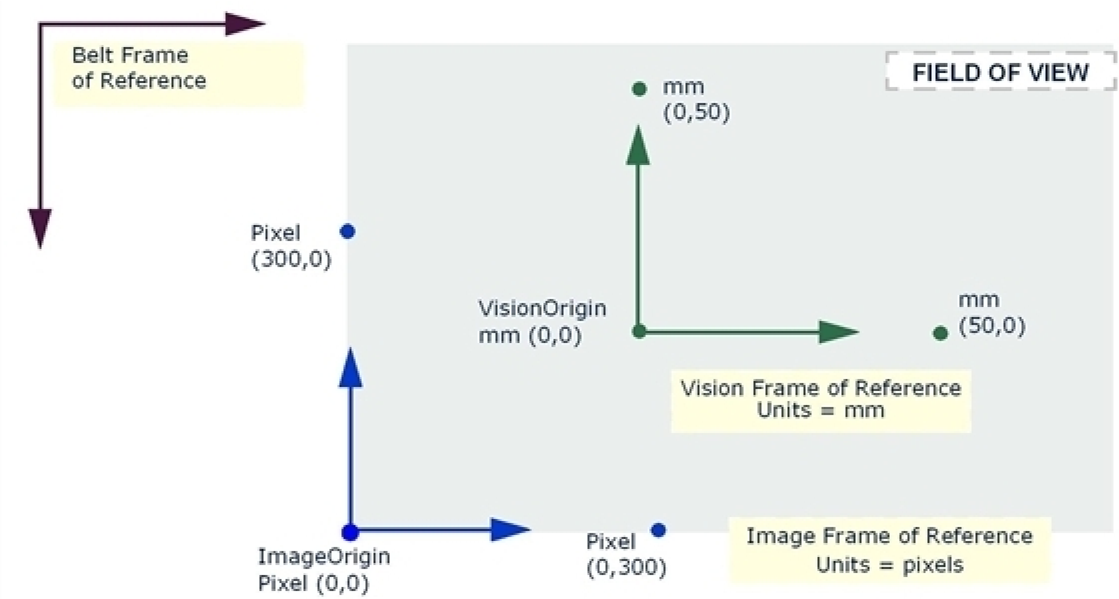


Figure: Illustration of ImageOrigin and VisionOrigin Properties

Syntax

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10052, index_id, frame_id)
```

MicroV+

```
Not applicable. Conveyor tracking is supported only in V+.
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10052: the value used to reference this property.
index	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

Related Properties

ImageOriginBelt
ImageOriginRobot
VisionOriginRobot

VisionOriginRobot

VLOCATION
10050

Origin of the vision frame of reference, which was defined during the calibration. It is expressed as a transform relative to the robot frame of reference. This property is read-only.

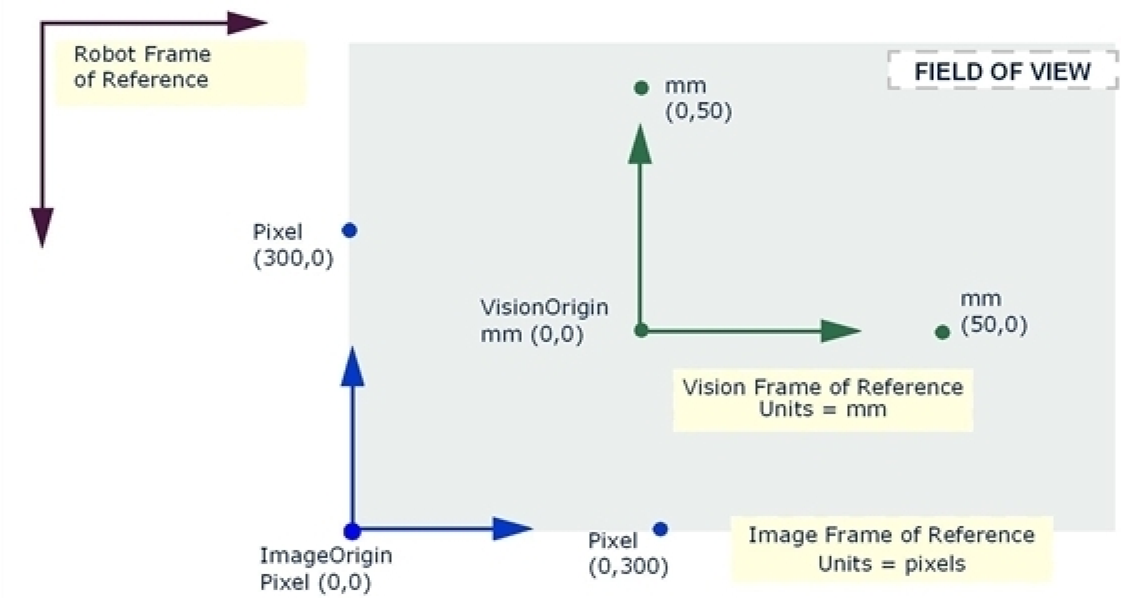


Figure: Illustration of ImageOrigin and VisionOrigin Properties

Syntax

MicroV+

```
value = VLOCATION (sequence_id, tool_id, instance_id, 10050, index_id, frame_id)
```

V+

```
value = VLOCATION ($ip, sequence_id, tool_id, instance_id, 10050, index_id, frame_id)
```

Type

Location

Parameters

$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.
sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
instance_id	Index of the instance for which you want the transform. 1-based.
ID	10050: the value used to reference this property.
index_id	Reserved for internal use. Value is always 1.
frame_id	Index of the frame that contains the specified instance.

Related Properties

ImageOriginBelt
ImageOriginRobot
VisionOriginBelt

VisionRotation

VPARAMETER
10403

Specifies the rotation required to define the InverseKinematics property for a tool-mounted camera. This rotation is defined by the angle between the robot X-axis and the vision X-axis.

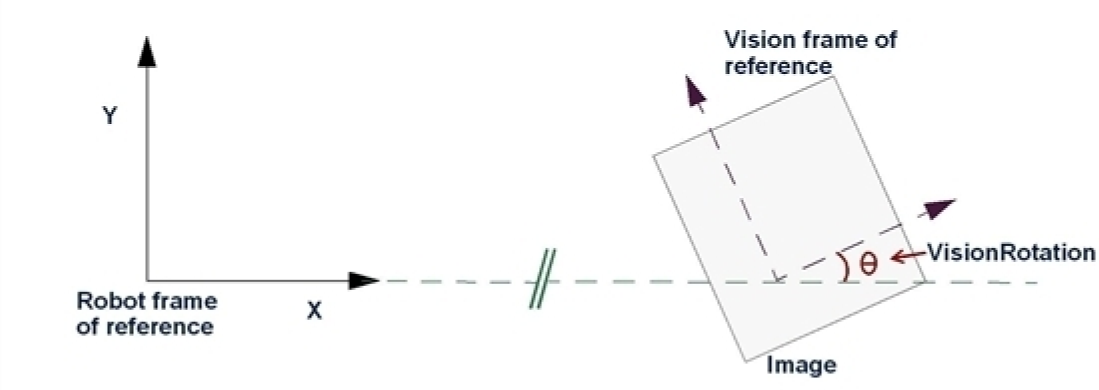


Figure: Illustration of the VisionRotation Angle

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10403, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10403, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10403, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10403, index_id, object_id)
```

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.

ID	10403: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Example

See the InverseKinematics for an example of this property and related properties.

Related Properties

InverseKinematics
RobotXPosition
RobotYPosition
VisionXPosition
VisionYPosition

VisionXPosition

VPARAMETER 10401

X-coordinate (in millimeters) of a position in the vision frame of reference, which is required for the InverseKinematics property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10401, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10401, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10401, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10401, index_id, object_id)
```

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	10401: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Example

See the InverseKinematics for an example of this property and related properties

Related Properties

InverseKinematics

RobotXPosition

VisionRotation
VisionYPosition

VisionYPosition

VPARAMETER
10402

Y-coordinate (in millimeters) of a position in the vision frame of reference, which is required for the InverseKinematics property.

Syntax

MicroV+

```
VPARAMETER (sequence_id, tool_id, 10402, index_id, object_id) = value
value = VPARAMETER (sequence_id, tool_id, 10402, index_id, object_id)
```

V+

```
VPARAMETER (sequence_id, tool_id, 10402, index_id, object_id) $ip = value
value = VPARAMETER ($ip, sequence_id, tool_id, 10402, index_id, object_id)
```

Type

Double

Parameters

sequence_id	Index of the vision sequence. The first sequence is 1.
tool_id	Index of the tool in the vision sequence. The first tool is 1.
ID	10402: the value used to reference this property.
index_id	N/A
object_id	N/A
$ip	IP address of the vision server. Applies to V+ syntax only. Uses standard IP address format, for example: 192.168.1.120.

Example

See the InverseKinematics for an example of this property and related properties

Related Properties

InverseKinematics
RobotXPosition

RobotYPosition
VisionRotation
VisionXPosition

ACE Sight Properties Quick Reference

The following lookup tables provide a quick reference for ACE Sight properties.

- **Property Name**: The name of the property or method.
- **Keyword**: The V+/ MicroV+ keyword that must be used to access the property or method.
- **ID**: The identifier parameter required by the V+/MicroV+ keyword to access the property or method.

There are two ways to locate the information:

- Search for Properties by Name
- Search for Properties by ID

Search for Properties by Name

A B C D E F G

H I K L M N O

P R S T V

Table 1 Global List of ACE Sight Properties - Sorted by Property Name

Property Name	V+/MicroV+ Keyword	Property ID
Abort	VPARAMETER	5501
ActiveCalibration	VPARAMETER	5504
ActiveSettings	VPARAMETER	5505
ArcMustBeTotallyEnclosed	VPARAMETER	5141
ArithmeticClippingMode	VPARAMETER	5360
ArithmeticConstant	VPARAMETER	5361
ArithmeticScale	VPARAMETER	5362
AssignmentConstant	VPARAMETER	5365
AssignmentHeight	VPARAMETER	5366
AssignmentWidth	VPARAMETER	5367

Property Name	V+/MicroV+ Keyword	Property ID
AutoCoarsenessSelectionEnabled	VPARAMETER	5421
AutomaticCandidateCountEnabled	VPARAMETER	5301
AverageContrast	VRESULT	1801
BeltCalibrationDownstreamLimit	VLOCATION	10002
BeltCalibrationFrame	VLOCATION	10000
BeltCalibrationNearsideLimit	VLOCATION	10003
BeltCalibrationScale	VPARAMETER	10004
BeltCalibrationUpstreamLimit	VLOCATION	10001
BeltLatchCalibrationOffset	VLOCATION	10010
BilinearInterpolationEnabled	VPARAMETER	120
BlobArea	VRESULT	1611
BlobBoundingBoxBottom	VRESULT	1648
BlobBoundingBoxCenterX	VRESULT	1624
BlobBoundingBoxCenterY	VRESULT	1625
BlobBoundingBoxHeight	VRESULT	1626
BlobBoundingBoxLeft	VRESULT	1645
BlobBoundingBoxRight	VRESULT	1646
BlobBoundingBoxRotation	VRESULT	1649
BlobBoundingBoxTop	VRESULT	1647
BlobBoundingBoxWidth	VRESULT	1627
BlobChainCode	VRESULT	1656
BlobChainCodeDeltaX	VRESULT	1659
BlobChainCodeDeltaY	VRESULT	1660
BlobChainCodeLength	VRESULT	1655
BlobChainCodeStartX	VRESULT	1657
BlobChainCodeStartY	VRESULT	1658
BlobConvexPerimeter	VRESULT	1614
BlobCount	VRESULT	1610
BlobElongation	VRESULT	1616
BlobExtentBottom	VRESULT	1653

Property Name	V+/MicroV+ Keyword	Property ID
BlobExtentLeft	VRESULT	1650
BlobExtentRight	VRESULT	1651
BlobExtentTop	VRESULT	1652
BlobGreyLevelMaximum	VRESULT	1622
BlobGreyLevelMean	VRESULT	1618
BlobGreyLevelMinimum	VRESULT	1621
BlobGreyLevelRange	VRESULT	1619
BlobGreyLevelStdDev	VRESULT	1620
BlobHoleCount	VRESULT	1654
BlobInertiaMaximum	VRESULT	1633
BlobInertiaMinimum	VRESULT	1632
BlobInertiaXAxis	VRESULT	1634
BlobInertiaYAxis	VRESULT	1635
BlobIntrinsicBoundingBoxBottom	VRESULT	1639
BlobIntrinsicBoundingBoxCenterX	VRESULT	1628
BlobIntrinsicBoundingBoxCenterY	VRESULT	1629
BlobIntrinsicBoundingBoxHeight	VRESULT	1630
BlobIntrinsicBoundingBoxLeft	VRESULT	1636
BlobIntrinsicBoundingBoxRight	VRESULT	1637
BlobIntrinsicBoundingBoxRotation	VRESULT	1640
BlobIntrinsicBoundingBoxTop	VRESULT	1638
BlobIntrinsicBoundingBoxWidth	VRESULT	1631
BlobIntrinsicExtentBottom	VRESULT	1644
BlobIntrinsicExtentLeft	VRESULT	1641
BlobIntrinsicExtentRight	VRESULT	1642
BlobIntrinsicExtentTop	VRESULT	1643
BlobPositionX	VRESULT	1612
BlobPositionY	VRESULT	1613
BlobPrincipalAxesRotation	VRESULT	1617
BlobRawPerimeter	VRESULT	1615

Property Name	V+/MicroV+ Keyword	Property ID
BlobRoundness	VRESULT	1623
CalibratedUnitsEnabled	VPARAMETER	103
CandidatePointsCount	VPARAMETER	5300
ChainCodeResultsEnabled	VPARAMETER	1607
ColorFilterCount	VPARAMETER	5700
ColorFilterMatchPixelCount	VRESULT	2502
ColorFilterMatchQuality	VRESULT	2501
CommunicationToolResults	VPARAMETER	2600
ConformityTolerance	VPARAMETER	556
ConformityToleranceRangeEnabled	VPARAMETER	553
Connectivity	VPARAMETER	5120
Constraints	VPARAMETER	5220
ContrastPolarity	VPARAMETER	522
ContrastThreshold	VPARAMETER	303
ContrastThresholdMode	VPARAMETER	302
DefaultConformityTolerance	VPARAMETER	552
DetailLevel	VPARAMETER	301
Edge1Constraints	VPARAMETER	5221
Edge1Magnitude	VRESULT	1940
Edge1MagnitudeConstraint	VPARAMETER	5227
Edge1MagnitudeScore	VRESULT	1942
Edge1PolarityMode	VPARAMETER	5211
Edge1PositionConstraint	VPARAMETER	5224
Edge1PositionScore	VRESULT	1944
Edge1PositionX	VRESULT	1946
Edge1PositionY	VRESULT	1947
Edge1Radius	VRESULT	1954
Edge1Rotation	VRESULT	1950
Edge1Score	VRESULT	1952
Edge1ScoreThreshold	VPARAMETER	5241

Property Name	V+/MicroV+ Keyword	Property ID
Edge2Constraints	VPARAMETER	5222
Edge2Magnitude	VRESULT	1941
Edge2MagnitudeConstraint	VPARAMETER	5228
Edge2MagnitudeScore	VRESULT	1943
Edge2PolarityMode	VPARAMETER	5212
Edge2PositionConstraint	VPARAMETER	5225
Edge2PositionScore	VRESULT	1945
Edge2PositionX	VRESULT	1948
Edge2PositionY	VRESULT	1949
Edge2Radius	VRESULT	1955
Edge2Rotation	VRESULT	1951
Edge2Score	VRESULT	1953
Edge2ScoreThreshold	VPARAMETER	5242
EdgeCount	VRESULT	1900
EdgeFilterHalfWidth	VPARAMETER	5203
EdgeMagnitude	VRESULT	1901
EdgeMagnitudeScore	VRESULT	1902
EdgeMagnitudeThreshold	VPARAMETER	5201
EdgePolarityMode	VPARAMETER	5210
EdgePositionScore	VRESULT	1903
EdgePositionX	VRESULT	1904
EdgePositionY	VRESULT	1905
EdgeRadius	VRESULT	1908
EdgeRotation	VRESULT	1906
EdgeScore	VRESULT	1907
EdgeSortResultsEnabled	VPARAMETER	5243
ElapsedTime	VRESULT	1001
ExtrinsicInertiaResultsEnabled	VPARAMETER	1604
FilterCount	VPARAMETER	5601
FilterHalfWidth	VPARAMETER	5202

Property Name	V+/MicroV+ Keyword	Property ID
FilterHueTolerance	VPARAMETER	5716
FilterHueValue	VPARAMETER	5713
FilteringClippingMode	VPARAMETER	5370
FilteringKernelSize	VPARAMETER	5371
FilteringScale	VPARAMETER	5372
FilterLuminanceTolerance	VPARAMETER	5718
FilterLuminanceValue	VPARAMETER	5715
FilterSaturationTolerance	VPARAMETER	5717
FilterSaturationValue	VPARAMETER	5714
FitMode	VPARAMETER	5140
FitQuality	VRESULT	1803
Found	VRESULT	1800
FrameCount	VRESULT	2410
FrameIntrinsicBoundingBox	VRESULT	2420
FrameRotation	VRESULT	2402
FrameTranslationX	VRESULT	2400
FrameTranslationY	VRESULT	2401
GreylevelRange	VRESULT	1508
GreyLevelResultsEnabled	VPARAMETER	1608
GripperInputClose	VPARAMETER	5515
GripperInputExtend	VPARAMETER	5518
GripperInputOpen	VPARAMETER	5514
GripperInputRetract	VPARAMETER	5519
GripperOffset	VLOCATION	10100
GripperOutputClose	VPARAMETER	5512
GripperOutputExtend	VPARAMETER	5516
GripperOutputOpen	VPARAMETER	5511
GripperOutputRelease	VPARAMETER	5513
GripperOutputRetract	VPARAMETER	5517
GripperToolTransform	VLOCATION	11000
Histogram	VRESULT	1511

Property Name	V+/MicroV+ Keyword	Property ID
HistogramPixelCount	VRESULT	1512
HistogramThreshold	VPARAMETER	5385
HoleFillingEnabled	VPARAMETER	5002
ImageHeight	VRESULT	1021
ImageOriginBelt	VLOCATION	10053
ImageOriginRobot	VLOCATION	10051
ImagePixelCount	VRESULT	1513
ImageSubsampling	VPARAMETER	5324
ImageWidth	VRESULT	1020
InspectionFilterMeasuredValue	VRESULT	2700
InspectionFilterPassStatus	VRESULT	2702
InstanceClearQuality	VRESULT	1319
InstanceCount	VRESULT	1310
InstanceFitQuality	VRESULT	1317
InstanceIntrinsicBoundingBox	VRESULT	1330
InstanceLocation	VLOCATION	1311
InstanceLocationGripperOffsetMaximum	VLOCATION	1499
InstanceLocationGripperOffsetMinimum	VLOCATION	1400
InstanceMatchQuality	VRESULT	1318
InstanceModel	VRESULT	1312
InstanceOrdering	VPARAMETER	530
InstanceOrderingReferenceX	VPARAMETER	531
InstanceOrderingReferenceY	VPARAMETER	532
InstanceRobotLocation	VLOCATION	1371
InstanceRotation	VRESULT	1314
InstanceScaleFactor	VRESULT	1313
InstanceSymmetry	VRESULT	1320
InstanceTime	VRESULT	1322
InstanceToolOffset	VLOCATION	1372
InstanceTranslationX	VRESULT	1315

Property Name	V+/MicroV+ Keyword	Property ID
InstanceTranslationY	VRESULT	1316
InstanceVisible	VRESULT	1321
InstanceVisionOffset	VLOCATION	1373
InterpolatePositionMode	VPARAMETER	5122
InterpolatePositionModeEnabled	VPARAMETER	5123
IntrinsicBoxResultsEnabled	VPARAMETER	1605
InverseKinematics	VPARAMETER	10060
KernelSize	VPARAMETER	5304
LastOperation	VRESULT	2200
LastOutputType	VRESULT	2201
LogicalConstant	VPARAMETER	5380
MagnitudeConstraint	VPARAMETER	5226
MagnitudeThreshold	VPARAMETER	5200
MatchCount	VRESULT	2100
MatchPositionX	VRESULT	2102
MatchPositionY	VRESULT	2103
MatchQuality	VRESULT	1802
MatchRotation	VRESULT	2104
MatchStrength	VRESULT	2101
MatchThreshold	VPARAMETER	5420
MaximumAngleDeviation	VPARAMETER	5102
MaximumBlobArea	VPARAMETER	5001
MaximumGreylevelValue	VRESULT	1507
MaximumInstanceCount	VPARAMETER	519
MaximumInstanceCountEnabled	VPARAMETER	518
MaximumRotation	VPARAMETER	517
MaximumScaleFactor	VPARAMETER	513
Mean	VRESULT	1500
MeasurementPointsCount	VRESULT	2002
Median	VRESULT	1501

Property Name	V+/MicroV+ Keyword	Property ID
MinimumArcPercentage	VPARAMETER	5142
MinimumBlobArea	VPARAMETER	5000
MinimumClearPercentage	VPARAMETER	559
MinimumClearPercentageEnabled	VPARAMETER	558
MinimumGreylevelValue	VRESULT	1506
MinimumLinePercentage	VPARAMETER	5130
MinimumModelPercentage	VPARAMETER	557
MinimumRequiredFeatures	VPARAMETER	560
MinimumRotation	VPARAMETER	516
MinimumScaleFactor	VPARAMETER	512
Mode	VRESULT	1504
ModelDisambiguationEnabled	VPARAMETER	403
ModePixelCount	VRESULT	1505
MorphologicalNeighborhoodSize	VPARAMETER	5390
NominalRotation	VPARAMETER	515
NominalRotationEnabled	VPARAMETER	514
NominalScaleFactor	VPARAMETER	511
NominalScaleFactorEnabled	VPARAMETER	510
Operation	VPARAMETER	5355
Operator	VPARAMETER	5600
OutlineLevel	VPARAMETER	300
OutputArcAngle	VRESULT	1841
OutputArcCenterPointX	VRESULT	1846
OutputArcCenterPointY	VRESULT	1847
OutputArcRadius	VRESULT	1840
OutputLineAngle	VRESULT	1820
OutputLineEndPointX	VRESULT	1823
OutputLineEndPointY	VRESULT	1824
OutputLineStartPointX	VRESULT	1821
OutputLineStartPointY	VRESULT	1822

Property Name	V+/MicroV+ Keyword	Property ID
OutputLineVectorPointX	VRESULT	1825
OutputLineVectorPointY	VRESULT	1826
OutputPointX	VRESULT	1810
OutputPointY	VRESULT	1811
OutputSymmetricInstances	VPARAMETER	520
OverrideType	VPARAMETER	5351
OverrideTypeEnabled	VPARAMETER	5350
PairCount	VRESULT	1920
PairPositionX	VRESULT	1921
PairPositionY	VRESULT	1922
PairRotation	VRESULT	1923
PairScore	VRESULT	1924
PairSize	VRESULT	1925
ParametersBasedOn	VPARAMETER	304
PatternHeight	VPARAMETER	5403
PatternPositionX	VPARAMETER	5400
PatternPositionY	VPARAMETER	5401
PatternRotation	VPARAMETER	5404
PatternWidth	VPARAMETER	5402
PerimeterResultsEnabled	VPARAMETER	1602
PolarityMode	VPARAMETER	5100
PositionConstraint	VPARAMETER	5223
PositioningLevel	VPARAMETER	561
ProjectionMode	VPARAMETER	140
RecognitionLevel	VPARAMETER	550
Reset	VPARAMETER	5500
ResultCount	VRESULT	1010
RobotXPosition	VPARAMETER	10404
RobotYPosition	VPARAMETER	10405
SamplingStepCustom	VPARAMETER	124

Property Name	V+/MicroV+ Keyword	Property ID
SamplingStepCustomEnabled	VPARAMETER	121
SaveImage	VPARAMETER	10327
ScoreThreshold	VPARAMETER	5240
SearchBasedOnOutlineLevelOnly	VPARAMETER	521
SearchCoarseness	VPARAMETER	5430
SearchMode	VPARAMETER	5101
SegmentationDark	VPARAMETER	5005
SegmentationDynamicDark	VPARAMETER	5009
SegmentationDynamicInside	VPARAMETER	5010
SegmentationDynamicLight	VPARAMETER	5008
SegmentationDynamicOutside	VPARAMETER	5011
SegmentationInside	VPARAMETER	5006
SegmentationLight	VPARAMETER	5004
SegmentationMode	VPARAMETER	5003
SegmentationOutside	VPARAMETER	5007
SequenceExecutionMode	VPARAMETER	10200
Sharpness	VRESULT	2000
SharpnessPeak	VRESULT	2001
ShowResultsGraphics	VPARAMETER	150
SortBlobsBy	VPARAMETER	1601
SortResultsEnabled	VPARAMETER	1600
StandardDeviation	VRESULT	1503
StandardDeviationThreshold	VPARAMETER	5302
SubsamplingLevel	VPARAMETER	5110
TailBlack	VPARAMETER	5323
TailBlackGreylevelValue	VRESULT	1509
TailWhite	VPARAMETER	5322
TailWhiteGreylevelValue	VRESULT	1510
ThresholdBlack	VPARAMETER	5320
ThresholdWhite	VPARAMETER	5321

Property Name	V+/MicroV+ Keyword	Property ID
Timeout	VPARAMETER	501
TimeoutEnabled	VPARAMETER	500
ToolGuidelineOffset	VPARAMETER	130
ToolHeight	VPARAMETER	111
ToolOpening	VPARAMETER	137
ToolPositionX	VPARAMETER	100
ToolPositionY	VPARAMETER	101
ToolRadius	VPARAMETER	135
ToolRotation	VPARAMETER	112
ToolSkew	VPARAMETER	113
ToolThickness	VPARAMETER	136
ToolWidth	VPARAMETER	110
TopologicalResultsEnabled	VPARAMETER	1609
TransformFlags	VPARAMETER	5395
Variance	VRESULT	1502
VideoExposure	VPARAMETER	5502
VideoGain	VPARAMETER	5503
VisionOriginBelt	VLOCATION	10052
VisionOriginRobot	VLOCATION	10050
VisionRotation	VPARAMETER	10403
VisionXPosition	VPARAMETER	10401
VisionYPosition	VPARAMETER	10402

Search for Properties by ID

Table 2 Global List of ACE Sight Properties - Sorted by Property ID

Property ID	Property Name	V+/MicroV+ Keyword
100	ToolPositionX	VPARAMETER
101	ToolPositionY	VPARAMETER
103	CalibratedUnitsEnabled	VPARAMETER
110	ToolWidth	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
111	ToolHeight	VPARAMETER
112	ToolRotation	VPARAMETER
113	ToolSkew	VPARAMETER
120	BilinearInterpolationEnabled	VPARAMETER
121	SamplingStepCustomEnabled	VPARAMETER
124	SamplingStepCustom	VPARAMETER
130	ToolGuidelineOffset	VPARAMETER
135	ToolRadius	VPARAMETER
136	ToolThickness	VPARAMETER
137	ToolOpening	VPARAMETER
140	ProjectionMode	VPARAMETER
150	ShowResultsGraphics	VPARAMETER
300	OutlineLevel	VPARAMETER
301	DetailLevel	VPARAMETER
302	ContrastThresholdMode	VPARAMETER
303	ContrastThreshold	VPARAMETER
304	ParametersBasedOn	VPARAMETER
403	ModelDisambiguationEnabled	VPARAMETER
500	TimeoutEnabled	VPARAMETER
501	Timeout	VPARAMETER
510	NominalScaleFactorEnabled	VPARAMETER
511	NominalScaleFactor	VPARAMETER
512	MinimumScaleFactor	VPARAMETER
513	MaximumScaleFactor	VPARAMETER
514	NominalRotationEnabled	VPARAMETER
515	NominalRotation	VPARAMETER
516	MinimumRotation	VPARAMETER
517	MaximumRotation	VPARAMETER
518	MaximumInstanceCountEnabled	VPARAMETER
519	MaximumInstanceCount	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
520	OutputSymmetricInstances	VPARAMETER
521	SearchBasedOnOutlineLevelOnly	VPARAMETER
522	ContrastPolarity	VPARAMETER
530	InstanceOrdering	VPARAMETER
531	InstanceOrderingReferenceX	VPARAMETER
532	InstanceOrderingReferenceY	VPARAMETER
550	RecognitionLevel	VPARAMETER
552	DefaultConformityTolerance	VPARAMETER
553	ConformityToleranceRangeEnabled	VPARAMETER
556	ConformityTolerance	VPARAMETER
557	MinimumModelPercentage	VPARAMETER
558	MinimumClearPercentageEnabled	VPARAMETER
559	MinimumClearPercentage	VPARAMETER
560	MinimumRequiredFeatures	VPARAMETER
561	PositioningLevel	VPARAMETER
1001	ElapsedTime	VRESULT
1010	ResultCount	VRESULT
1020	ImageWidth	VRESULT
1021	ImageHeight	VRESULT
1310	InstanceCount	VRESULT
1311	InstanceLocation	VLOCATION
1312	InstanceModel	VRESULT
1313	InstanceScaleFactor	VRESULT
1314	InstanceRotation	VRESULT
1315	InstanceTranslationX	VRESULT
1316	InstanceTranslationY	VRESULT
1317	InstanceFitQuality	VRESULT
1318	InstanceMatchQuality	VRESULT
1319	InstanceClearQuality	VRESULT
1320	InstanceSymmetry	VRESULT

Property ID	Property Name	V+/MicroV+ Keyword
1321	InstanceVisible	VRESULT
1322	InstanceTime	VRESULT
1330	InstanceIntrinsicBoundingBox	VRESULT
1371	InstanceRobotLocation	VLOCATION
1372	InstanceToolOffset	VLOCATION
1373	InstanceVisionOffset	VLOCATION
1400	InstanceLocationGripperOffsetMinimum	VLOCATION
1499	InstanceLocationGripperOffsetMaximum	VLOCATION
1500	Mean	VRESULT
1501	Median	VRESULT
1502	Variance	VRESULT
1503	StandardDeviation	VRESULT
1504	Mode	VRESULT
1505	ModePixelCount	VRESULT
1506	MinimumGreylevelValue	VRESULT
1507	MaximumGreylevelValue	VRESULT
1508	GreylevelRange	VRESULT
1509	TailBlackGreylevelValue	VRESULT
1510	TailWhiteGreylevelValue	VRESULT
1511	Histogram	VRESULT
1512	HistogramPixelCount	VRESULT
1513	ImagePixelCount	VRESULT
1600	SortResultsEnabled	VPARAMETER
1601	SortBlobsBy	VPARAMETER
1602	PerimeterResultsEnabled	VPARAMETER
1604	ExtrinsicInertiaResultsEnabled	VPARAMETER
1605	IntrinsicBoxResultsEnabled	VPARAMETER
1607	ChainCodeResultsEnabled	VPARAMETER
1608	GreyLevelResultsEnabled	VPARAMETER
1609	TopologicalResultsEnabled	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
1610	BlobCount	VRESULT
1611	BlobArea	VRESULT
1612	BlobPositionX	VRESULT
1613	BlobPositionY	VRESULT
1614	BlobConvexPerimeter	VRESULT
1615	BlobRawPerimeter	VRESULT
1616	BlobElongation	VRESULT
1617	BlobPrincipalAxesRotation	VRESULT
1618	BlobGreyLevelMean	VRESULT
1619	BlobGreyLevelRange	VRESULT
1620	BlobGreyLevelStdDev	VRESULT
1621	BlobGreyLevelMinimum	VRESULT
1622	BlobGreyLevelMaximum	VRESULT
1623	BlobRoundness	VRESULT
1624	BlobBoundingBoxCenterX	VRESULT
1625	BlobBoundingBoxCenterY	VRESULT
1626	BlobBoundingBoxHeight	VRESULT
1627	BlobBoundingBoxWidth	VRESULT
1628	BlobIntrinsicBoundingBoxCenterX	VRESULT
1629	BlobIntrinsicBoundingBoxCenterY	VRESULT
1630	BlobIntrinsicBoundingBoxHeight	VRESULT
1631	BlobIntrinsicBoundingBoxWidth	VRESULT
1632	BlobInertiaMinimum	VRESULT
1633	BlobInertiaMaximum	VRESULT
1634	BlobInertiaXAxis	VRESULT
1635	BlobInertiaYAxis	VRESULT
1636	BlobIntrinsicBoundingBoxLeft	VRESULT
1637	BlobIntrinsicBoundingBoxRight	VRESULT
1638	BlobIntrinsicBoundingBoxTop	VRESULT
1639	BlobIntrinsicBoundingBoxBottom	VRESULT

Property ID	Property Name	V+/MicroV+ Keyword
1640	BlobIntrinsicBoundingBoxRotation	VRESULT
1641	BlobIntrinsicExtentLeft	VRESULT
1642	BlobIntrinsicExtentRight	VRESULT
1643	BlobIntrinsicExtentTop	VRESULT
1644	BlobIntrinsicExtentBottom	VRESULT
1645	BlobBoundingBoxLeft	VRESULT
1646	BlobBoundingBoxRight	VRESULT
1647	BlobBoundingBoxTop	VRESULT
1648	BlobBoundingBoxBottom	VRESULT
1649	BlobBoundingBoxRotation	VRESULT
1650	BlobExtentLeft	VRESULT
1651	BlobExtentRight	VRESULT
1652	BlobExtentTop	VRESULT
1653	BlobExtentBottom	VRESULT
1654	BlobHoleCount	VRESULT
1655	BlobChainCodeLength	VRESULT
1656	BlobChainCode	VRESULT
1657	BlobChainCodeStartX	VRESULT
1658	BlobChainCodeStartY	VRESULT
1659	BlobChainCodeDeltaX	VRESULT
1660	BlobChainCodeDeltaY	VRESULT
1800	Found	VRESULT
1801	AverageContrast	VRESULT
1802	MatchQuality	VRESULT
1803	FitQuality	VRESULT
1810	OutputPointX	VRESULT
1811	OutputPointY	VRESULT
1820	OutputLineAngle	VRESULT
1821	OutputLineStartPointX	VRESULT
1822	OutputLineStartPointY	VRESULT

Property ID	Property Name	V+/MicroV+ Keyword
1823	OutputLineEndPointX	VRESULT
1824	OutputLineEndPointY	VRESULT
1825	OutputLineVectorPointX	VRESULT
1826	OutputLineVectorPointY	VRESULT
1840	OutputArcRadius	VRESULT
1841	OutputArcAngle	VRESULT
1846	OutputArcCenterPointX	VRESULT
1847	OutputArcCenterPointY	VRESULT
1900	EdgeCount	VRESULT
1901	EdgeMagnitude	VRESULT
1902	EdgeMagnitudeScore	VRESULT
1903	EdgePositionScore	VRESULT
1904	EdgePositionX	VRESULT
1905	EdgePositionY	VRESULT
1906	EdgeRotation	VRESULT
1907	EdgeScore	VRESULT
1908	EdgeRadius	VRESULT
1920	PairCount	VRESULT
1921	PairPositionX	VRESULT
1922	PairPositionY	VRESULT
1923	PairRotation	VRESULT
1924	PairScore	VRESULT
1925	PairSize	VRESULT
1940	Edge1Magnitude	VRESULT
1941	Edge2Magnitude	VRESULT
1942	Edge1MagnitudeScore	VRESULT
1943	Edge2MagnitudeScore	VRESULT
1944	Edge1PositionScore	VRESULT
1945	Edge2PositionScore	VRESULT
1946	Edge1PositionX	VRESULT

Property ID	Property Name	V+/MicroV+ Keyword
1947	Edge1PositionY	VRESULT
1948	Edge2PositionX	VRESULT
1949	Edge2PositionY	VRESULT
1950	Edge1Rotation	VRESULT
1951	Edge2Rotation	VRESULT
1952	Edge1Score	VRESULT
1953	Edge2Score	VRESULT
1954	Edge1Radius	VRESULT
1955	Edge2Radius	VRESULT
2000	Sharpness	VRESULT
2001	SharpnessPeak	VRESULT
2002	MeasurementPointsCount	VRESULT
2100	MatchCount	VRESULT
2101	MatchStrength	VRESULT
2102	MatchPositionX	VRESULT
2103	MatchPositionY	VRESULT
2104	MatchRotation	VRESULT
2200	LastOperation	VRESULT
2201	LastOutputType	VRESULT
2400	FrameTranslationX	VRESULT
2401	FrameTranslationY	VRESULT
2402	FrameRotation	VRESULT
2410	FrameCount	VRESULT
2420	FrameIntrinsicBoundingBox	VRESULT
2501	ColorFilterMatchQuality	VRESULT
2502	ColorFilterMatchPixelCount	VRESULT
2600	CommunicationToolResults	VPARAMETER
2700	InspectionFilterMeasuredValue	VRESULT
2702	InspectionFilterPassStatus	VRESULT
5000	MinimumBlobArea	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
5001	MaximumBlobArea	VPARAMETER
5002	HoleFillingEnabled	VPARAMETER
5003	SegmentationMode	VPARAMETER
5004	SegmentationLight	VPARAMETER
5005	SegmentationDark	VPARAMETER
5006	SegmentationInside	VPARAMETER
5007	SegmentationOutside	VPARAMETER
5008	SegmentationDynamicLight	VPARAMETER
5009	SegmentationDynamicDark	VPARAMETER
5010	SegmentationDynamicInside	VPARAMETER
5011	SegmentationDynamicOutside	VPARAMETER
5100	PolarityMode	VPARAMETER
5101	SearchMode	VPARAMETER
5102	MaximumAngleDeviation	VPARAMETER
5110	SubsamplingLevel	VPARAMETER
5120	Connectivity	VPARAMETER
5122	InterpolatePositionMode	VPARAMETER
5123	InterpolatePositionModeEnabled	VPARAMETER
5130	MinimumLinePercentage	VPARAMETER
5140	FitMode	VPARAMETER
5141	ArcMustBeTotallyEnclosed	VPARAMETER
5142	MinimumArcPercentage	VPARAMETER
5200	MagnitudeThreshold	VPARAMETER
5201	EdgeMagnitudeThreshold	VPARAMETER
5202	FilterHalfWidth	VPARAMETER
5203	EdgeFilterHalfWidth	VPARAMETER
5210	EdgePolarityMode	VPARAMETER
5211	Edge1PolarityMode	VPARAMETER
5212	Edge2PolarityMode	VPARAMETER
5220	Constraints	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
5221	Edge1Constraints	VPARAMETER
5222	Edge2Constraints	VPARAMETER
5223	PositionConstraint	VPARAMETER
5224	Edge1PositionConstraint	VPARAMETER
5225	Edge2PositionConstraint	VPARAMETER
5226	MagnitudeConstraint	VPARAMETER
5227	Edge1MagnitudeConstraint	VPARAMETER
5228	Edge2MagnitudeConstraint	VPARAMETER
5240	ScoreThreshold	VPARAMETER
5241	Edge1ScoreThreshold	VPARAMETER
5242	Edge2ScoreThreshold	VPARAMETER
5243	EdgeSortResultsEnabled	VPARAMETER
5300	CandidatePointsCount	VPARAMETER
5301	AutomaticCandidateCountEnabled	VPARAMETER
5302	StandardDeviationThreshold	VPARAMETER
5304	KernelSize	VPARAMETER
5320	ThresholdBlack	VPARAMETER
5321	ThresholdWhite	VPARAMETER
5322	TailWhite	VPARAMETER
5323	TailBlack	VPARAMETER
5324	ImageSubsampling	VPARAMETER
5350	OverrideTypeEnabled	VPARAMETER
5351	OverrideType	VPARAMETER
5355	Operation	VPARAMETER
5360	ArithmeticClippingMode	VPARAMETER
5361	ArithmeticConstant	VPARAMETER
5362	ArithmeticScale	VPARAMETER
5365	AssignmentConstant	VPARAMETER
5366	AssignmentHeight	VPARAMETER
5367	AssignmentWidth	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
5370	FilteringClippingMode	VPARAMETER
5371	FilteringKernelSize	VPARAMETER
5372	FilteringScale	VPARAMETER
5380	LogicalConstant	VPARAMETER
5385	HistogramThreshold	VPARAMETER
5390	MorphologicalNeighborhoodSize	VPARAMETER
5395	TransformFlags	VPARAMETER
5400	PatternPositionX	VPARAMETER
5401	PatternPositionY	VPARAMETER
5402	PatternWidth	VPARAMETER
5403	PatternHeight	VPARAMETER
5404	PatternRotation	VPARAMETER
5420	MatchThreshold	VPARAMETER
5421	AutoCoarsenessSelectionEnabled	VPARAMETER
5430	SearchCoarseness	VPARAMETER
5500	Reset	VPARAMETER
5501	Abort	VPARAMETER
5502	VideoExposure	VPARAMETER
5503	VideoGain	VPARAMETER
5504	ActiveCalibration	VPARAMETER
5505	ActiveSettings	VPARAMETER
5511	GripperOutputOpen	VPARAMETER
5512	GripperOutputClose	VPARAMETER
5513	GripperOutputRelease	VPARAMETER
5514	GripperInputOpen	VPARAMETER
5515	GripperInputClose	VPARAMETER
5516	GripperOutputExtend	VPARAMETER
5517	GripperOutputRetract	VPARAMETER
5518	GripperInputExtend	VPARAMETER
5519	GripperInputRetract	VPARAMETER

Property ID	Property Name	V+/MicroV+ Keyword
5600	Operator	VPARAMETER
5601	FilterCount	VPARAMETER
5700	ColorFilterCount	VPARAMETER
5713	FilterHueValue	VPARAMETER
5714	FilterSaturationValue	VPARAMETER
5715	FilterLuminanceValue	VPARAMETER
5716	FilterHueTolerance	VPARAMETER
5717	FilterSaturationTolerance	VPARAMETER
5718	FilterLuminanceTolerance	VPARAMETER
10000	BeltCalibrationFrame	VLOCATION
10001	BeltCalibrationUpstreamLimit	VLOCATION
10002	BeltCalibrationDownstreamLimit	VLOCATION
10003	BeltCalibrationNearsideLimit	VLOCATION
10004	BeltCalibrationScale	VPARAMETER
10010	BeltLatchCalibrationOffset	VLOCATION
10050	VisionOriginRobot	VLOCATION
10051	ImageOriginRobot	VLOCATION
10052	VisionOriginBelt	VLOCATION
10053	ImageOriginBelt	VLOCATION
10060	InverseKinematics	VPARAMETER
10100	GripperOffset	VLOCATION
10200	SequenceExecutionMode	VPARAMETER
10201	SequenceToolCount	VPARAMETER
10327	SaveImage	VPARAMETER
10401	VisionXPosition	VPARAMETER
10402	VisionYPosition	VPARAMETER
10403	VisionRotation	VPARAMETER
10404	RobotXPosition	VPARAMETER
10405	RobotYPosition	VPARAMETER
11000	GripperToolTransform	VLOCATION

ACE Sight Framework Properties

ACE Sight framework properties are properties that do not apply to vision tools.

The table below lists the ACE Sight properties that apply to the ACE Sight framework. Click on a property name to access the reference for the property.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BeltCalibrationDownstreamLimit	VLOCATION	10002	R - O	location
BeltCalibrationFrame	VLOCATION	10000	R - O	location
BeltCalibrationNearsideLimit	VLOCATION	10003	R - O	location
BeltCalibrationScale	VPARAMETER	10004	R - O	Double
BeltCalibrationUpstreamLimit	VLOCATION	10001	R - O	location
BeltLatchCalibrationOffset	VLOCATION	10010	R - O	location
GripperOffset	VLOCATION	10100	R - O	location
ImageOriginBelt	VLOCATION	10053	R - O	location
ImageOriginRobot	VLOCATION	10051	R - O	location
InstanceLocation	VLOCATION	1311	R - O	location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	location
InstanceRobotLocation	VLOCATION	1371	R - O	location
BlobExtentBottom	VLOCATION	1653	R - O	location
BlobExtentLeft	VLOCATION	1650	R - O	location
BlobExtentRight	VLOCATION	1651	R - O	location
BlobExtentTop	VLOCATION	1652	R - O	location
RobotXPosition	VPARAMETER	10404	R - W	Double
RobotYPosition	VPARAMETER	10405	R - W	Double
SaveImage	VPARAMETER	10327	R - W	Long
SequenceExecutionMode	VPARAMETER	10200	R - W	Long
SequenceToolCount	VPARAMETER	10201	R-O	Long
VisionOriginBelt	VLOCATION	10052	R-O	location
VisionOriginRobot	VLOCATION	10050	R-O	location
VisionRotation	VPARAMETER	10403	R - W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
VisionXPosition	VPARAMETER	10401	R - W	Double
VisionYPosition	VPARAMETER	10402	R - W	Double

ACE Sight Tool Properties

The following sections list the properties that apply to the selected vision tool.

Arc Caliper Properties

AnyFeeder Properties

Arc Edge Locator Properties

Arc Finder Properties

Blob Analyzer Properties

Calculated Arc Properties

Calculated Frame Properties

Calculated Line Properties

Calculated Point Properties

Calibration Grid Locator Properties

Caliper Properties

Color Matching Tool Properties

Communication Tool Properties

Custom Vision Tool Properties

Edge Locator Properties

Gripper Clearance Tool Properties

Image Histogram Tool Properties

Image Processing Tool Properties

Image Sampling Tool Properties

Image Sharpness Tool Properties

Inspection Tool Properties

Line Finder Properties

Locator Model Tool Properties (**NOTE:** This tool cannot be accessed by ACE Sight)

Locator Tool Properties

Overlap Tool Properties

Pattern Locator Properties

Point Finder Properties

Remote Vision Tool Properties

Virtual Camera Tool Properties

AnyFeeder Properties

The AnyFeeder manages and controls the AnyFeeder hardware.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
AnyFeederVRunCommand	VPARAMETER	6000	R / W	Long
AnyFeederDispenseIterations	VPARAMETER	6012	R / W	Long
AnyFeederDispenseSpeed	VPARAMETER	6002	R / W	Long
AnyFeederFeedBackwardIterations	VPARAMETER	6015	R / W	Long
AnyFeederFeedBackwardSpeed	VPARAMETER	6005	R / W	Long
AnyFeederFeedFlipBackwardIterations	VPARAMETER	6018	R / W	Long
AnyFeederFeedFlipBackwardSpeed	VPARAMETER	6005	R / W	Long
AnyFeederFeedFlipForwardIterations	VPARAMETER	6017	R / W	Long
AnyFeederFeedFlipForwardSpeed	VPARAMETER	6007	R / W	Long
AnyFeederFeedForwardIterations	VPARAMETER	6011	R / W	Long
AnyFeederFeedForwardSpeed	VPARAMETER	6001	R / W	Long
AnyFeederFlipIterations	VPARAMETER	6013	R / W	Long
AnyFeederFlipSpeed	VPARAMETER	6003	R / W	Long
AnyFeederHeavyDispenseIterations	VPARAMETER	6016	R / W	Long
AnyFeederHeavyDispenseSpeed	VPARAMETER	6006	R / W	Long
AnyFeederPurgeIterations	VPARAMETER	6014	R / W	Long
AnyFeederPurgeSpeed	VPARAMETER	6004	R / W	Long

Arc Caliper Properties

The Arc Caliper finds and locates one or more edge pairs on an arc-shaped or circular area and measures distances between the two edges within each pair.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
Edge1Constraints	VPARAMETER	5221	R / W	Long
Edge1Magnitude	VRESULT	1940	R - O	Double
Edge1MagnitudeConstraint	VPARAMETER	5227	R / W	Long
Edge1MagnitudeScore	VRESULT	1942	R - O	Double
Edge1PolarityMode	VPARAMETER	5211	R / W	Long
Edge1PositionConstraint	VPARAMETER	5224	R / W	Double
Edge1PositionScore	VRESULT	1944	R - O	Double
Edge1PositionX	VRESULT	1946	R - O	Double
Edge1PositionY	VRESULT	1947	R - O	Double
Edge1Radius	VRESULT	1954	R - O	Double
Edge1Rotation	VRESULT	1950	R - O	Double
Edge1Score	VRESULT	1952	R - O	Double
Edge1ScoreThreshold	VPARAMETER	5241	R / W	Double
Edge2Constraints	VPARAMETER	5222	R / W	Long
Edge2Magnitude	VRESULT	1941	R - O	Double
Edge2MagnitudeConstraint	VPARAMETER	5228	R / W	Long
Edge2MagnitudeScore	VRESULT	1943	R - O	Double
Edge2PolarityMode	VPARAMETER	5212	R / W	Long
Edge2PositionConstraint	VPARAMETER	5225	R / W	Double
Edge2PositionScore	VRESULT	1945	R - O	Double
Edge2PositionX	VRESULT	1948	R - O	Double
Edge2PositionY	VRESULT	1949	R - O	Double
Edge2Radius	VRESULT	1955	R - O	Double
Edge2Rotation	VRESULT	1951	R - O	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
Edge2Score	VRESULT	1953	R - O	Double
Edge2ScoreThreshold	VPARAMETER	5242	R / W	Double
EdgeFilterHalfWidth	VPARAMETER	5203	R / W	Long
EdgeMagnitudeThreshold	VPARAMETER	5201	R / W	Double
ElapsedTime	VRESULT	1001	R - O	Double
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
PairCount	VRESULT	1920	R - O	Long
PairPositionX	VRESULT	1921	R - O	Double
PairPositionY	VRESULT	1922	R - O	Double
PairRotation	VRESULT	1923	R - O	Double
PairScore	VRESULT	1924	R - O	Double
PairSize	VRESULT	1925	R - O	Double
ProjectionMode	VPARAMETER	140	R / W	Long
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolOpening	VPARAMETER	137	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRadius	VPARAMETER	135	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolThickness	VPARAMETER	136	R / W	Double

Arc Edge Locator Properties

The Arc Edge Locator finds and locates an edge or a set of edges in an arc-shaped or circular area.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
Constraints	VPARAMETER	5220	R / W	Long
EdgeCount	VRESULT	1900	R - O	Long
EdgeMagnitude	VRESULT	1901	R - O	Double
EdgeMagnitudeScore	VRESULT	1902	R - O	Double
EdgePolarityMode	VPARAMETER	5210	R / W	Long
EdgePositionScore	VRESULT	1903	R - O	Double
EdgePositionX	VRESULT	1904	R - O	Double
EdgePositionY	VRESULT	1905	R - O	Double
EdgeRadius	VRESULT	1908	R - O	Double
EdgeRotation	VRESULT	1906	R - O	Double
EdgeScore	VRESULT	1907	R - O	Double
EdgeSortResultsEnabled	VPARAMETER	5243	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
FilterHalfWidth	VPARAMETER	5202	R / W	Long
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
MagnitudeConstraint	VPARAMETER	5226	R / W	Long
MagnitudeThreshold	VPARAMETER	5200	R / W	Double
PositionConstraint	VPARAMETER	5223	R / W	Double
ProjectionMode	VPARAMETER	140	R / W	Long

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
ScoreThreshold	VPARAMETER	5240	R / W	Double
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolOpening	VPARAMETER	137	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRadius	VPARAMETER	135	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolThickness	VPARAMETER	136	R / W	Double

Arc Finder Properties

The Arc Finder finds and locates circular features on objects and returns the coordinates of the center of the arc, the start and end angles, and the radius.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ArcMustBeTotallyEnclosed	VPARAMETER	5141	R / W	Boolean
AverageContrast	VRESULT	1801	R - O	Double
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ConformityTolerance	VPARAMETER	556	R / W	Double
ContrastThreshold	VPARAMETER	303	R / W	Long
ContrastThresholdMode	VPARAMETER	302	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
FitMode	VPARAMETER	5140	R / W	Long
FitQuality	VRESULT	1803	R - O	Double
Found	VRESULT	1800	R - O	Boolean
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
MatchQuality	VRESULT	1802	R - O	Double
MaximumAngleDeviation	VPARAMETER	5102	R / W	Double
MinimumArcPercentage	VPARAMETER	5142	R / W	Double
OutputArcAngle	VRESULT	1841	R - O	Double
OutputArcCenterPointX	VRESULT	1846	R - O	Double
OutputArcCenterPointY	VRESULT	1847	R - O	Double
OutputArcRadius	VRESULT	1840	R - O	Double
PolarityMode	VPARAMETER	5100	R / W	Long
PositioningLevel	VPARAMETER	561	R / W	Long

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ResultCount	VRESULT	1010	R - O	Long
SearchMode	VPARAMETER	5101	R / W	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
SubsamplingLevel	VPARAMETER	5110	R / W	Long
ToolGuidelineOffset	VPARAMETER	130	R / W	Double
ToolOpening	VPARAMETER	137	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRadius	VPARAMETER	135	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolThickness	VPARAMETER	136	R / W	Double

Blob Analyzer Properties

The Blob Analyzer finds, labels and analyzes irregular shaped objects. This tool detects and computes intrinsic/extrinsic geometric and greylevel properties of blobs that meet user-defined criteria.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
BlobArea	VRESULT	1611	R - O	Double
BlobBoundingBoxBottom	VRESULT	1648	R - O	Double
BlobBoundingBoxCenterX	VRESULT	1624	R - O	Double
BlobBoundingBoxCenterY	VRESULT	1625	R - O	Double
BlobBoundingBoxHeight	VRESULT	1626	R - O	Double
BlobBoundingBoxLeft	VRESULT	1645	R - O	Double
BlobBoundingBoxRight	VRESULT	1646	R - O	Double
BlobBoundingBoxRotation	VRESULT	1649	R - O	Double
BlobBoundingBoxTop	VRESULT	1647	R - O	Double
BlobBoundingBoxWidth	VRESULT	1627	R - O	Double
BlobChainCode	VRESULT	1656	R - O	Long
BlobChainCodeDeltaX	VRESULT	1659	R - O	Double
BlobChainCodeDeltaY	VRESULT	1660	R - O	Double
BlobChainCodeLength	VRESULT	1655	R - O	Long
BlobChainCodeStartX	VRESULT	1657	R - O	Double
BlobChainCodeStartY	VRESULT	1658	R - O	Double
BlobConvexPerimeter	VRESULT	1614	R - O	Double
BlobCount	VRESULT	1610	R - O	Long
BlobElongation	VRESULT	1616	R - O	Double
BlobExtentBottom	VRESULT	1653	R - O	Double
BlobExtentLeft	VRESULT	1650	R - O	Double
BlobExtentRight	VRESULT	1651	R - O	Double
BlobExtentTop	VRESULT	1652	R - O	Double
BlobGreyLevelMaximum	VRESULT	1622	R - O	Long
BlobGreyLevelMean	VRESULT	1618	R - O	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BlobGreyLevelMinimum	VRESULT	1621	R - O	Long
BlobGreyLevelRange	VRESULT	1619	R - O	Long
BlobGreyLevelStdDev	VRESULT	1620	R - O	Double
BlobHoleCount	VRESULT	1654	R - O	Long
BlobInertiaMaximum	VRESULT	1633	R - O	Double
BlobInertiaMinimum	VRESULT	1632	R - O	Double
BlobInertiaXAxis	VRESULT	1634	R - O	Double
BlobInertiaYAxis	VRESULT	1635	R - O	Double
BlobIntrinsicBoundingBoxBottom	VRESULT	1639	R - O	Double
BlobIntrinsicBoundingBoxCenterX	VRESULT	1628	R - O	Double
BlobIntrinsicBoundingBoxCenterY	VRESULT	1629	R - O	Double
BlobIntrinsicBoundingBoxHeight	VRESULT	1630	R - O	Double
BlobIntrinsicBoundingBoxLeft	VRESULT	1636	R - O	Double
BlobIntrinsicBoundingBoxRight	VRESULT	1637	R - O	Double
BlobIntrinsicBoundingBoxRotation	VRESULT	1640	R - O	Double
BlobIntrinsicBoundingBoxTop	VRESULT	1638	R - O	Double
BlobIntrinsicBoundingBoxWidth	VRESULT	1631	R - O	Double
BlobIntrinsicExtentBottom	VRESULT	1644	R - O	Double
BlobIntrinsicExtentLeft	VRESULT	1641	R - O	Double
BlobIntrinsicExtentRight	VRESULT	1642	R - O	Double
BlobIntrinsicExtentTop	VRESULT	1643	R - O	Double
BlobPositionX	VRESULT	1612	R - O	Double
BlobPositionY	VRESULT	1613	R - O	Double
BlobPrincipalAxesRotation	VRESULT	1617	R - O	Double
BlobRawPerimeter	VRESULT	1615	R - O	Double
BlobRoundness	VRESULT	1623	R - O	Double
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ChainCodeResultsEnabled	VPARAMETER	1607	R / W	Boolean
ElapsedTime	VRESULT	1001	R - O	Double
ExtrinsicInertiaResultsEnabled	VPARAMETER	1604	R / W	Boolean

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
GreyLevelResultsEnabled	VPARAMETER	1608	R / W	Boolean
HoleFillingEnabled	VPARAMETER	5002	R / W	Boolean
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
IntrinsicBoxResultsEnabled	VPARAMETER	1605	R / W	Boolean
MaximumBlobArea	VPARAMETER	5001	R / W	Double
MinimumBlobArea	VPARAMETER	5000	R / W	Double
PerimeterResultsEnabled	VPARAMETER	1602	R / W	Boolean
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
SegmentationDark	VPARAMETER	5005	R / W	Long
SegmentationDynamicDark	VPARAMETER	5009	R / W	Double
SegmentationDynamicInside	VPARAMETER	5010	R / W	Double
SegmentationDynamicLight	VPARAMETER	5008	R / W	Double
SegmentationDynamicOutside	VPARAMETER	5011	R / W	Double
SegmentationInside	VPARAMETER	5006	R / W	Long
SegmentationLight	VPARAMETER	5004	R / W	Long
SegmentationMode	VPARAMETER	5003	R / W	Long
SegmentationOutside	VPARAMETER	5007	R / W	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
SortBlobsBy	VPARAMETER	1601	R / W	Long
SortResultsEnabled	VPARAMETER	1600	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double
TopologicalResultsEnabled	VPARAMETER	1609	R / W	Boolean

Calculated Arc Properties

The Calculated Arc calculates the circle enclosing an arc based on a specific calculation mode. Possible modes are:

- Three points on the arc
- Center point and one point on the arc

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Calculated Frame Properties

The Calculated Frame is used to create a vision frame from other features. Frames allow you to place vision tools on objects that are not always in the same location or orientation. This tool calculates a frame based on a specific calculation mode. Possible modes are:

- Two lines (X axis line, Y axis line)
- Two points (Origin, +X point)
- A fixed frame of reference
- Single point (Origin point with no rotation)
- Relative to a frame
- One point and a line (An origin point following the angle of the line)

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Calculated Line Properties

The Calculated Line can be created from two points, or from a point and a line. In the latter case, the calculated line will be running through the point and perpendicular to the line. This tool calculates a line based on a specific calculation mode. Possible modes are:

- Two points
- Perpendicular line

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Calculated Point Properties

The Calculated Point can be calculated based on the intersection two lines, a line and a circle, or midway between two points. A Calculated Point tool is also used to place a fixed point in the field of view. A fixed point could be used if you want to make all your measurements from a known reference point.. This tool calculates a point based on a specific calculation mode. Possible modes are:

- Midpoint (midpoint between two points)
- Point and a line (closet point on the line to another point)
- Point and an arc (closest point on the arc to another point)
- Fixed point
- A line and an arc (Line/Arc intersection)
- Two lines (Line-line intersection)
- Two arcs (Arc-arc intersection)

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Calibration Grid Locator Properties

The Calibration Grid Locator is used to locate a collection of dots in the field of view. It is used by the grid calibration procedure.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ContrastThreshold	VPARAMETER	303	R / W	Long
ContrastThresholdMode	VPARAMETER	302	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
OutlineLevel	VPARAMETER	300	R / W	Long
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Caliper Properties

The Caliper finds and locates one or more edge pairs and measures distances between the two edges within each pair.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
Edge1Constraints	VPARAMETER	5221	R / W	Long
Edge1Magnitude	VRESULT	1940	R - O	Double
Edge1MagnitudeConstraint	VPARAMETER	5227	R / W	Long
Edge1MagnitudeScore	VRESULT	1942	R - O	Double
Edge1PolarityMode	VPARAMETER	5211	R / W	Long
Edge1PositionConstraint	VPARAMETER	5224	R / W	Double
Edge1PositionScore	VRESULT	1944	R - O	Double
Edge1PositionX	VRESULT	1946	R - O	Double
Edge1PositionY	VRESULT	1947	R - O	Double
Edge1Rotation	VRESULT	1950	R - O	Double
Edge1Score	VRESULT	1952	R - O	Double
Edge1ScoreThreshold	VPARAMETER	5241	R / W	Double
Edge2Constraints	VPARAMETER	5222	R / W	Long
Edge2Magnitude	VRESULT	1941	R - O	Double
Edge2MagnitudeConstraint	VPARAMETER	5228	R / W	Long
Edge2MagnitudeScore	VRESULT	1943	R - O	Double
Edge2PolarityMode	VPARAMETER	5212	R / W	Long
Edge2PositionConstraint	VPARAMETER	5225	R / W	Double
Edge2PositionScore	VRESULT	1945	R - O	Double
Edge2PositionX	VRESULT	1948	R - O	Double
Edge2PositionY	VRESULT	1949	R - O	Double
Edge2Rotation	VRESULT	1951	R - O	Double
Edge2Score	VRESULT	1953	R - O	Double
Edge2ScoreThreshold	VPARAMETER	5242	R / W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
EdgeFilterHalfWidth	VPARAMETER	5203	R / W	Long
EdgeMagnitudeThreshold	VPARAMETER	5201	R / W	Double
ElapsedTime	VRESULT	1001	R - O	Double
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
PairCount	VRESULT	1920	R - O	Long
PairPositionX	VRESULT	1921	R - O	Double
PairPositionY	VRESULT	1922	R - O	Double
PairRotation	VRESULT	1923	R - O	Double
PairScore	VRESULT	1924	R - O	Double
PairSize	VRESULT	1925	R - O	Double
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolSkew	VPARAMETER	113	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Color Matching Tool Properties

The Color Matching tool analyzes images according and outputs statistics on areas that match defined color filters.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ColorFilterCount	VPARAMETER	5700	R - O	Long
ColorFilterMatchPixelCount	VRESULT	2502	R - O	Long
ColorFilterMatchQuality	VRESULT	2501	R - O	Double
ElapsedTime	VRESULT	1001	R - O	Double
FilterHueTolerance	VPARAMETER	5716	R / W	Long
FilterHueValue	VPARAMETER	5713	R / W	Long
FilterLuminanceTolerance	VPARAMETER	5718	R / W	Long
FilterLuminanceValue	VPARAMETER	5715	R / W	Long
FilterSaturationTolerance	VPARAMETER	5717	R / W	Long
FilterSaturationValue	VPARAMETER	5714	R / W	Long
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Communication Tool Properties

The Communication tool manages and sends vision instances to a queue on the Adept Controller. The queue can be accessed using the "getinstance" program by a V+ program.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
CommunicationToolResults	VRESULT	2600	R - O	Integer
InstanceCount	VRESULT	1310	R - O	Long
Reset	VPARAMETER	5500	W - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Custom Vision Tool Properties

The Custom Vision Tool is used to fill in the program that is called when the tool is executed. From within a Custom Vision tool, other tools can be executed, and return a set of results which are used as the output of the tool.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long

Edge Locator Properties

The Edge Locator finds and locates an edge or a set of edges that meet user-defined criteria.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
Constraints	VPARAMETER	5220	R / W	Long
EdgeCount	VRESULT	1900	R - O	Long
EdgeMagnitude	VRESULT	1901	R - O	Double
EdgeMagnitudeScore	VRESULT	1902	R - O	Double
EdgePolarityMode	VPARAMETER	5210	R / W	Long
EdgePositionScore	VRESULT	1903	R - O	Double
EdgePositionX	VRESULT	1904	R - O	Double
EdgePositionY	VRESULT	1905	R - O	Double
EdgeRotation	VRESULT	1906	R - O	Double
EdgeScore	VRESULT	1907	R - O	Double
EdgeSortResultsEnabled	VPARAMETER	5243	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
FilterHalfWidth	VPARAMETER	5202	R / W	Long
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
MagnitudeConstraint	VPARAMETER	5226	R / W	Long
MagnitudeThreshold	VPARAMETER	5200	R / W	Double
PositionConstraint	VPARAMETER	5223	R / W	Double
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
ScoreThreshold	VPARAMETER	5240	R / W	Double
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolSkew	VPARAMETER	113	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Flexibowl Feeder Properties

The Flexibowl Feeder manages and controls the Flexinowl Feeder hardware.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
FlexibowlFeederVRunCommand	VPARAMETER	7000	R / W	Long
FlexibowlFeederBlowTime	VPARAMETER	7012	R / W	Long
FlexibowlFeederFlipCount	VPARAMETER	7005	R / W	Long
FlexibowlFeederFlipDelay	VPARAMETER	7006	R / W	Long
FlexibowlFeederForwardAcceleration	VPARAMETER	7001	R / W	Long
FlexibowlFeederForwardAngle	VPARAMETER	7003	R / W	Long
FlexibowlFeederForwardDeceleration	VPARAMETER	7002	R / W	Long
FlexibowlFeederForwardSpeed	VPARAMETER	7004	R / W	Long
FlexibowlFeederShakeAcceleration	VPARAMETER	7007	R / W	Long
FlexibowlFeederShakeAngle	VPARAMETER	7009	R / W	Long
FlexibowlFeederShakeCount	VPARAMETER	7011	R / W	Long
FlexibowlFeederShakeDeceleration	VPARAMETER	7008	R / W	Long
FlexibowlFeederShakeSpeed	VPARAMETER	7010	R / W	Long

Gripper Clearance Tool Properties

The Gripper Clearance Tool is used to check that a region around an instance is clear of any obstructions. This is done by applying a user-defined number of histogram checks to the region (typically one histogram per gripper finger) to determine whether there is another instance that would interfere with the gripper when picking up an instance.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
Found	VRESULT	1800	R - O	Boolean
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double

Image Histogram Tool Properties

The Image Histogram tool computes greylevel statistics within a user-defined region of interest.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
GreylevelRange	VRESULT	1508	R - O	Long
Histogram	VRESULT	1511	R - O	Long
HistogramPixelCount	VRESULT	1512	R - O	Long
ImageHeight	VRESULT	1021	R - O	Long
ImagePixelCount	VRESULT	1513	R - O	Long
ImageSubsampling	VPARAMETER	5324	R / W	Long
ImageWidth	VRESULT	1020	R - O	Long
MaximumGreylevelValue	VRESULT	1507	R - O	Long
Mean	VRESULT	1500	R - O	Double
Median	VRESULT	1501	R - O	Double
MinimumGreylevelValue	VRESULT	1506	R - O	Long
Mode	VRESULT	1504	R - O	Long
ModePixelCount	VRESULT	1505	R - O	Long
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	121	R / W	Boolean
SamplingStepCustom	VPARAMETER	124	R / W	Double
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
StandardDeviation	VRESULT	1503	R - O	Double
TailBlack	VPARAMETER	5323	R / W	Double
TailBlackGreylevelValue	VRESULT	1509	R - O	Long
TailWhite	VPARAMETER	5322	R / W	Double
TailWhiteGreylevelValue	VRESULT	1510	R - O	Long
ThresholdBlack	VPARAMETER	5320	R / W	Long
ThresholdWhite	VPARAMETER	5321	R / W	Long
ToolHeight	VPARAMETER	111	R / W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double
Variance	VRESULT	1502	R - O	Double

Image Processing Tool Properties

The Image Processing Tool processes grey-scale images by applying arithmetic, assignment, logical, filtering, morphological or histogram operators. Users can define custom filtering operators.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ArithmeticClippingMode	VPARAMETER	5360	R / W	Long
ArithmeticConstant	VPARAMETER	5361	R / W	Long
AssignmentConstant	VPARAMETER	5365	R / W	Long
ArithmeticScale	VPARAMETER	5362	R / W	Double
AssignmentHeight	VPARAMETER	5366	R - O	Long
AssignmentWidth	VPARAMETER	5367	R - O	Long
FilteringClippingMode	VPARAMETER	5370	R / W	Long
FilteringKernelSize	VPARAMETER	5371	R / W	Long
FilteringScale	VPARAMETER	5372	R / W	Double
HistogramThreshold	VPARAMETER	5385	R / W	Long
LastOperation	VRESULT	2200	R - O	Long
LastOutputType	VRESULT	2201	R - O	Long
LogicalConstant	VPARAMETER	5380	R / W	Long
MorphologicalNeighborhoodSize	VPARAMETER	5390	R / W	Long
Operation	VPARAMETER	5355	R / W	Long
OverrideType	VPARAMETER	5351	R / W	Long
OverrideTypeEnabled	VPARAMETER	5350	R / W	Boolean
TransformFlags	VPARAMETER	5395	R / W	Long

Image Sampling Tool Properties

The Image Sampling tool is used to extract an area of an image and output it as a separate Image.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double

Image Sharpness Tool Properties

The Image Sharpness Tool computes the sharpness of preponderant edges in a user-defined region of interest.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
AutomaticCandidateCountEnabled	VPARAMETER	5301	R / W	Long
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
CandidatePointsCount	VPARAMETER	5300	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
KernelSize	VPARAMETER	5304	R / W	Long
MeasurementPointsCount	VRESULT	2002	R - O	Long
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
SamplingStepCustomEnabled	VPARAMETER	121	R / W	Boolean
Sharpness	VRESULT	2000	R - O	Double
SharpnessPeak	VRESULT	2001	R - O	Double
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
StandardDeviationThreshold	VPARAMETER	5302	R / W	Double
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Inspection Tool Properties

The Inspection tool analyzes the results of other tools based on a configuration of inspection filters. Logical operators AND and OR are applied to a set of conditions that apply to the results of other tools in a vision sequence.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InspectionFilterMeasuredValue	VRESULT	2700	R - O	Double
InspectionFilterPassStatus	VRESULT	2702	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long

Line Finder Properties

The Line Finder finds and locates linear features on objects and returns the line angle and point coordinates.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
AverageContrast	VRESULT	1801	R - O	Double
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ConformityTolerance	VPARAMETER	556	R / W	Double
ContrastThreshold	VPARAMETER	303	R / W	Long
ContrastThresholdMode	VPARAMETER	302	R / W	Long
DefaultConformityTolerance	VPARAMETER	552	R - O	Double
ElapsedTime	VRESULT	1001	R - O	Double
FitQuality	VRESULT	1803	R - O	Double
Found	VRESULT	1800	R - O	Boolean
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
MatchQuality	VRESULT	1802	R - O	Double
MaximumAngleDeviation	VPARAMETER	5102	R / W	Double
MinimumLinePercentage	VPARAMETER	5130	R / W	Double
OutputLineAngle	VRESULT	1820	R - O	Double
OutputLineEndPointX	VRESULT	1823	R - O	Double
OutputLineEndPointY	VRESULT	1824	R - O	Double
OutputLineStartPointX	VRESULT	1821	R - O	Double
OutputLineStartPointY	VRESULT	1822	R - O	Double
OutputLineVectorPointX	VRESULT	1825	R - O	Double
OutputLineVectorPointY	VRESULT	1826	R - O	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
PolarityMode	VPARAMETER	5100	R / W	Long
PositioningLevel	VPARAMETER	561	R / W	Long
ResultCount	VRESULT	1010	R - O	Long
SearchMode	VPARAMETER	5101	R / W	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
SubsamplingLevel	VPARAMETER	5110	R / W	Long
ToolGuidelineOffset	VPARAMETER	130	R / W	Double
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Locator Tool Properties

The Locator finds and locates objects based on the geometry of their contours. Scale factor, orientation, and position are provided for each located instance. Models are built using the integrated Model Editor.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ConformityTolerance	VPARAMETER	556	R / W	Double
ConformityToleranceRangeEnabled	VPARAMETER	553	R / W	Boolean
ContrastPolarity	VPARAMETER	522	R / W	Long
ContrastThreshold	VPARAMETER	303	R / W	Long
ContrastThresholdMode	VPARAMETER	302	R / W	Long
DefaultConformityTolerance	VPARAMETER	552	R - O	Double
DetailLevel	VPARAMETER	301	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceClearQuality	VRESULT	1319	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceFitQuality	VRESULT	1317	R - O	Double
InstanceIntrinsicBoundingBox	VRESULT	1330	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
InstanceMatchQuality	VRESULT	1318	R - O	Double
InstanceModel	VRESULT	1312	R - O	Long
InstanceOrdering	VPARAMETER	530	R / W	Long
InstanceOrderingReferenceX	VPARAMETER	531	R / W	Double
InstanceOrderingReferenceY	VPARAMETER	532	R / W	Double
InstanceRobotLocation	VLOCATION	1371	R - O	Location

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
InstanceRotation	VRESULT	1314	R - O	Double
InstanceScaleFactor	VRESULT	1313	R - O	Double
InstanceSymmetry	VRESULT	1320	R - O	Long
InstanceTime	VRESULT	1322	R - O	Double
InstanceTranslationX	VRESULT	1315	R - O	Double
InstanceTranslationY	VRESULT	1316	R - O	Double
InstanceVisible	VRESULT	1321	R - O	Double
MaximumInstanceCount	VPARAMETER	519	R / W	Long
MaximumInstanceCountEnabled	VPARAMETER	518	R / W	Long
MaximumRotation	VPARAMETER	517	R / W	Double
MaximumScaleFactor	VPARAMETER	513	R / W	Double
MinimumClearPercentage	VPARAMETER	559	R / W	Double
MinimumClearPercentageEnabled	VPARAMETER	558	R / W	Boolean
MinimumModelPercentage	VPARAMETER	557	R / W	Double
MinimumRequiredFeatures	VPARAMETER	560	R / W	Double
MinimumRotation	VPARAMETER	516	R / W	Double
MinimumScaleFactor	VPARAMETER	512	R / W	Double
ModelDisambiguationEnabled	VPARAMETER	403	R / W	Boolean
NominalRotation	VPARAMETER	515	R / W	Double
NominalRotationEnabled	VPARAMETER	514	R / W	Boolean
NominalScaleFactor	VPARAMETER	511	R / W	Double
NominalScaleFactorEnabled	VPARAMETER	510	R / W	Boolean
OutlineLevel	VPARAMETER	300	R / W	Long
OutputSymmetricInstances	VPARAMETER	520	R / W	Long
ParametersBasedOn	VPARAMETER	304	R / W	Long
PositioningLevel	VPARAMETER	561	R / W	Long
RecognitionLevel	VPARAMETER	550	R / W	Long
ResultCount	VRESULT	1010	R-O	Long
SearchBasedOnOutlineLevelOnly	VPARAMETER	521	R / W	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
Timeout	VPARAMETER	501	R / W	Long
TimeoutEnabled	VPARAMETER	500	R / W	Boolean
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Overlap Tool Properties

The purpose of the overlap tool is to filter instance found in an input image so that the robot is not instructed to pick up the same part more than once.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
Reset	VPARAMETER	5500	W - O	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Pattern Locator Properties

The Pattern Locator finds and locates instances of a greyscale pattern

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
AutoCoarsenessSelectionEnabled	VPARAMETER	5421	R / W	Long
BilinearInterpolationEnabled	VPARAMETER	120	R / W	Boolean
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
ElapsedTime	VRESULT	1001	R - O	Double
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
MatchCount	VRESULT	2100	R - O	Long
MatchPositionX	VRESULT	2102	R - O	Double
MatchPositionY	VRESULT	2103	R - O	Double
MatchRotation	VRESULT	2104	R - O	Double
MatchStrength	VRESULT	2101	R - O	Double
MatchThreshold	VPARAMETER	5420	R / W	Double
MaximumInstanceCount	VPARAMETER	519	R / W	Long
MaximumInstanceCountEnabled	VPARAMETER	518	R / W	Long
PatternHeight	VPARAMETER	5403	R / W	Double
PatternPositionX	VPARAMETER	5400	R / W	Double
PatternPositionY	VPARAMETER	5401	R / W	Double
PatternRotation	VPARAMETER	5404	R / W	Double
PatternWidth	VPARAMETER	5402	R / W	Double
ResultCount	VRESULT	1010	R - O	Long
SamplingStepCustom	VPARAMETER	124	R / W	Double
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ToolHeight	VPARAMETER	111	R / W	Double
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Point Finder Properties

The Point Finder finds and locates point features on objects and returns the angle as well as the coordinates of the found point.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
AverageContrast	VRESULT	1801	R - O	Double
CalibratedUnitsEnabled	VPARAMETER	103	R - O	Boolean
Connectivity	VPARAMETER	5120	R / W	Long
ContrastThreshold	VPARAMETER	303	R / W	Long
ContrastThresholdMode	VPARAMETER	302	R / W	Long
ElapsedTime	VRESULT	1001	R - O	Double
Found	VRESULT	1800	R - O	Boolean
FrameCount	VRESULT	2410	R - O	Long
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceLocationGripperOffsetMaximum	VLOCATION	1499	R - O	Location
InstanceLocationGripperOffsetMinimum	VLOCATION	1400	R - O	Location
InterpolatePositionMode	VPARAMETER	5122	R / W	Long
InterpolatePositionModeEnabled	VPARAMETER	5123	R / W	Boolean
OutputPointX	VRESULT	1810	R - O	Double
OutputPointY	VRESULT	1811	R - O	Double
PolarityMode	VPARAMETER	5100	R / W	Long
PositioningLevel	VPARAMETER	561	R / W	Long
ResultCount	VRESULT	1010	R - O	Long
SearchMode	VPARAMETER	5101	R / W	Long
ShowResultsGraphics	VPARAMETER	150	R / W	Boolean
SubsamplingLevel	VPARAMETER	5110	R / W	Long
ToolGuidelineOffset	VPARAMETER	130	R / W	Double
ToolHeight	VPARAMETER	111	R / W	Double

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ToolPositionX	VPARAMETER	100	R / W	Double
ToolPositionY	VPARAMETER	101	R / W	Double
ToolRotation	VPARAMETER	112	R / W	Double
ToolWidth	VPARAMETER	110	R / W	Double

Recipe Manager Properties

The recipe manager controls the recipe subsystem.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
RecipeManagerActiveRecipe	VPARAMETER	8001	R - O	Long
RecipeManagerRecipeCount	VPARAMETER	8000	R - O	Long
RecipeManagerRecipeSelection	VPARAMETER	8002	R / W	Long
RecipeManagerRecipeDelete	VPARAMETER	8003	R - O	Long
RecipeManagerLoadFile	VPARAMETER	8004	R - O	Long
RecipeManagerSaveFile	VPARAMETER	8005	R - O	Long

Remote Vision Tool Properties

The Custom Vision Tool allows an ACE application, such as the ACE PackXpert, to run a vision tool on a remote PC.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
ElapsedTime	VRESULT	1001	R - O	Double
FrameRotation	VRESULT	2402	R - O	Double
FrameTranslationX	VRESULT	2400	R - O	Double
FrameTranslationY	VRESULT	2401	R - O	Double
InstanceCount	VRESULT	1310	R - O	Long
InstanceLocation	VLOCATION	1311	R - O	Location
InstanceRobotLocation	VLOCATION	1371	R - O	Location
ResultCount	VRESULT	1010	R - O	Long

Virtual Camera Tool Properties

The Virtual Camera tool provides input images to tools in a vision sequence. A Virtual Camera tool can acquire images from a camera, or from a database of images, through a virtual camera called an Emulation device.

Property Name	V+/MicroV+ Keyword	V+/MicroV+ ID	Access	Type
Abort	VPARAMETER	5501	W - O	Boolean
ActiveCalibration	VPARAMETER	5504	R / W	Long
ActiveSettings	VPARAMETER	5505	R / W	Long
VideoExposure	VPARAMETER	5502	R / W	Long
VideoGain	VPARAMETER	5503	R / W	Long

Customizer Reference Guide

There are many ways which ACE can be extended, modified, or changed. This section will detail many examples of customization. You should also reference the **AceDemo.zip** file included with the ACE installation. It contains several samples using Visual Studio for creating 3rd party applications that work with ACE.

See Also

PC Customization

Process Manager Customization Overview

PC Customization

This section details the customizations that can be done on the PC.

See Also

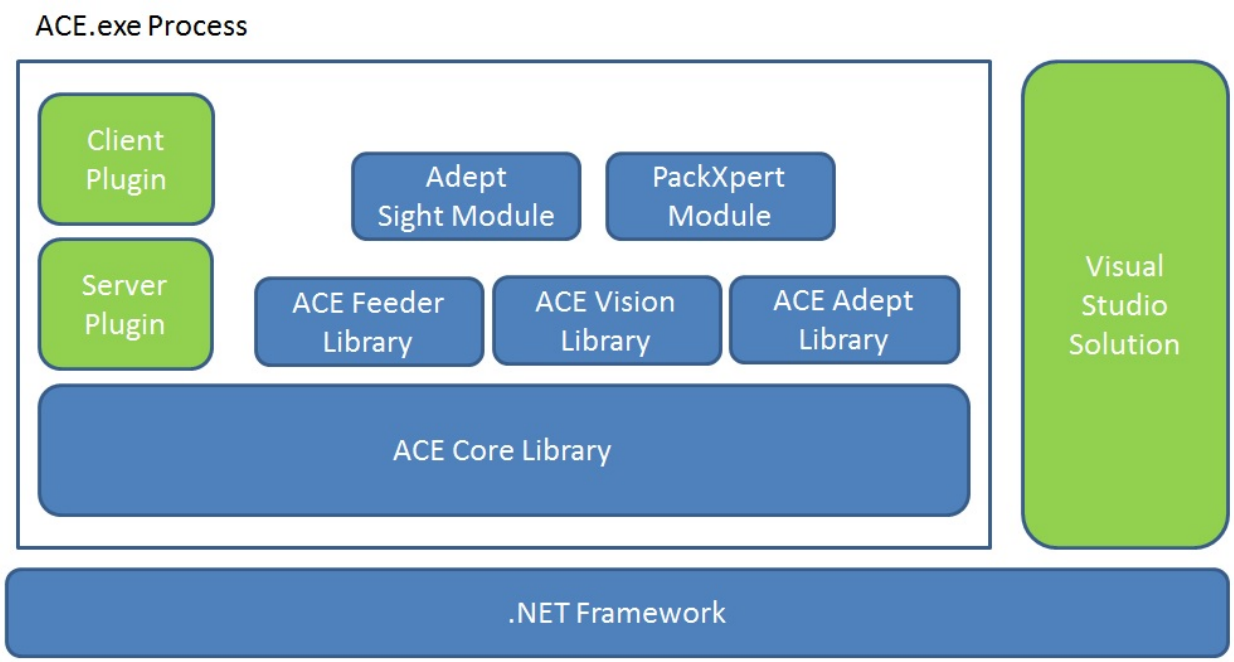
Overview

Data Collection Sample

Vision Tool Customization

Overview

ACE is a software package that can be used to develop a wide array of robotic applications. The software is divided into several components, each of which provides some of the functionality available to the user. Though ACE is generally sufficient for most applications, it was designed so it can be extended. Below is a simplified diagram showing the various functional components of ACE:



ACE Core Library

The ACE core library contains all the objects and classes that define the ACE runtime framework. This runtime framework is build upon the .NET framework.

ACE Vision Library

The ACE Vision library encapsulates the vision system functions that can be used to trigger vision operations on images supplied from cameras connected to the PC.

ACE Adept Library

The ACE Adept library contains all the communications and class structures needed to interact with all robots and controllers supplied by Adept.

ACE Sight Module

ACE Sight is a higher level functional module that exposes the ACE Vision library to a user through the V+ ACE Sight keywords.

ACE PackXpert Module

ACE PackXpert is a higher level functional module that provides a solution for packaging applications.

Client and Server Side Plugins

Plugins are 3rd party libraries developed with Visual Studio that are loaded directly within the ACE.exe process. Client Plugins are assciated with the ACE user interface. Server Plugins are associated with the ACE runtime. Examples of Client and Server plugins can be located in the ***AceDemo.zip*** file that ships with ACE.

Visual Studio Solution

A user can develop external applications that interact with ACE using a tool like Visual Studio. In this case, Visual Studio is used to create a separate executable that communicates with the ACE.exe process using the ACE API. Examples of this can also be found in the ***AceDemo.zip*** file that ships with ACE.

Application Domains

When developing a Plugin, Visual Studio solution that interacts with ACE or scriptable objects within ACE, it is important to understand Application Domains.

Application domains are sandboxes which allow for the isolation of software components. The ACE software is a multi-application domain application and under this model, the AceServer is run in its own isolated application domain. It uses the .NET framework remoting infrastructure to expose itself to software components outside of its application domain. Depending on how you are accessing the AceServer, there are implications for users seeking to customize ACE. This table details the different ways the ACE product can be customized and identifies if it is running in the AceServer application domain:

Customization Type	Running in AceServer Application Domain
C# Program Object	No
Custom Allocation Script	Yes
C# Custom Vision Tooloffset	No
Client Side Plug-In	Yes or no, depending how ACE was launched.
Server Side Plug-In	Yes

Crossing the Application Domain Boundaries

If you are accessing a property or a method that exists in a different application domain, it should be understood that the access involves transferring information from the current application domain to the remote application domain. The process of transferring information is known as *marshalling* data. The way data is marshaled depends on the kind data being passed. If the object is considered a value type, then a copy of the data is made and passed to the remote domain. Any changes made to the value type on the server are not reflected in the original application domain where the object was originally created.

If the object is considered a reference type, then the object itself will be passed across the domain boundary. Any changes made in the remote application domain are reflected in the original object. All reference types in the ACE API will derive from ***Ace.Core.RemoteableObject*** which derives from the ***System.MarshalByRef*** base class.

Remoting, Object Scope, and Serialization

When an object is created, it is always created within the scope of the application domain in which it was created. If, for example, you are creating an object that derives from ***RemotableObject***, it will be owned by the current application domain. If your intention is to create an object and used it in the AceServer application domain, you must use the ***AceServer::CreateObject*** method to create the object. This method will force the object to be created in the AceServer application domain.

This is important because there are limitations imposed by the .NET framework when saving information that crosses an application domain boundary. Using the ***AceServer::CreateObject*** method will ensure all objects existing in the application domain of the AceServer and that there are no access or serialization issues if the data is saved as part of the workspace.

Here is an example showing a C# script creating an object that is saved as part of the workspace incorrectly:

```
public void Main () {

    IProcessManager processManager = (IProcessManager) ace["/Process Manager"];
    IPartType part = (IPartType) ace["/Part"];

    ExclusionConfiguration exclusion0 = new ExclusionConfiguration();
    exclusion0.ExcludedIndexes = new int[] {0,1};

    IPartProcess proc = processManager.Processes[0];
    proc = processManager.Processes[0];

    proc.PickConfiguration.UpdateMultiConfiguration(0, part, exclusion0, 0);


}
```

The correct way to create remotable objects that cross the application domain boundary would be:

```
public void Main () {

    IProcessManager processManager = (IProcessManager) ace["/Process Manager"];
    IPartType part = (IPartType) ace["/Part"];

    ExclusionConfiguration exclusion0 = ace.CreateObject(typeof(ExclusionConfiguration));
    exclusion0.ExcludedIndexes = new int[] {0,1};

    IPartProcess proc = processManager.Processes[0];
    proc = processManager.Processes[0];

    proc.PickConfiguration.UpdateMultiConfiguration(0, part, exclusion0, 0);


}
```

This special handling of object creation is not required for value types, objects that will not cross the application domain boundary. or objects that are not saved as part of the workspace.

Server Side Plug-Ins

As noted in the table above, server side plug-ins always run in the application domain of the AceServer. As such, all objects created by the plug-in already belong to the AceServer application domain. However, the ***AceServer::CreateObject*** also handles other initializations, assignments, and type mapping that may be needed when an object is created. Because of this, even server-side plug-ins should use the ***AceServer::CreateObject*** method when dealing with ***RemotableObject*** types.

Data Collection Sample

This sample shows how to use data collection from within a C# program.

Default C# Program

When a C# Program is created, the following is created:

The script is comprised of several major elements:

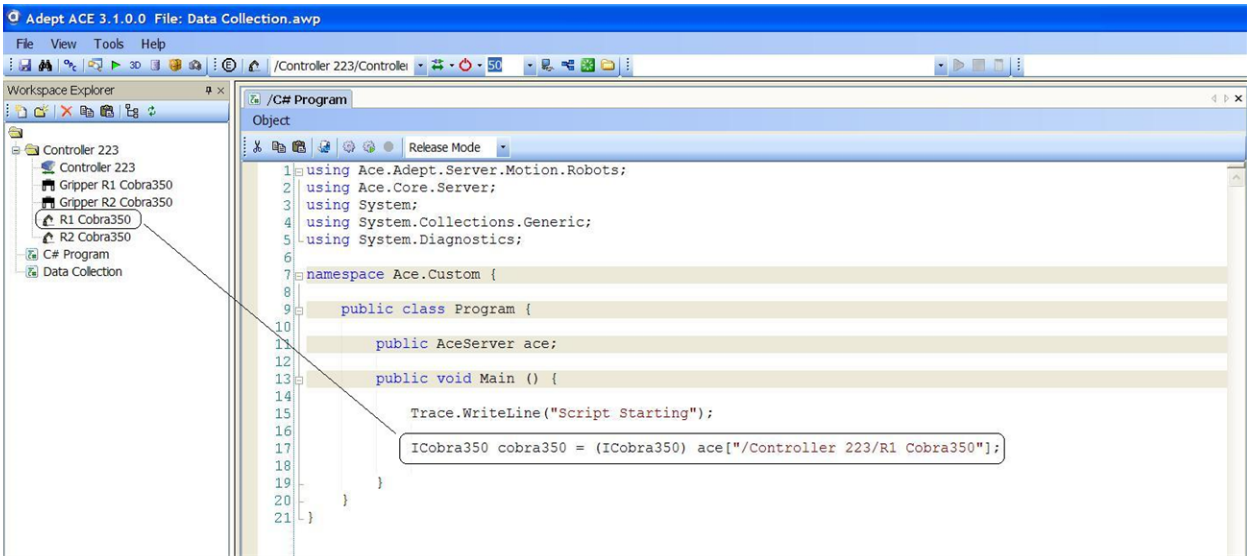
- C# class defenition
- IAceServer field
- Main method entry point

The script contains many lines that cannot be modified and are considered protected. These are designated by the light grey background. Notice that the namespace, class name, IAceServer field name, and ***Main*** method lines are protected and cannot be changed.

When the C# Program is executed, the text in the script is dynamically compiled in memory and the ***Main*** method is called. It is up to the user to define the logic of the script.

Referencing other objects and tools

It is often useful to reference other objects in the wqorkspace when running a script. You can reference other objects by dragging and dropping vision tools from the Workspace Explorer to the body of the script as such:



Data Collection Overview

In this sample, the script will collect the encoder position for all 4 motors of a Cobra robot while the robot is moving. Do to this, the following steps will be required:

- Acquire a handle to a robot
- Identify the data to be collected
- Collect the data
- Extract the data
- Save the data

Step 1: Acquire a handle to a robot

We first drag and drop a robot from the workspace into the script. Since the script will be moving the robot, the script will ensure that robot power is enabled and the robot is calibrated.

```
// Get a handle to a robot in the workspace
ICobra350 cobra350 = (ICobra350) ace["/Controller 223/R1 Cobra350"];

// Ensure the robot power is on and calibrated
if (cobra350.Power == false) {
    cobra350.Power = true;
    cobra350.Calibrate();
}
```

Step 2: Identify the data to be collected

The script must create a ***DataCollectionConfiguration*** collection and add ***DataCollectionItem*** objects associated with each data item that should be collected. In our case, we must create a ***DataCollectionItem*** for each motor of the robot and associate each one with the ServoData we wish to collect.

The ***DataCollectionConfiguration*** also must be told how many samples are to be collected each second and how many seconds to run the test.

Lastly, the ***DataCollectionConfiguration*** needs to be associated with the robot.

```
// Create a data collection configuration and collect 500 samples/second
// for 2 seconds.
DataCollectionConfiguration dataConfig = new DataCollectionConfiguration();
dataConfig.SamplesPerSecond = 500;
dataConfig.TimeToCollect = 2;

// Add a data collection item for each motor monitoring the position
for(int i=0; i<cobra350.Motors.Count; i++){
    DataCollectionItem item = new DataCollectionItem(cobra350.Motors[i], ServoData.EncoderPosition);
    dataConfig.Add(item);
}

// Associate the data collection configuration with the robot
cobra350.DataCollectionConfiguration = dataConfig;
```

Step 3: Collect the data

Before collection is started, a ***CartesianMove*** object is created so we can command the robot to move to a cartesian position while the data collection is being performed. The Move command is a non-blocking command, so once issued, the script continues to run.

After the robot is ordered to move to the cartesian position, the data collection is invoked. While the data is being collected and the robot is moving, the script will delay. Once the delay is complete, the script waits until the motion has completed, detaches the robot, and disposes of the move command.

```
// Start the robot moving
CartesianMove move = ace.CreateObject(typeof(CartesianMove)) as CartesianMove;
move.Param.Speed = 1;
move.Param.Accel = 100;
move.Param.Decel = 100;
move.WorldLocation = cobra350.WorldLocationWithTool.Shift(0,0,25);
cobra350.Move(move);

// Collect the data
cobra350.DataCollectionEnabled = true;
Thread.Sleep(4000);
cobra350.DataCollectionEnabled = false;

// Make sure the move has completed
cobra350.WaitMoveDone();

// Force a DETACH on the robot
cobra350.AutomaticControlActive = false;

// Release access to the motion
move.Dispose();
move = null;
```

Step 4: Extract the data

After data collection has competed, the data resides on the Adept Controller and must be extracted. This extraction is done using the ***AdeptRobot::ReadDataCollection*** method as noted below.

The script is accessing the position of each motor. Once the motor positions are extracted, the motor positions are combined to calculate the world location of the robot at each sample point.

```
// Read the data from the collection
List<double[]> dataValues = new List<double[]>();
for(int i=0; i<cobra350.Motors.Count; i++){
    DataCollectionItem item = dataConfig[i];
    double[] data = cobra350.ReadDataCollection(item, false, 0, 0);
    dataValues.Add(data);
}

// Go through all data items and convert to a transform
List<Transform3D> locations = new List<Transform3D>();
for(int i=0; i<dataValues[0].Length; i++){
    List<double> motorPositions = new List<double>();
    foreach (double[] dataValue in dataValues){
        motorPositions.Add(dataValue[i]);
    }

    double[] jointPositions = cobra350.MotorToJoint(motorPositions.ToArray());
    Transform3D worldLocation = cobra350.ForwardKinematics(jointPositions);
    locations.Add(worldLocation);
}
```

Step 5: Save the data to a file

Lastly, the transformation data is written to a file

```
// Delete the data file if it already exists
if (File.Exists(@"C:\data.txt"))
    File.Delete(@"C:\data.txt");

// Go through all transforms and append to a file
foreach (Transform3D location in locations) {
    File.AppendAllText(@"C:\data.txt", location.ToString() + Environment.NewLine);
}
```

Output Data File

Here is a sample of the data that was collected with the script listed above. The example moves the Z axis of the robot up. This can be seen in the data snippet:

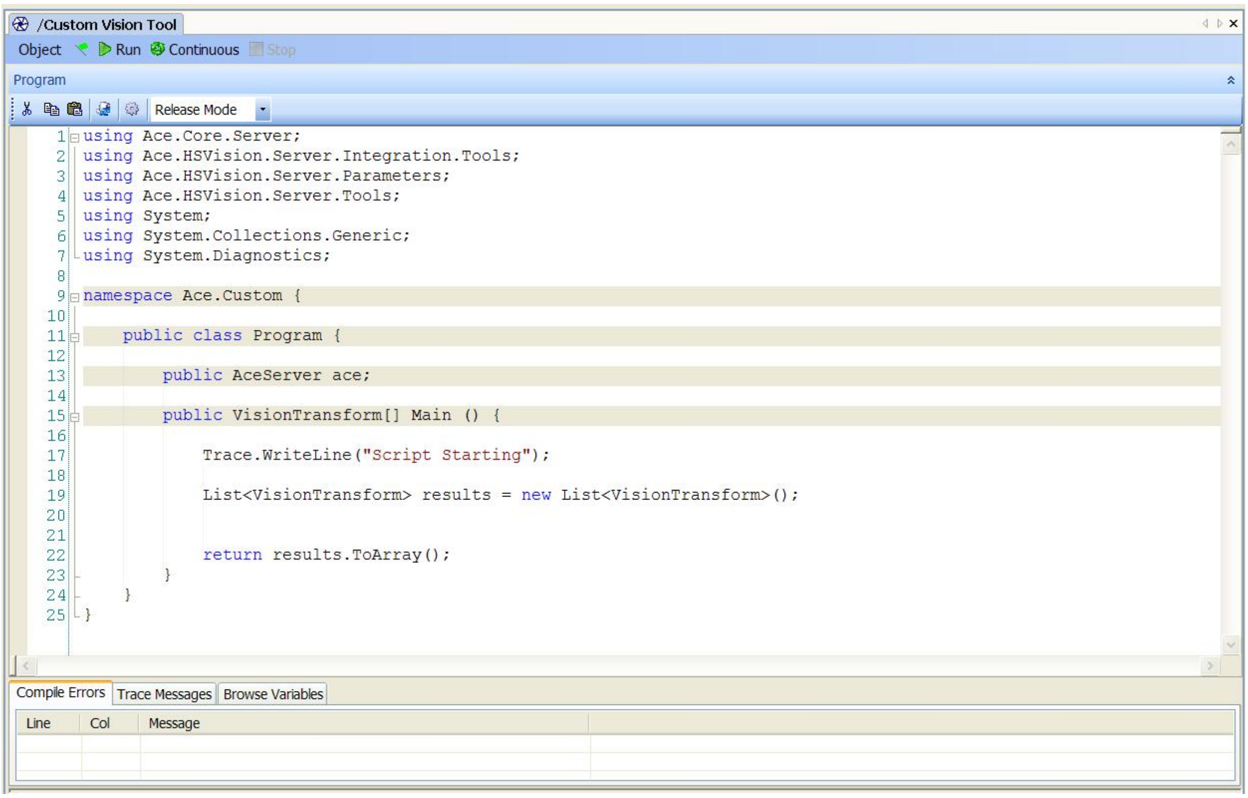
```
268.227 -102.832 66.469 0.000 180.000 -116.872
268.227 -102.832 66.469 0.000 180.000 -116.860
268.227 -102.832 66.470 0.000 180.000 -116.864
268.227 -102.832 66.470 0.000 180.000 -116.874
268.227 -102.832 66.470 0.000 180.000 -116.868
268.227 -102.832 66.470 0.000 180.000 -116.859
268.227 -102.832 66.471 0.000 180.000 -116.867
268.227 -102.832 66.471 0.000 180.000 -116.874
268.227 -102.833 66.472 0.000 180.000 -116.864
268.227 -102.832 66.472 0.000 180.000 -116.860
268.227 -102.832 66.474 0.000 180.000 -116.871
268.227 -102.832 66.475 0.000 180.000 -116.872
268.227 -102.832 66.476 0.000 180.000 -116.861
268.227 -102.832 66.478 0.000 180.000 -116.862
268.227 -102.832 66.481 0.000 180.000 -116.873
268.227 -102.832 66.483 0.000 180.000 -116.869
268.227 -102.832 66.485 0.000 180.000 -116.859
268.227 -102.832 66.489 0.000 180.000 -116.865
268.227 -102.832 66.492 0.000 180.000 -116.874
268.227 -102.832 66.494 0.000 180.000 -116.866
268.227 -102.832 66.497 0.000 180.000 -116.859
268.227 -102.832 66.501 0.000 180.000 -116.869
268.227 -102.832 66.505 0.000 180.000 -116.874
268.227 -102.832 66.508 0.000 180.000 -116.863
268.227 -102.832 66.511 0.000 180.000 -116.860
268.227 -102.832 66.515 0.000 180.000 -116.872
268.227 -102.832 66.519 0.000 180.000 -116.872
268.227 -102.832 66.522 0.000 180.000 -116.860
268.227 -102.832 66.526 0.000 180.000 -116.864
268.227 -102.832 66.530 0.000 180.000 -116.874
268.227 -102.832 66.533 0.000 180.000 -116.868
268.227 -102.832 66.536 0.000 180.000 -116.859
268.227 -102.832 66.541 0.000 180.000 -116.867
268.227 -102.832 66.545 0.000 180.000 -116.875
268.227 -102.832 66.549 0.000 180.000 -116.865
268.227 -102.832 66.552 0.000 180.000 -116.860
268.227 -102.832 66.557 0.000 180.000 -116.871
268.227 -102.832 66.562 0.000 180.000 -116.873
268.227 -102.832 66.565 0.000 180.000 -116.861
268.227 -102.832 66.569 0.000 180.000 -116.862
268.227 -102.832 66.574 0.000 180.000 -116.874
268.227 -102.832 66.579 0.000 180.000 -116.870
```

Vision Tool Customization

This sample shows how to create a custom vision tool.

Default Custom Vision Tool

When a custom vision tool is created, the following is created:



The script is comprised of several major elements:

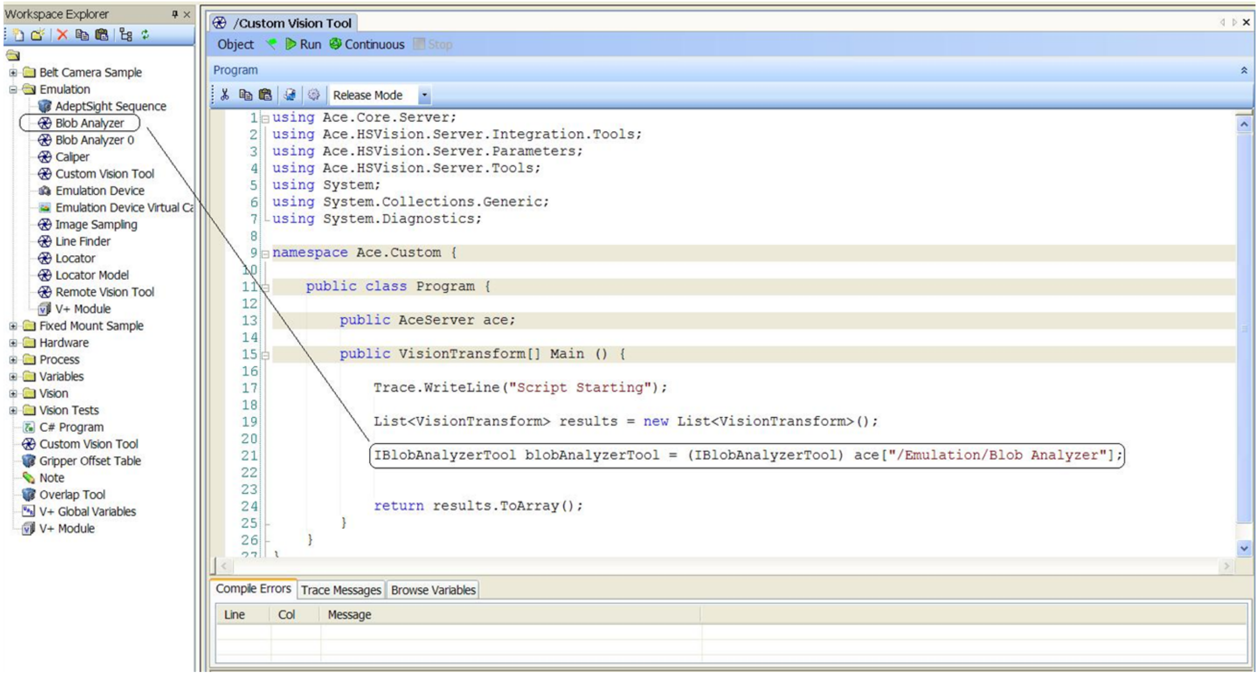
- C# class defenition
- IAceServer field
- Main method entry point
- return values from the script

The script contains many lines that cannot be modified and are considered protected. These are designated by the light grey background. Notice that the namespace, class name, IAceServer field name, and ***Main*** method lines are protected and cannot be changed.

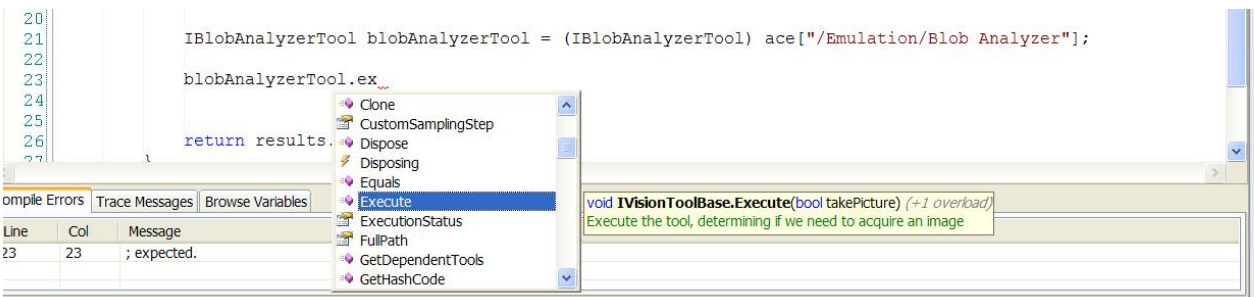
When the custom vision tool is invoked, the text in the script is dynamically compiled in memory and the ***Main*** method is called. By default, the script must return an array of ***VisionTransform*** objects. These define the coordinates of any objects located by the tool. The default custom vision tool simply returns an empty array of transformations. It is up to the user to define the logic of the script and to return an array of transformations.

Referencing other objects and tools

It is often useful to reference other vision tools or objects when running a script. You can reference other objects by dragging and dropping vision tools from the Workspace Explorer to the body of the script as such:



Any object can be dragged and dropped from the workspace explorer to the body of a script. When an object is dropped, the script automatically creates a reference to the object through the IAceServer field defined in the scrip. Once an object has been inserted into a script, you can access the properties and methods of that object as such:



Here is a sample of a custom vision tool that executes a blob tool and returns the results of the blob tool:

```
        public VisionTransform[] Main () {

            Trace.WriteLine("Script Starting");

            List<VisionTransform> results = new List<VisionTransform>();

            IBlobAnalyzerTool blobAnalyzerTool = (IBlobAnalyzerTool) ace["/Emulation/Blob Analyzer"];

            // Execute the blob tool without taking a new picture
            blobAnalyzerTool.Execute(false);

            // Add all found blobs but rotate the coordinates 45 degrees
            foreach (BlobResult res in blobAnalyzerTool.Results){
                VisionTransform newPosition = res.Position * new VisionTransform(0,0,45);
                results.Add(newPosition);
            }

            return results.ToArray();
        }
```

Complete Example

Here is an example showing a custom vision tool that will use a blob tool and a line finder to locate a square part. The vision tool will initially execute a blob tool to coarsely locate all parts in the vision window.

/Custom Vision Tool

Object Run Continuous Stop

Program

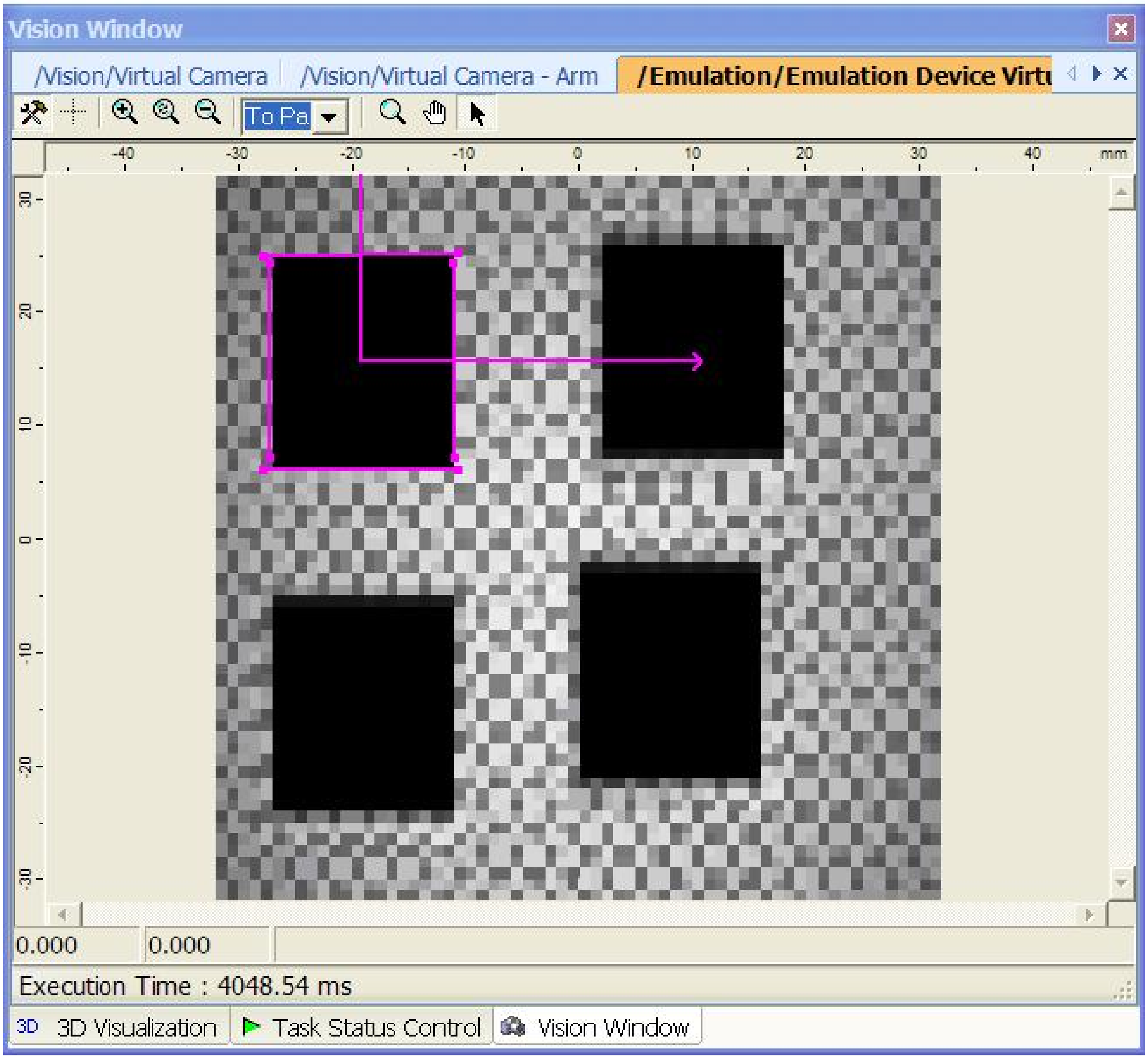
Debug Mode

```
using Ace.Core.Server;
using Ace.HSVision.Server.Integration.Tools;
using Ace.HSVision.Server.Parameters;
using Ace.HSVision.Server.Tools;
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace Ace.Custom {

    public class Program {

        public AceServer ace;

        public VisionTransform[] Main () {

            List<VisionTransform> results = new List<VisionTransform>();

            // Get the blob and line finder tool
            IBlobAnalyzerTool blobAnalyzerTool = (IBlobAnalyzerTool) ace["/Emulation/Blob Analyzer"];
            ILineFinderTool lineFinderTool = (ILineFinderTool) ace["/Emulation/Line Finder"];

            // Disable results graphics for the blob tool
            blobAnalyzerTool.ShowResultsGraphics = false;

            // Execute the blob tool without taking a new picture
            blobAnalyzerTool.Execute(false);
```

Once the blob tool executes, the script will iterate through all the results and will position a line finder tool and execute the tool at 0, 90, 180, and 270 degrees offset from the blob center. It will use the line returned from the line finder to calculate the 4 corners and, ultimately, the center of the square. The center point will be returned as the result for the tool.

```
            // Go through all blob results
            foreach (BlobResult res in blobAnalyzerTool.Results){

                VisionLine[] lines = new VisionLine[4];

                // At each blob result, execute 4 line finders around the part
                // and save the line
                for(int i=0; i<4; i++) {
                    lineFinderTool.Offset = new VisionTransform(res.Position.X, res.Position.Y, 90*i)*new VisionTransform(10,0,0);
                    lineFinderTool.Execute(false);
                    System.Threading.Thread.Sleep(1000);
                    lines[i] = lineFinderTool.Results[0].Line;
                }

                // Calculate the corners of the square
                VisionPoint corner1 = lines[0].GetLineIntersection(lines[1]);
                VisionPoint corner2 = lines[1].GetLineIntersection(lines[2]);
                VisionPoint corner3 = lines[2].GetLineIntersection(lines[3]);
                VisionPoint corner4 = lines[3].GetLineIntersection(lines[0]);

                //  Calculate the lines running from the corners to the center of the square
                VisionLine cornerLine1 = new VisionLine(corner1, corner3);
                VisionLine cornerLine2 = new VisionLine(corner2, corner4);

                // Calculate the middle point and add it as a result
                VisionPoint center = cornerLine1.GetLineIntersection(cornerLine2);
                results.Add(new VisionTransform(center));
            }

            return results.ToArray();
        }
    }
}
```

Here is the output of the tool in the vision window:



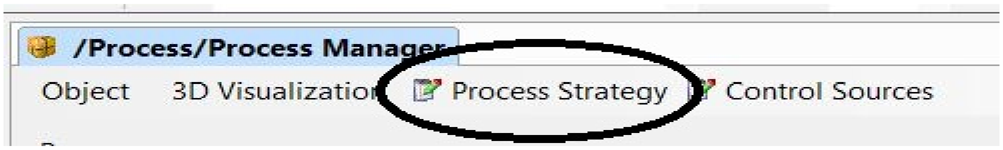
Process Manager Customization Overview

The process manager architecture was designed with extensibility and customization in mind. Many aspects of the product can be changed to meet custom application needs. When you customize some aspect of a process manager application, the system will create a copy of the default behavior as a starting point for your changes.

Customizations can be defined at various places in the process manager user interface. This section details provides an overview of the possible customizations, identifies where these are located in the user interface, and provides links to the relevant topics in the customizer guide.

Process Strategy Customization

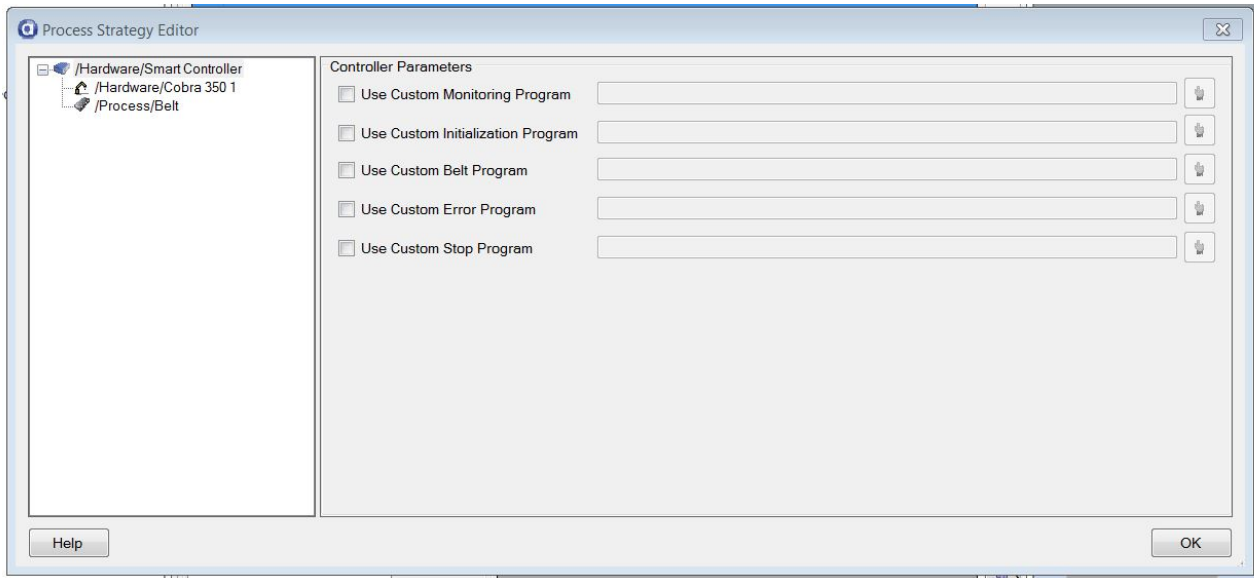
The process strategy section contains parameters related to the way processes are managed. Settings related to the process strategy can be access in the process manager editor here:



The form containing the process strategy parameters presents them into functional categories based on the hardware components of the application.

Controller Customizations

The general controller parameters are displayed when the controller node is selected in the process strategy editor:

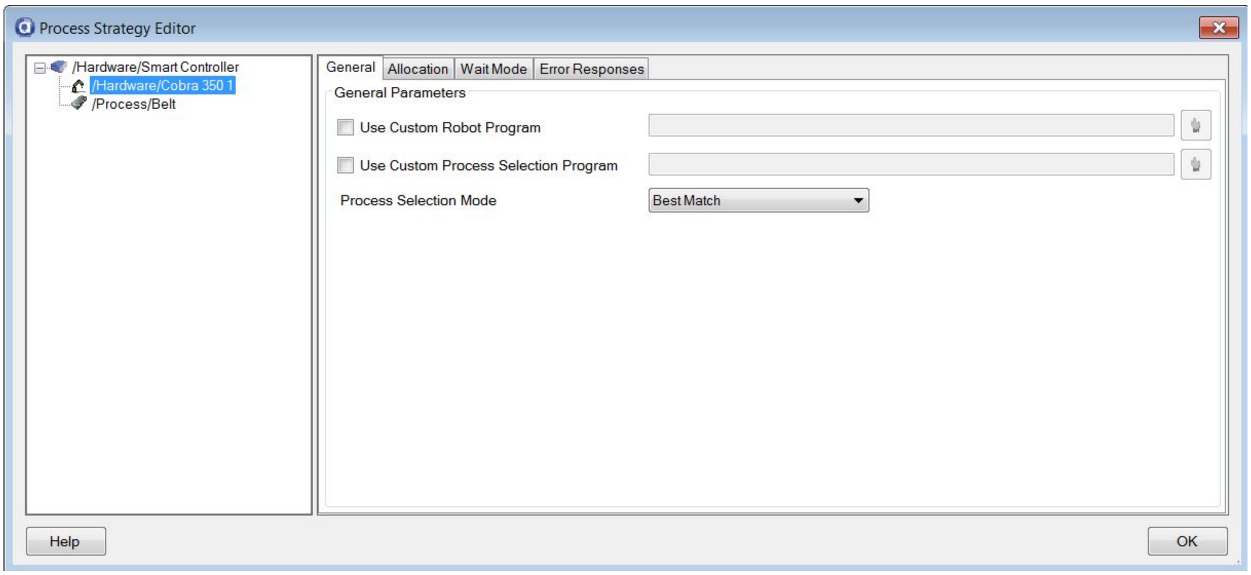


Custom Error Programs	V+ program called when any robot error occurs.
Customizing Stop Behavior	V+ program that runs when process manager stops running.

Robot Customizations

The robot parameters are displayed when a robot node is selected in the process strategy editor. There are many parameters associated with a robot. These settings are further broken down into a variety of categories represented by tabs in the display.

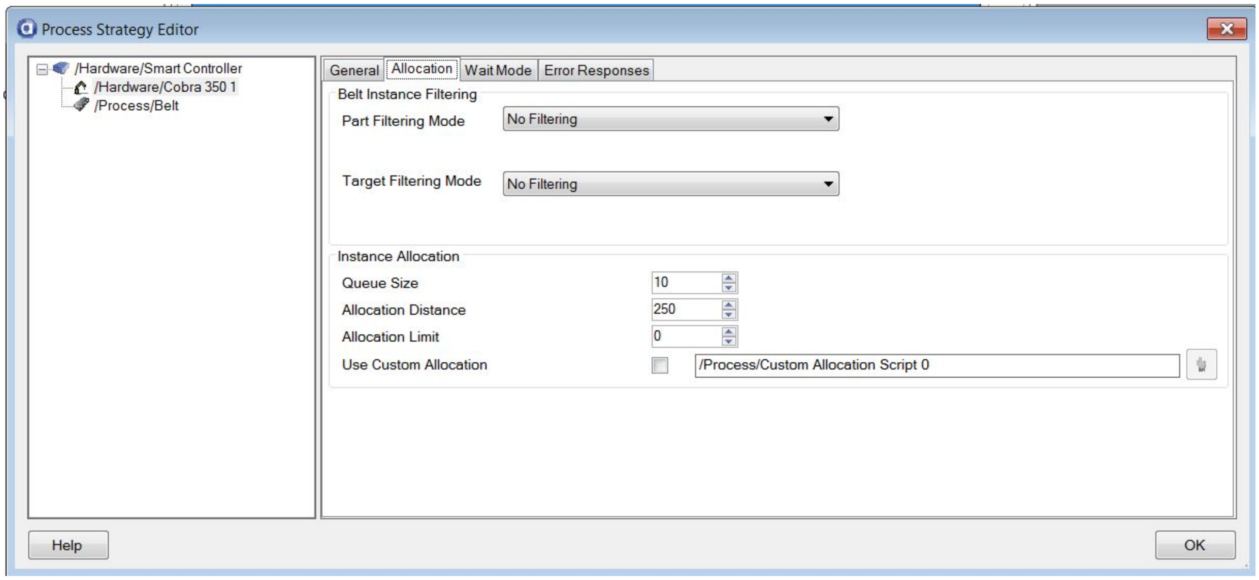
Robot: General



Custom Robot Program: Overview	Main V+ robot program that initiates all robot logic and behaviors.
Custom Robot Process Selection Program	V+ program called to select a process for the robot to execute.

Robot: Allocation

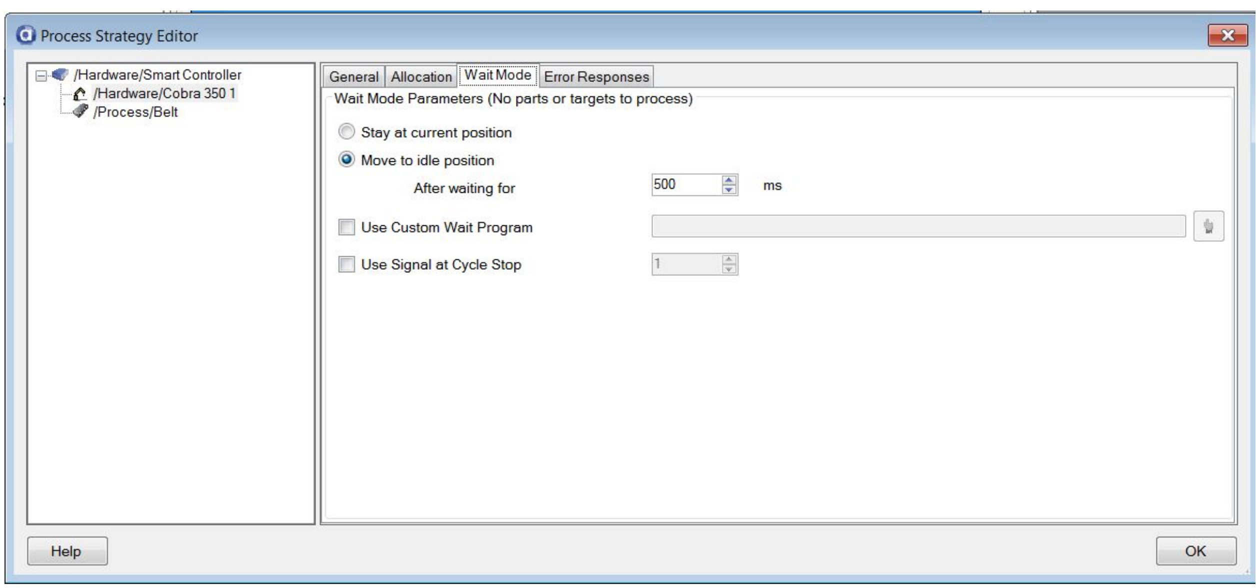
The parameters in the allocation section control how part and target instances are allocated to the associated robot:



Custom Allocation Scripts	C# Script running on the PC which decides which instances to allocate to a robot.

Robot: Wait Mode

The parameters in the wait mode section control how the robot behaves when an instance or process is not available for processing:

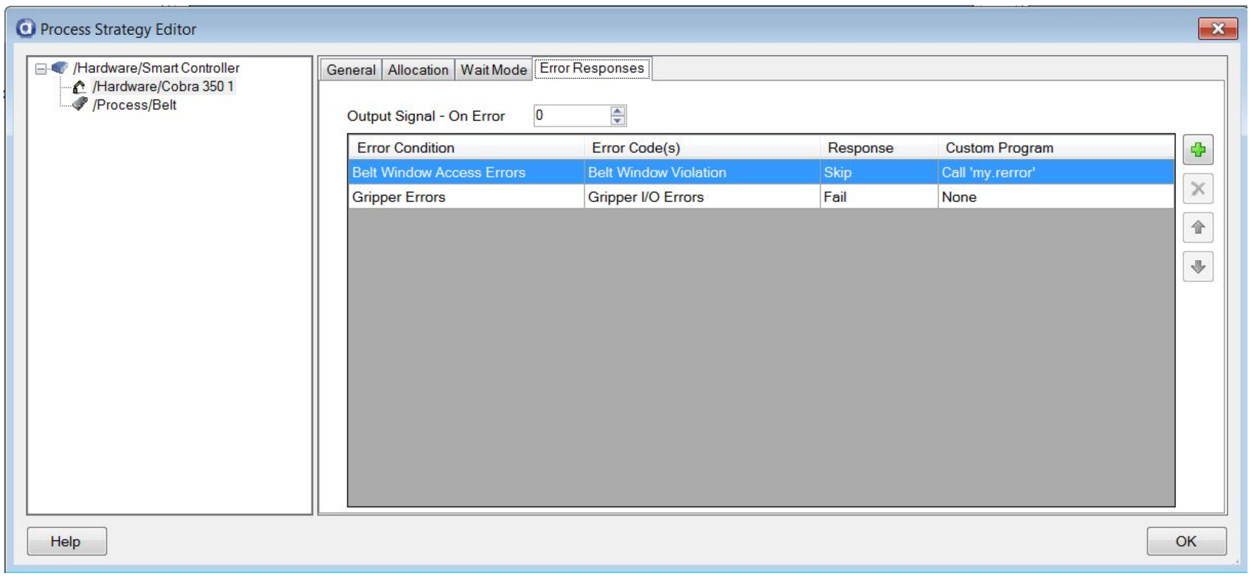


Custom Robot Wait	V+ program deciding what a robot will do if no instances are

Programs	available for processing.

Robot: Error Responses

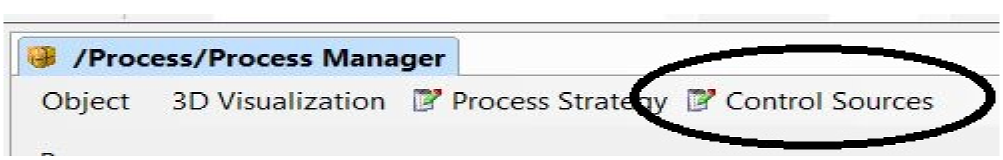
The error responses section detail how individual error conditions should be handled by the selected robot:



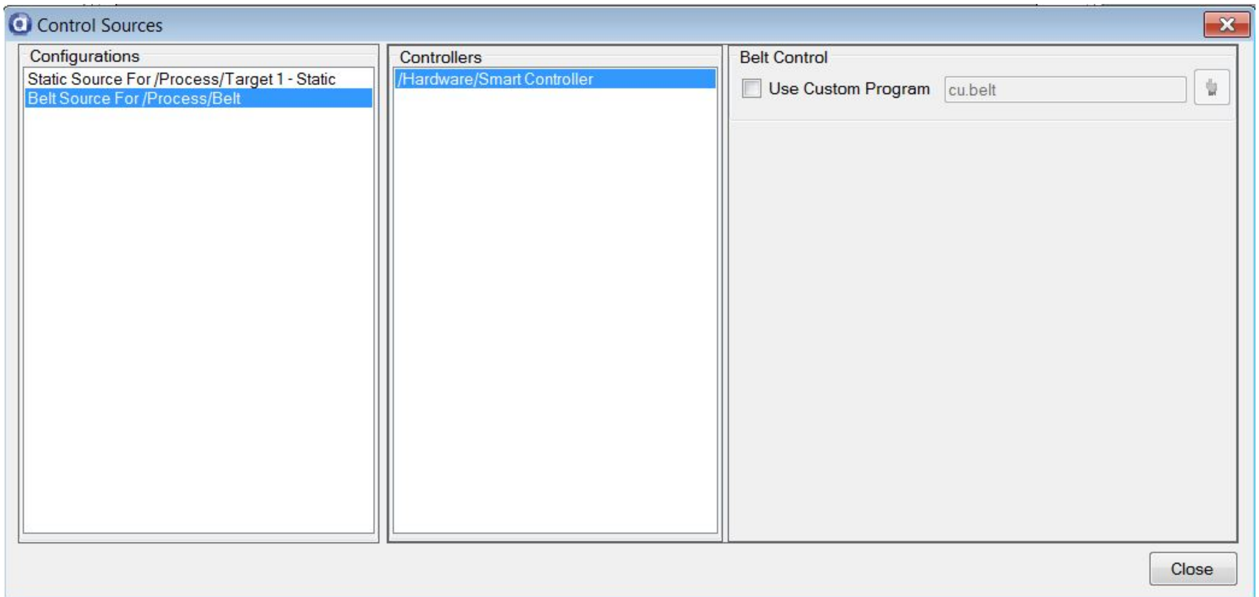
Custom Robot Error Response Programs	V+ program that is called when a specific error condition is detected for a given robot.

Control Sources: Belt Monitoring

When a belt is used to present part or target instances to a robot, the software creates a default V+ program to manage the belt operation. The settings associated with the belt monitoring can be viewed under the ***Control Sources*** section in the process manager editor:



When selected, the available control sources are presented in a list on the left. When you select the belt source for a given belt object in your process configuration, you will see a page similar to this:

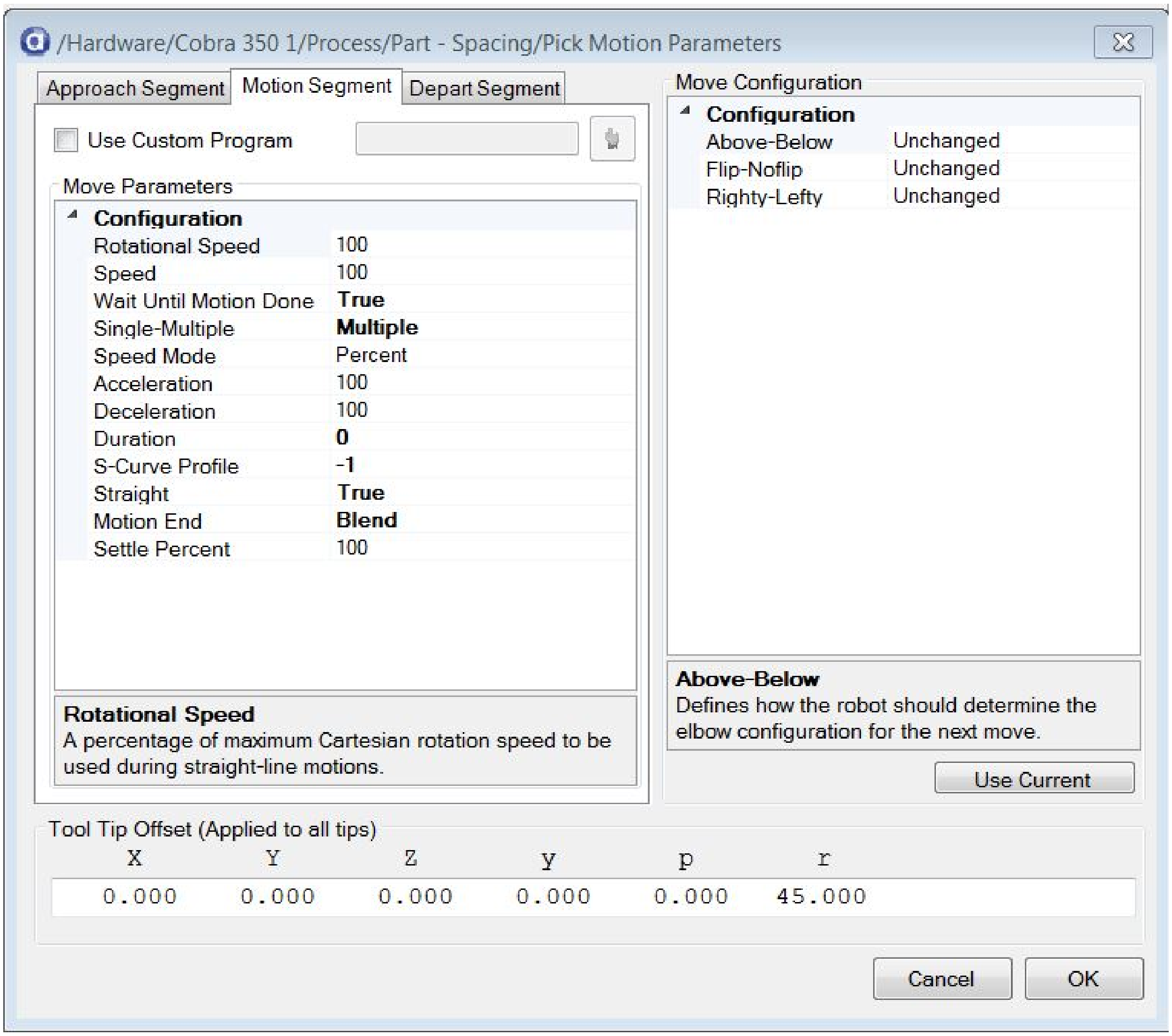


Depending on your specific configurations, different settings will be available.

Belt Monitoring	V+ program that monitors the status of a belt, and any associated latches, cameras, and spacing configurations. Reports the status of these items to the PC.

Motion Settings: Pick and Place

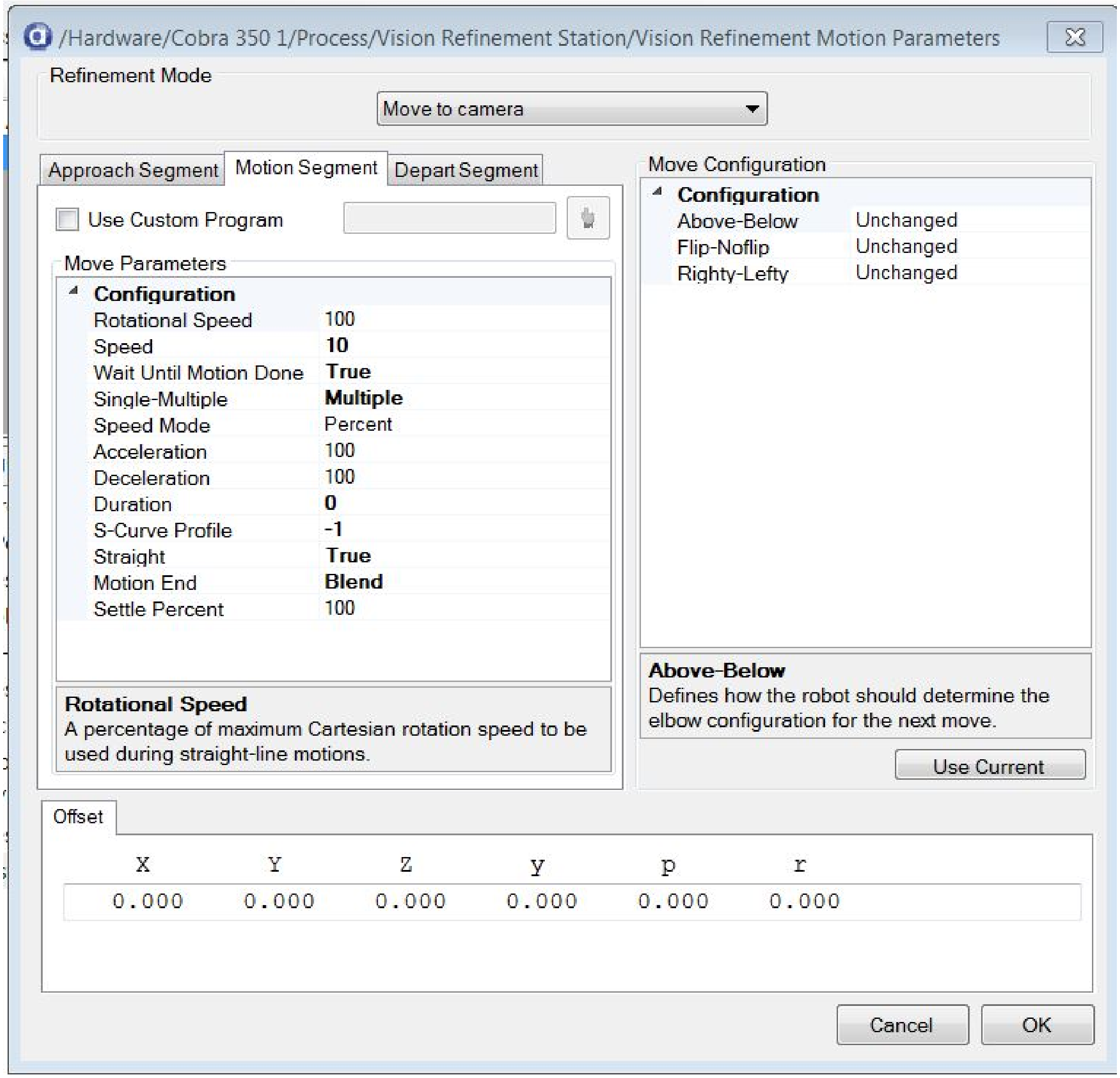
When the user defines a process that is composed of a part and a part target, the software will create motion settings that are used when processing the parts and targets. You can access the settings in the configuration item section. When one is selected, it will look like this:



Custom Pick and Place Motion Programs	V+ Program called when an instance is to be picked or placed.

Motion Settings: Refinement

A process can be configured to refine the location of a part in the gripper before placement. Once this is defined, the software will create motion settings for the refinement operation in the configuration item section. When selected, it will look like this:



Custom Refinement Motion Programs	V+ Program called when a refinement operation is to be performed.

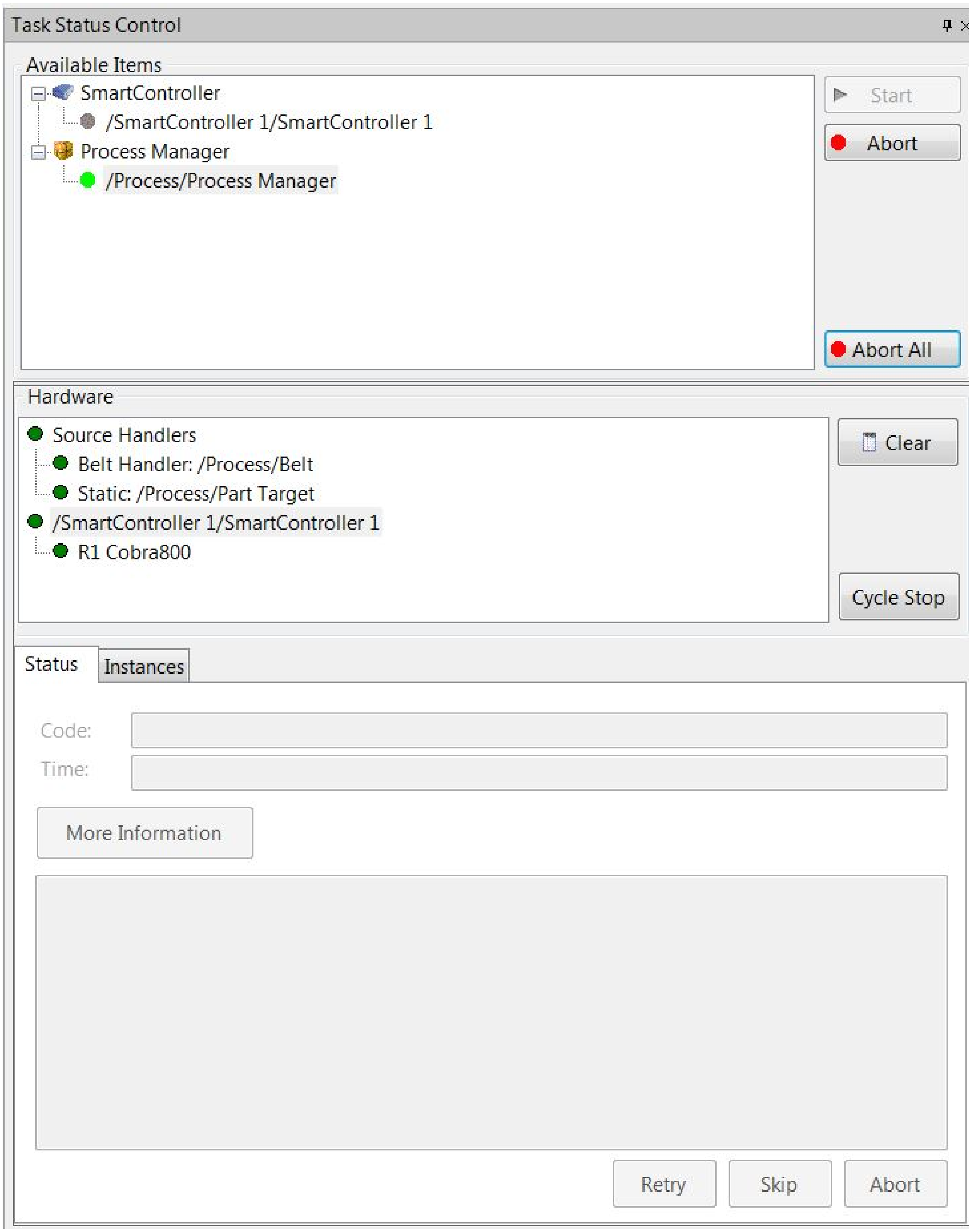
Advanced Debugging Techniques

When running a process manager, there may be times where you need to troubleshoot what the software is doing. There are several features in the software that can be used to understand the state of the system:

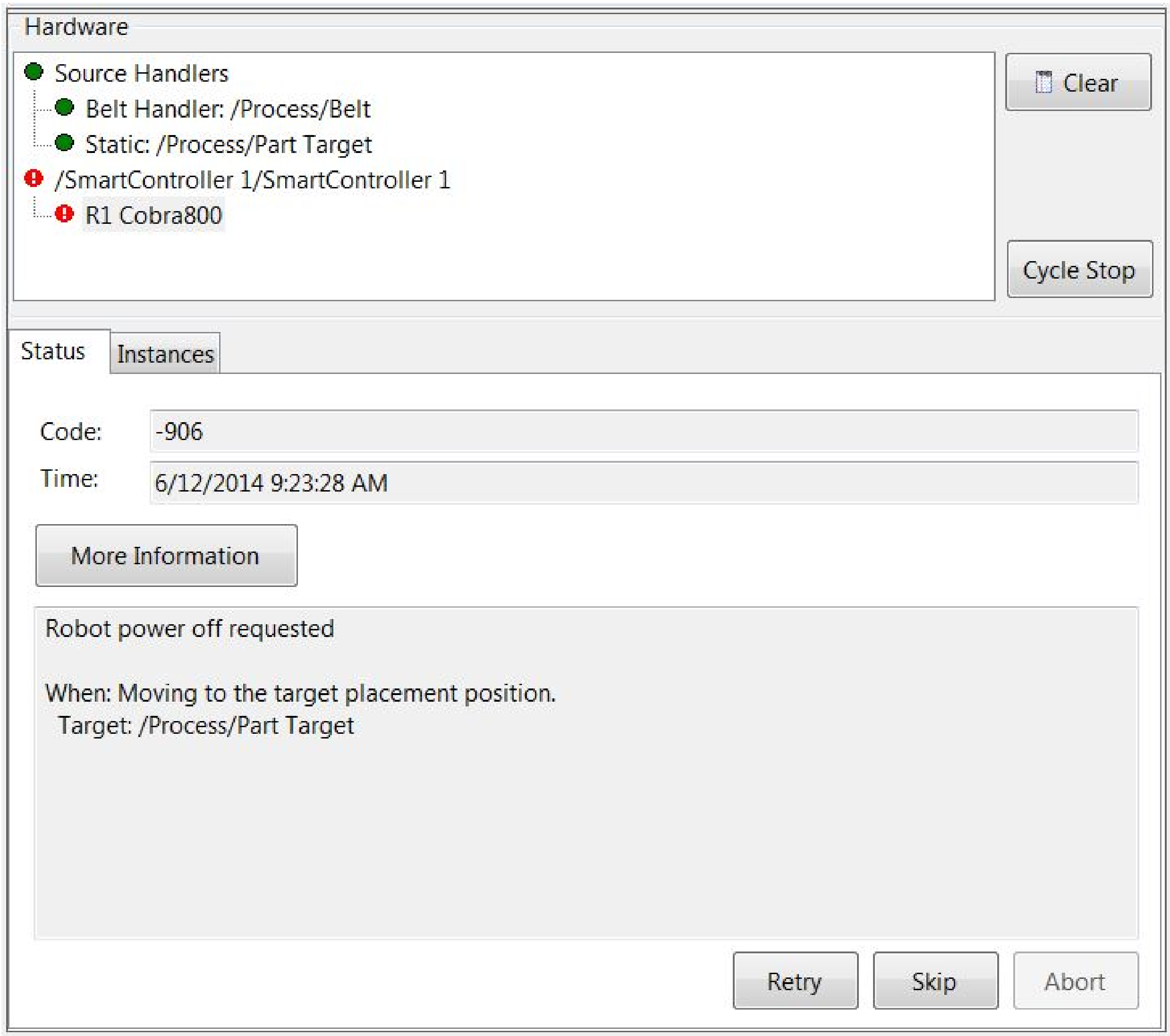
- Process Manager Runtime Control
- Event Log Display
- Triggering Additional V+ Logging
- Vision Results Logging
- Virtual Camera Image Logging
- Diagnostic Summary Display
- V+ Profiler Display
- System Monitor Display

Process Manager Runtime Control

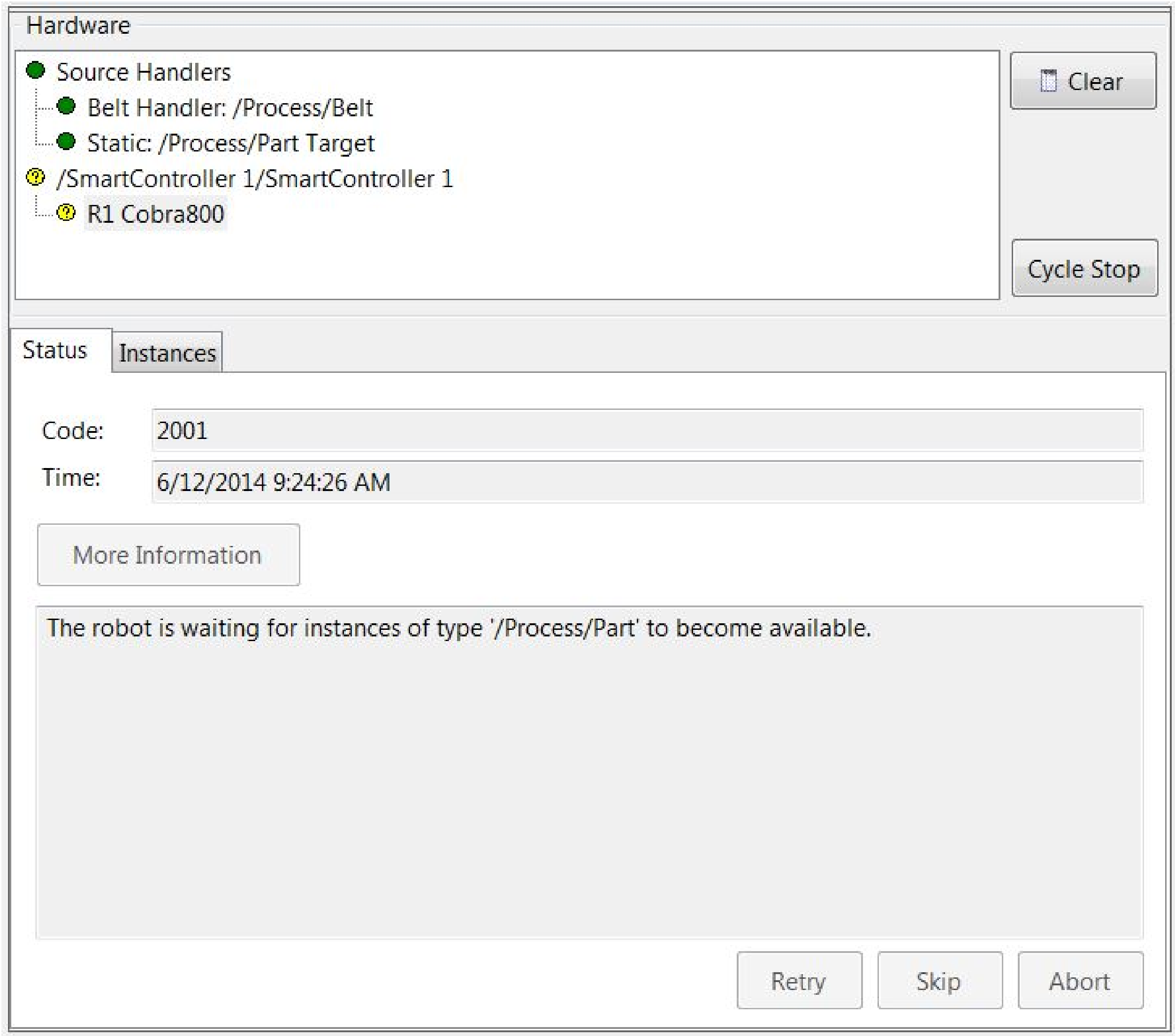
The process manager runtime control will give an overview of the status of the process manager:


Task Status Control
Available Items
SmartController
/SmartController 1/SmartController 1
Process Manager
/Process/Process Manager
Start
Abort
Abort All
Hardware
Source Handlers
Belt Handler: /Process/Belt
Static: /Process/Part Target
/SmartController 1/SmartController 1
R1 Cobra800
Clear
Cycle Stop
Status
Instances
Code:
Time:
More Information
Retry
Skip
Abort

The hardware section will show you the status of the hardware associated with the process configuration. When errors occur, they are displayed in the status tab:

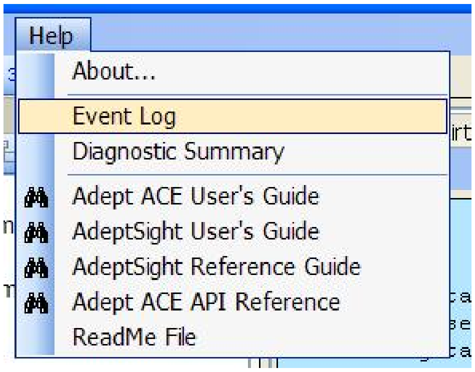


If a robot is waiting for a part or target, the process manager display will show a warning rather than an error as such:

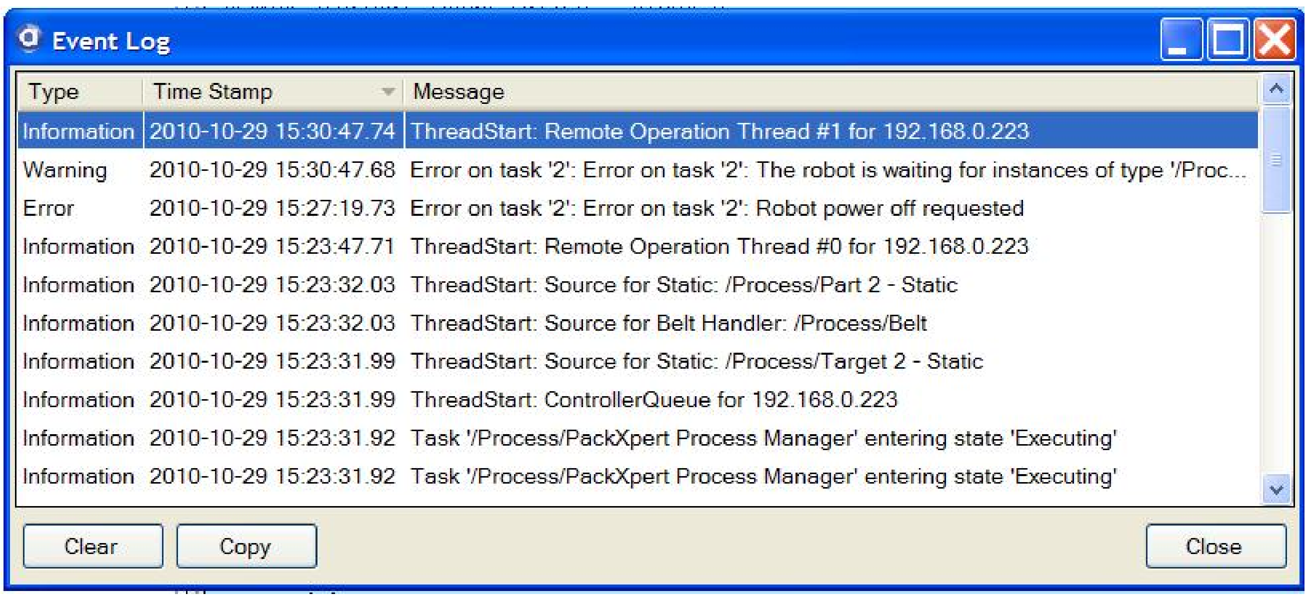


Event Log Display

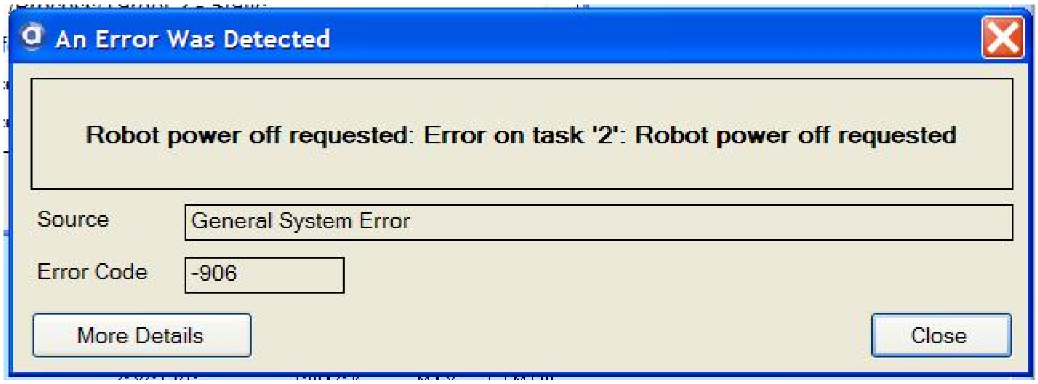
Any errors and general messages that are generated by the system are placed into the event log. The event log can be located under the main menu Help pull down:



When the event log is selected, the following window will appear:



Clicking on the columns will allow you to sort the event details. The copy button will convert the log to a text format and copy it to the clipboard. If you double click on an error or warning message, additional details will be displayed concerning the log entry. For example, double clicking on an error message will display the details of the error message:



Triggering Additional V+ Logging

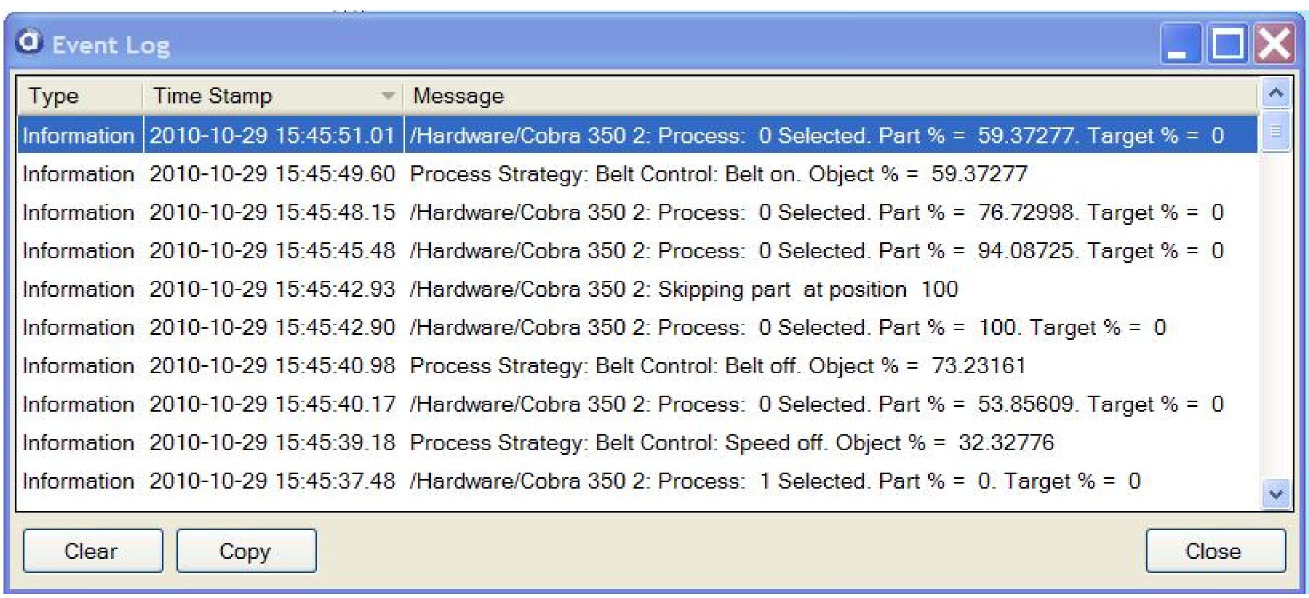
You can enable additional logging information by setting special V+ variables at the V+ monitor window. To enable logging for all process manager tasks, you can type the following at the monitor window:

SETR pm.log = TRUE

You may not want to enable logging for all V+ tasks. In that case, you can set a task specific variable. For example, to enable logging for task 3, you can type the following at the monitor window:

SETR pm.tsk.log[3] = TRUE

When additional logging is enabled, additional entries will be placed in the AceServer event log. For example:

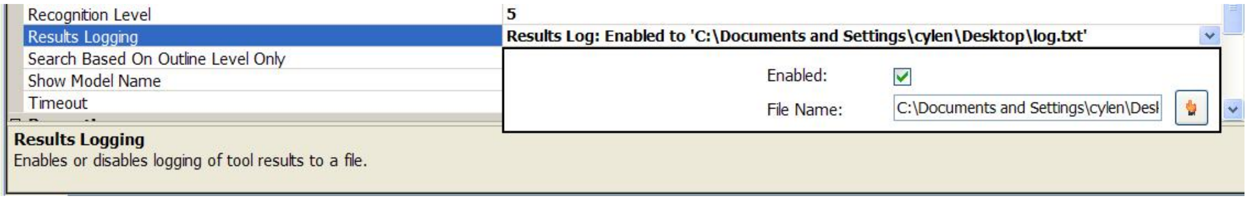


When advanced logging is enabled, the following will be written to the log:

- Process Selection information
- Belt on/off and speed control
- Robot waiting for an instance to track into a belt window
- Robot skipping an instance because it travels out of the belt window
- Latch signals detected by the belt handler program.

Vision Results Logging

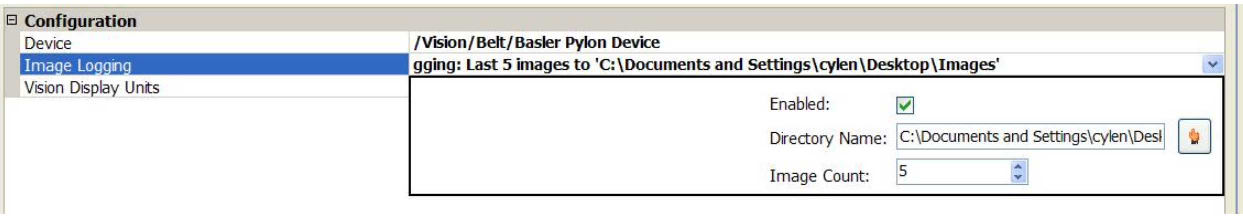
Many of the vision tools supports an advanced parameter called ***Results Logging***. When enabled, it will log the results to the specified text file:



The file extension will dictate the format for the data. If the extension is "csv", then the file will be a comma separated value file. Any other extension will result in a common text format familiar to users of AdeptSight 2.

Virtual Camera Image Logging

The virtual camera can be configured to automatically save images to a folder:

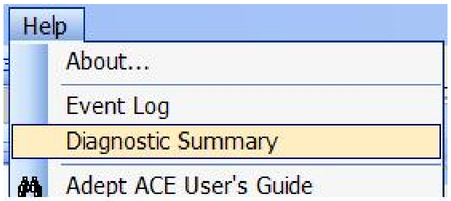


When enabled, the virtual camera will maintain a rolling buffer of image files in the specified directory. When the total number of images has been written, the oldest image will be deleted when the next image is saved.

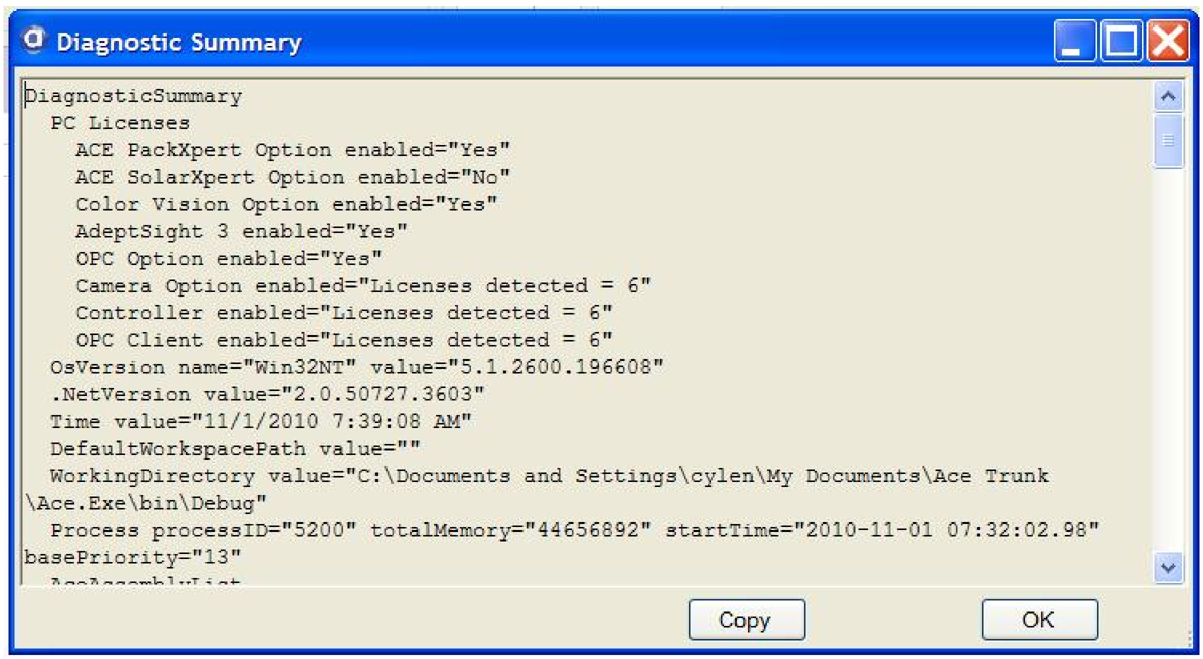
These images can be imported directly into an emulation camera and associated with a virtual camera to test vision tools offline.

Diagnostic Summary Display

The diagnostic summary display shows detailed status information on many different aspects of the system. This can be accessed under the help menu:



When the diagnostic summary is selected, the following window is displayed:



When selected, details of the process manager are placed into the summary. If a process manager is running, it will detail the following information:

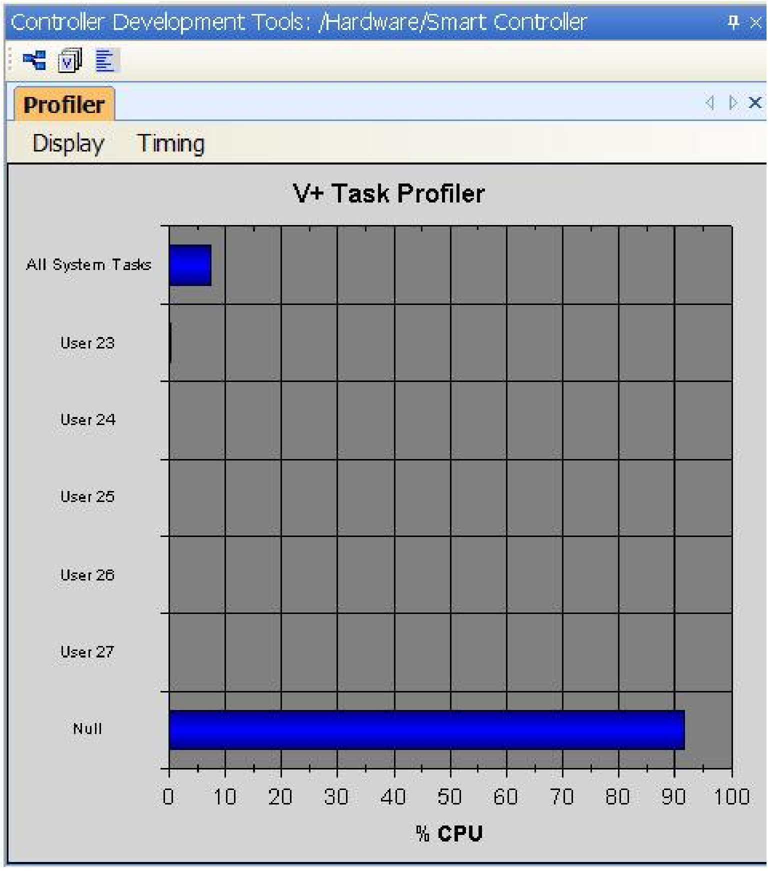
- Process manager name and run status
- The communication status of all robots and instances associated with each robot
- The instances associated with each hardware source.

This information can be used to check the allocation status.

V+ Profiler Display

Under the controller development tools is a profiler display that shows the amount of processor time being used on the selected controller. The profiler button is located on the controller development toolbar and looks like this:

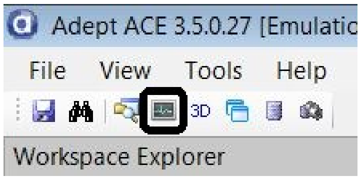
When selected, the following is displayed:



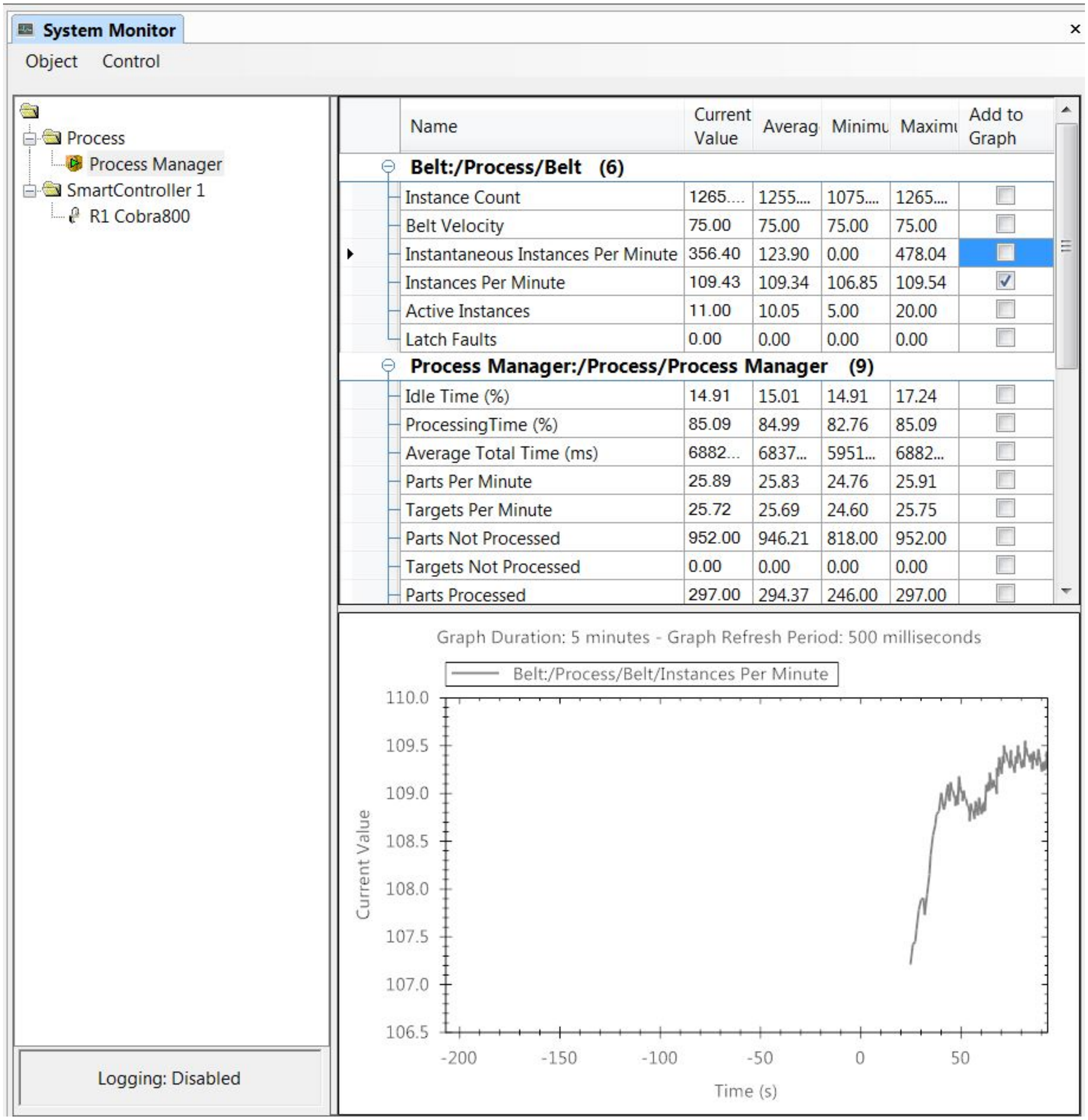
When an application is running, if you have a low amount of Null processor time, it is an indication that the V+ operating system may be overloaded and not operating efficiently.

System Monitor Display

The system monitor is a central place where statistics concerning a process manager can be accessed. The system monitor can be quickly accessed using the toolbar icon:

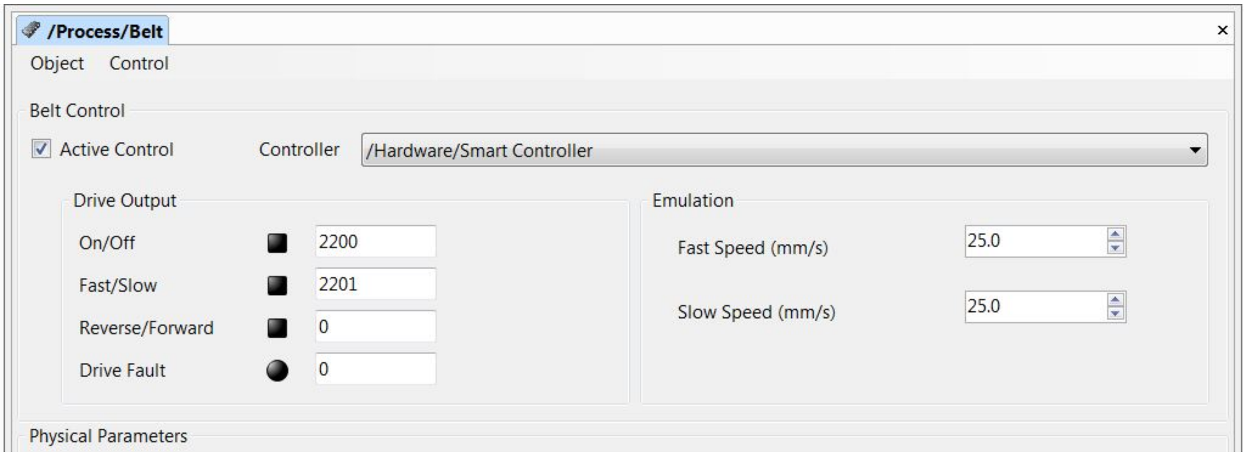


When displayed, you can select a process manager in your workspace and view/graph statistics relating to the process manager runtime:



Belt Control I/O

The ACE PackXpert software can be configured to control a belt using digital inputs and outputs associated with a controller. These belt control signals must first be defined in the Belt object in the workspace:



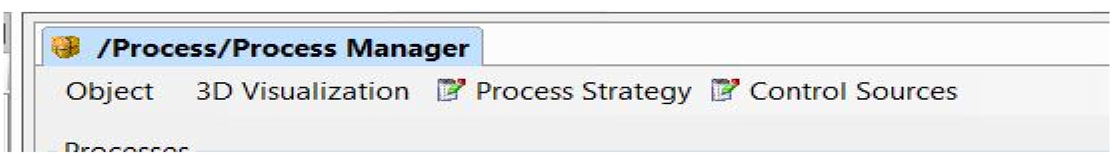
Once the controller is identified, you must specify the digital outputs from the controller that will be wired as inputs to the controller dirver. The following output signals are used by ACE PackXpert:

- On/Off: Used to turn the belt on and off. When the signal is asserted, the belt is expected to turn on.
- Fast/Slow: Relative speed of the belt. When the signal is asserted, the belt is expected to move faster. This allows for 2 levels of speed control.
- Reverse/Forward: Direction of the belt. When the signal is off, the conveyor is expected to travel in the nominal direction representing normal part flow through the system.

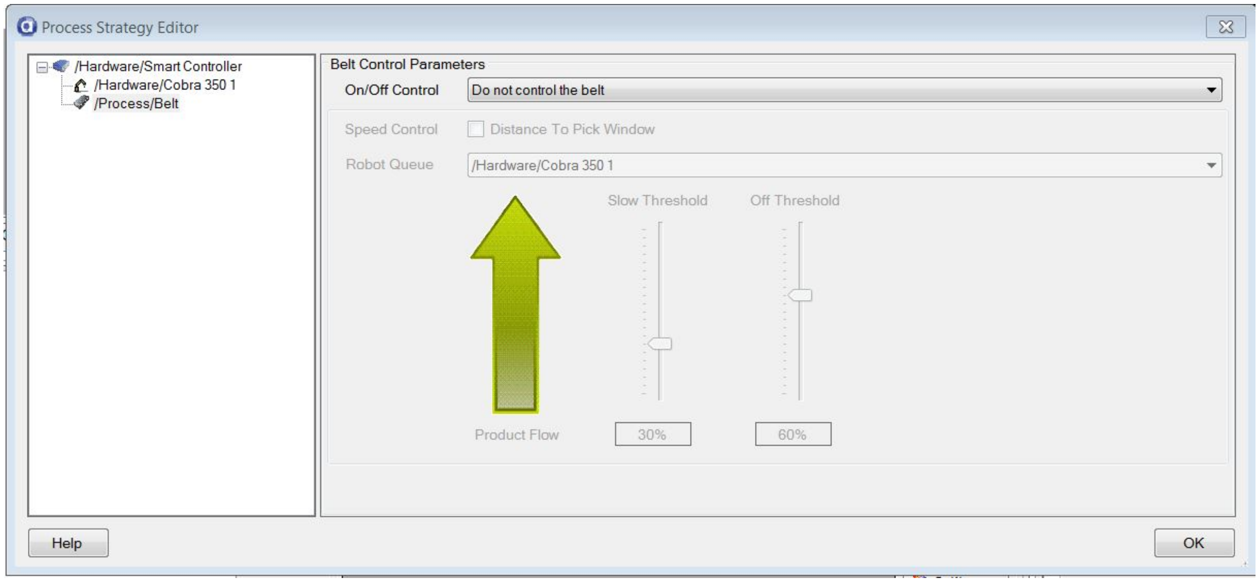
The Fast/Slow and Reverse/Forward signals are optional. If they are not used, then the signals should be set to 0.

Belt Control Settings

After the belt control signals are defined, the user needs to configure how those signals will normally be used. This is done in the Process Manager editor under the Process Strategy settings:



Once selected, navigate to the belt object in the Process Strategy Editor:



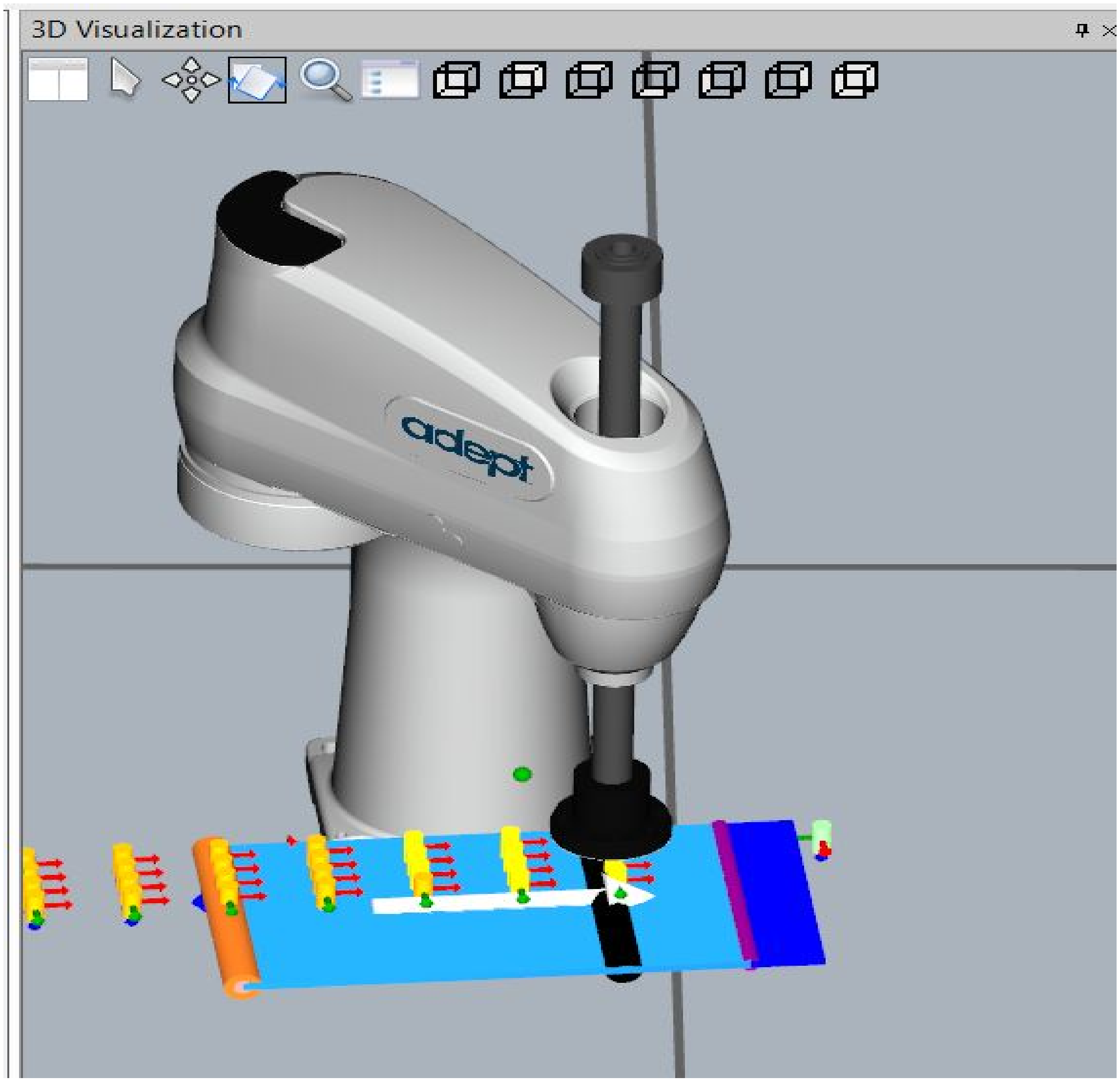
The belt control has 3 different modes of operation:

- ***Do not control the belt***: The I/O signals defined in the Belt editor are not used.
- ***Leave the belt always on***: When the process manager starts, the belt will be turned on and will remain on.
- ***Control the belt***: When the process manager starts, the belt will be turned on and off based on the product flow through the system.

When the ***Control the belt*** option is selected, the belt will be turned off based on the product flow for a selected robot. A typical configuration would look like this:

Normally, you would select the robot that is furthest downstream in your system. You can then set the *Off threshold.* If an instance on the belt reach that position in the robot belt window, the conveyor would be turned off. Once the robot processed that instance and there are no instances beyond the off threshold, the belt would be turned on. You can also enable speed control to be used. When enabled, you can set a second threshold in the belt window. When an instance reaches that threshold, the speed will be changed from fast to slow. If used correctly, you can minimize the number of times the belt will completely stop.

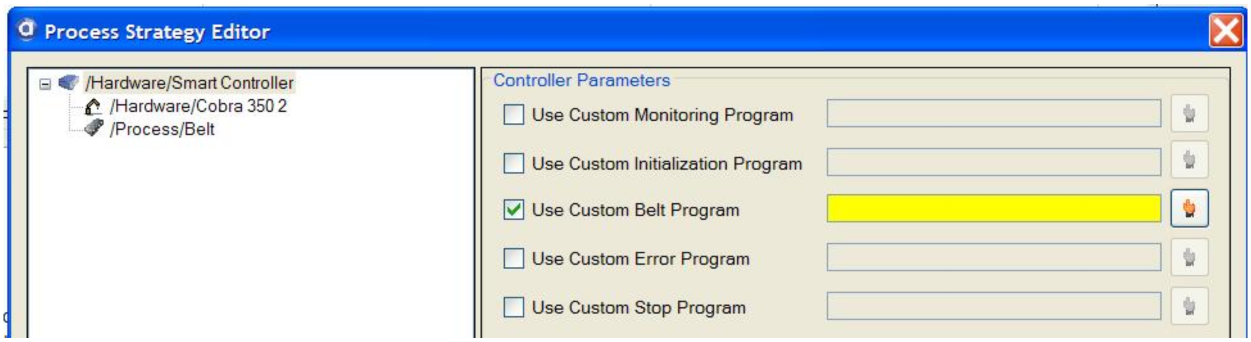
In the above graphic, the robot queue that is selected is ***/Hardware/Cobra 350 1***. When running, if an instance reaches 30% into the robot belt window, the fast/slow I/O signal defined in the belt editor will be turned off. If an instance passes the 75% threshold, the belt would be turned off. The location of the belt stop line can be visualized in the 3D virtual display. For example, if the process manager is being displayed, you might see the following:



The black line represents the stop line at the 75% position along the belt window.

Belt Control: Defining a Custom Belt Program

The default belt monitoring program is responsible for monitoring all belts being controlled on a single controller. That program monitors the part and target queues associated with the specified robots and applying the I/O signals associated with each belt. If the default behavior is not acceptable, the default program can be customized in the process strategy configuration section located here:



When a custom belt control program is specified, you will start with a copy of the default belt monitoring program. That program, as noted, is responsible for monitoring all belts that are defined in the process manager. The person customizing the belt program is responsible for ensuring the customized program behaves as expected for the application.

Belt Control: Default Belt Program Behavior

In order to provide a custom belt program, it will be helpful to review the workings of the default belt program. To do this, we will break the code into sections.

Iterate Through the Belts

In the first section, the default belt program starts by going through all defined belts. If any belts are configured to not be controlled, those belts are skipped.

```
AUTO REAL farthest.object, i, mode, off.threshold, on.output
AUTO REAL rob.idx, speed.mode, speed.output, speed.threshold
AUTO $source.belt

; Process through each belt being monitored

FOR i = 0 TO psp.belt[psp.bltcount]-1

    ; Get the control mode

    mode = psp.belt[i,psp.blt.md]

    ; If no belt control, then skip to next belt

    IF (mode == psp.blt.md.no) THEN
        NEXT
    END
```

Evaluate Type of Belt Control

For any belts that are being controlled, if the process manager is actively running, it will check to see if the belt control was configured to "always on.". If that is the case, the belt is turned on. If the belt mode is not always on, it will find the farthest downstream position of any part or target instance associated with the robot on the belt being checked. This is done in the call to ***pm.ps.obj.dist***. The farthest downstream instance position is compared against the off and speed threshold. The appropriate signals are asserted based on those comparisons.

It is important to note that the instance position is based on all instances currently in the robot queues. Instances on the PC that have not been sent to the robot are not taken into account when calculating the farthest downstream instance position.

```
        IF (running == TRUE) THEN

            ; If system is running and the belt mode
            ; is set to always on, make sure belt is on.

            IF (mode == psp.blt.md.on) THEN

                IF (SIG(-on.output)) THEN
                    SIGNAL on.output
                END
            ELSE

                ; Active on/off control mode is enabled.
                ; Get the speed mode enumeration

                speed.mode = psp.belt[i,psp.blt.spdm]

                ; Get the name of the belt

                $source.belt = $psp.belt[i,psp.blt.name]

                ; Go through each object in the robot queue and get the object that
                ; is furthest upstream.

                rob.idx = psp.belt[i,psp.blt.rob]
                CALL pm.ps.obj.dist(rob.idx, $source.belt, farthest.object)

                ; Get the thresholds for on/off and speed

                off.threshold = psp.belt[i,psp.blt.offt]
                speed.threshold = psp.belt[i,psp.blt.slwt]

                IF (speed.mode) THEN
                    IF (farthest.object > speed.threshold) THEN
                        SIGNAL -speed.output
                    ELSE
                        SIGNAL speed.output
                    END
                END

                IF (farthest.object > off.threshold) THEN
                    SIGNAL -on.output
                ELSE
                    SIGNAL on.output
                END
            END
```

Run Check: Turn Belt Off

If the program was called when the process manager was shutting down, the code would turn off the belt if belt control was enabled.

```
        ELSE

            ; We are not running, turn belt off if it is active control

            IF (mode == psp.blt.md.ctrl) THEN
                SIGNAL -on.output
            END
        END
    END
```

Belt Monitoring

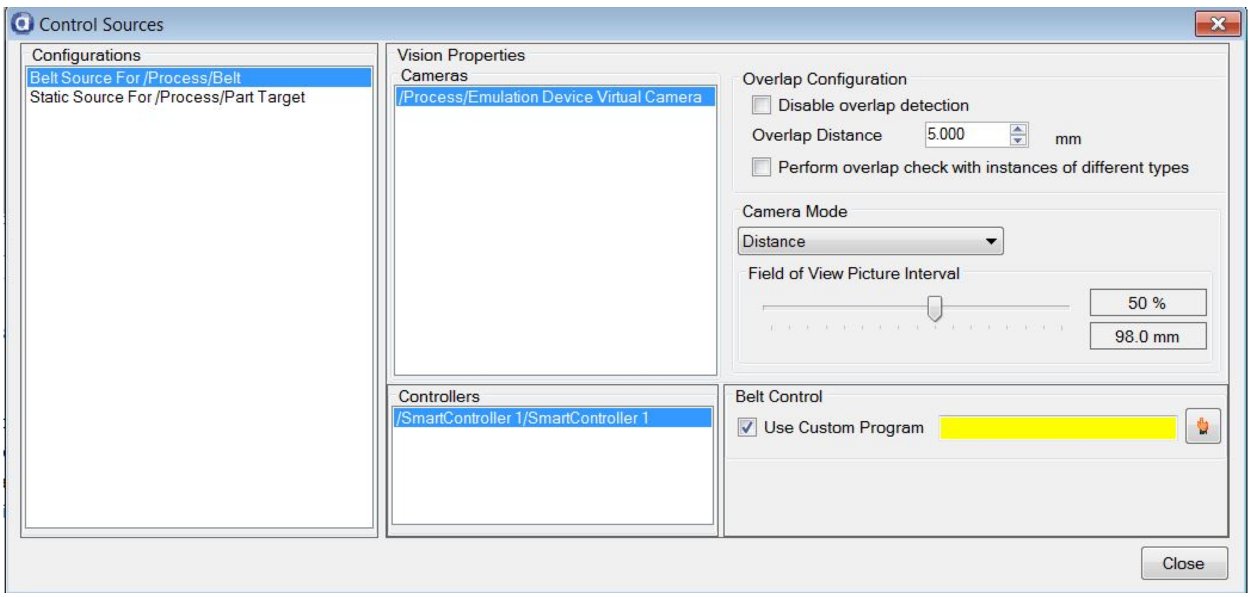
A part or target can be configured for presentation on a belt in one of three modes:

- Belt Camera
- Latch Input signal
- Instances created at a spacing interval

To handle these 3 cases, the process manager must download and run a V+ program. This program must monitor and control when parts and targets are created based on feedback from the belt encoders. As the various belt encoders change position, the program will send messages to the PC triggering the creation of parts and targets.

Belt Monitoring

If the default behavior of the belt program is not sufficient, then the user can customize the default belt control program. The customization can be done in the control source editor:



The belt control program is a rather long program. To discuss the behavior, it can best be consumed by breaking it into separate sections:

- Initialization
- Monitor encoder motion
- Monitor spacing points
- Monitor hardware latches
- Monitor camera points

Initialization

When the belt program starts, it will first initialize all tracking structures. The code section starts around here:

```
; Initialize Variables

       ; Read the encoders initially and send the status to the PC

       FOR i = 0 TO pm.belt[blt.idx,pm.bcf.encoder]-1
           belt.num = pm.belt[blt.idx,pm.bcf.encoder,i]
           CALL pm.read.encoder(belt.num, encoder.value[blt.idx,i], encoder.vel[blt.idx,i])
...
```

It will initialize the following:

- Current position and velocity of the belt
- The spacing point based on the current belt position
- The camera reference point based on the current belt position
- Clears all tracking structures

Monitor Encoder Motion

Once the belt control finishes initializing control variables, it will monitor the encoder position and velocity. After it checks all of the encoders, it will periodically send the current encoder position and velocity to the PC using the call ***pm.ace.blt.pos***.

```
DO

    CALL pm.blt.init(blt.idx)             ; Monitor belt tracking structures from the PC
    CALL pm.blt.chk.dflt(blt.idx)         ; Check drive fault status

    ; Read the current status of the encoders
    ; Periodically update the PC with the encoder veliocity. If a 5%
    ; change in encoder velocity is detected, then send immediately

    send.encoder = (TIMER(-3) > send.time)
    FOR i = 0 TO pm.belt[blt.idx,pm.bcf.encoder]-1
        belt.num = pm.belt[blt.idx,pm.bcf.encoder,i]
        CALL pm.read.encoder(belt.num, encoder.value[blt.idx,i], encoder.vel[blt.idx,i])
        IF (last.enc.vel[blt.idx,i] <> encoder.vel[blt.idx,i]) THEN
            IF (last.enc.vel[blt.idx,i] == 0) OR (encoder.vel[blt.idx,i] == 0) THEN
                send.encoder = TRUE
            ELSE
                IF ABS((last.enc.vel[blt.idx,i]-encoder.vel[blt.idx,i])/encoder.vel[blt.idx,i]) > 0.05 THE
                    send.encoder = TRUE
                END
            END
        END
        last.enc.vel[blt.idx,i] = encoder.vel[blt.idx,i]
    END

    IF (send.encoder) THEN
        FOR i = 0 TO pm.belt[blt.idx,pm.bcf.encoder]-1
            CALL pm.ace.blt.pos(blt.idx, i, encoder.value[blt.idx,i], encoder.vel[blt.idx,i])
        END
        send.time = TIMER(-3)+send.interval
    END

    IF (TIMER(-3) < scan.time) THEN
        WAIT
        NEXT
    END
    scan.time = TIMER(-3)+scan.interval
```

Monitor Spacing Points

In the initialization section, tracking variables are initialized for the spacing reference points. When a spacing part or target is defined, the user specifies the distance between the generated instances. This spacing parameter is used, along with the scale, to determine how much the conveyor must move before a part is generated. After the conveyor moves the specified distance from the reference point, it will send a message to the PC to create an instance using ***pm.ace.blt.sp***.

```
; Look through the spacing reference points. If the encoder
; has reached a reference point, send the reference to ACE.
; Loop and make sure to send all reference points.

FOR i = 0 TO pm.belt[blt.idx,pm.bcf.sp.obj]-1

    idx = pm.belt[blt.idx,pm.bcf.sp.enc,i]
    encoder = encoder.value[blt.idx,idx]
    scale = pm.belt[blt.idx,pm.bcf.scale,idx]

    DO

        CALL pm.blt.travel(encoder, space.ref[blt.idx,i], distance)
        distance = distance*scale

        IF distance >= pm.belt[blt.idx,pm.bcf.spacing,i] THEN

            ; Update the reference position and handle
            ; encoder rollover.

            space.ref[blt.idx,i] = space.ref[blt.idx,i]+(pm.belt[blt.idx,pm.bcf.spacing,i]/scale)
            CALL pm.blt.convert(space.ref[blt.idx,i])

            $spacing.item = $pm.belt[blt.idx,pm.bcf.sp.obj,i]
            CALL pm.ace.blt.sp(blt.idx, $spacing.item, space.ref[blt.idx,i], "")

        END
    UNTIL distance < pm.belt[blt.idx,pm.bcf.spacing,i]
END
```

If you would like a custom tag to be associated with instances created by the spacing event, the call to ***pm.ace.blt.sp*** can be modified to pass in a tag for the last argument.

Monitor Hardware Latches

The program will go through the list of encoders that need to be monitored for latches. When a latch is detected, it will check the latched encoder position against the minimum latch distance that was defined in the belt editor. If the latch distance is sufficient, the latch information is sent to the PC using ***pm.ace.blt.ltch***.

```
        ; Look through the latch items. If a latch had been seen
        ; previously, then ensure the new latch is spaced far enough
        ; apart from the previous latch.

        FOR i = 0 TO pm.belt[blt.idx,pm.bcf.latch]-1

            idx = pm.belt[blt.idx,pm.bcf.latchenc,i]
            encoder = pm.belt[blt.idx,pm.bcf.encoder,idx]+1
            scale = pm.belt[blt.idx,pm.bcf.scale,idx]

            latch.num = ABS(LATCHED(-encoder))
            WHILE (latch.num) DO

                ; Get the index associated with the latch that was
                ; found and read the latch. The latch may not be one we are
                ; monitoring for. In that case, ignore the latch

                CALL pm.blt.gltchix(blt.idx, latch.num, idx)
                latch.value = DEVICE(0,encoder-1,sts,4)

                IF (idx >= 0) THEN

                    ; If latch distance checking is in effect, we must check
                    ; if the distance between latches is large enough.

                    IF latch.ref.chk[blt.idx,idx] THEN
                        CALL pm.blt.travel(latch.value, latch.ref[blt.idx,idx], distance)
                        scale = pm.belt[blt.idx,pm.bcf.scale,i]
                        distance = distance*scale
                        IF (distance < pm.belt[blt.idx,pm.bcf.latchdst,idx]) THEN
                            latch.num = 0
                        END
                    END

                    ; If there is a latch to report, send to the PC.
                    ; Update the latch reference for latch distance
                    ; checking when the next latch is detected.

                    IF latch.num THEN
                        IF (sts >= 0) THEN
                            CALL pm.ace.blt.ltch(blt.idx, encoder-1, latch.num, latch.value, "")
                            latch.ref[blt.idx,idx] = latch.value
                            latch.dist[blt.idx,idx] = 0
                            latch.ref.chk[blt.idx,idx] = TRUE
                        END
                    END
                END

                latch.num = ABS(LATCHED(-encoder))
            END
```

Some applications may want to change the logic for when latches are reported. For example, an application may require only sending every-other latch so a robot only processes 50% of the instances generated from the latch.

Additionally, if you would like a custom tag to be associated with instances created by the latch event, the call to ***pm.ace.blt.ltch*** can be modified to pass in a tag for the last argument.

Monitor Camera Points

In the initialization section, tracking variables are initialized for the camera reference points. When a belt-camera based part or target is referenced in the process manager, the user can specify if the picture is triggered by conveyor motion or a digital signal. When conveyor motion is specified, the user will specify a camera picture percent. This is used, along with the scale and the average field of view size, to determine how much the conveyor must move before a camera picture request is generated. After the conveyor moves the specified distance from the reference point, it will send a message to the PC to perform a picture using ***pm.ace.blt.cam***.

If the user specified a signal trigger, pictures will be taken when the signal is asserted.

```
; Look through the belt camera items

FOR i = 0 TO pm.belt[blt.idx,pm.bcf.cameras]-1

    take.picture = FALSE

    ; If operating in picture distance mode, then calculate how far the encoder
    ; has moved and see if a picture needs to be taken.

    IF pm.belt[blt.idx,pm.bcf.pic.md,i] == pm.bc.pmd.dist THEN

        idx = pm.belt[blt.idx,pm.bcf.cam.enc,i]
        encoder = encoder.value[blt.idx,idx]
        scale = pm.belt[blt.idx,pm.bcf.scale,idx]

        CALL pm.blt.travel(encoder, cam.ref[blt.idx,i], distance)
        distance = distance*scale

        inc = pm.belt[blt.idx,pm.bcf.cam.fov,i]*pm.belt[blt.idx,pm.bcf.fov.intv,i]

        IF distance > inc THEN

            take.picture = TRUE

            ; Update the reference position and look for
            ; rollover. If the camera gets behind by 2x the
            ; distance, set the reference based on the encoder
            ; position. The belt is moving faster than the
            ; camera can process.

            IF distance > 2*inc THEN
                cam.ref[blt.idx,i] = encoder-(inc/scale)
            ELSE
                cam.ref[blt.idx,i] = cam.ref[blt.idx,i]+(inc/scale)
            END
            CALL pm.blt.convert(cam.ref[blt.idx,i])

        END
    ELSE

        ; Operating in trigger mode. If the trigger signal goes on, then we need to take a picture

        IF SIG(pm.belt[blt.idx,pm.bcf.pic.trig,i]) THEN
            take.picture = TRUE
        END
    END

    IF take.picture THEN

        $spacing.item = $pm.belt[blt.idx,pm.bcf.cameras,i]
        remote = pm.belt[blt.idx,pm.bcf.cam.isrm,i]
        CALL pm.ace.blt.cam(blt.idx, $spacing.item, remote, cam.ref[blt.idx,i])

        ; If we are in trigger mode, then we need to wait until the signal goes low
        ; But we check if the process manager has not been stopped
        ; If we are in emulation mode, we will simply reset the trigger with no wait

        IF (pm.belt[blt.idx,pm.bcf.pic.md,i] == pm.bc.pmd.trig) THEN
            IF sv.emulate.mode THEN
                SIGNAL -pm.belt[blt.idx,pm.bcf.pic.trig,i]
            ELSE
                DO
                    CALL pm.chk.stat(run)
                    WAIT
                UNTIL NOT ((SIG(pm.belt[blt.idx,pm.bcf.pic.trig,i]) AND run == TRUE))
            END
        END
    END
END
```

Conceptual Overview

The process manager is an application development module within ACE. The goal is to allow for the development of packaging applications.

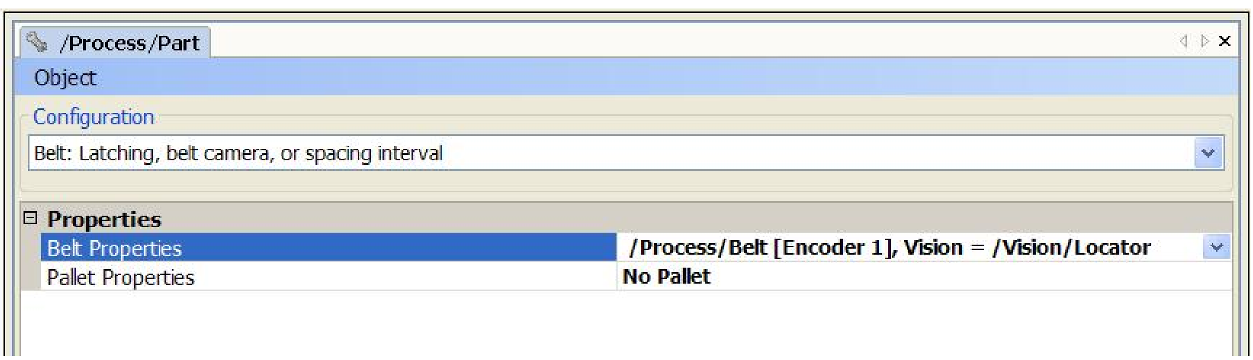
The term "Packaging" represents a class of applications that involve belt tracking of product which is being processed by multiple robots spread along a conveyor. These robots must coordinate their actions and ensure that the finished product is adequately handled. The classic example is multiple robots placing chocolates into boxes of candy. These robots must coordinate their actions and ensure that all boxes of chocolates have all slots filled by the time the last robot is done with a given box. In this application domain, multiple robots may be capable of processing the same part, with or without additional rules and restrictions. Additionally, robots are capable of performing multiple tasks. Lastly, multiple instances of a given product must be tracked simultaneously.

These kinds of applications focus heavily on the aspect of synchronizing multiple robots where a given robot's current behavior is selected based on availability of product. Because of the robot synchronization requirement and the fact that multiple instances of product must be tracked across multiple robot domains, these applications need to be built from a "top down" approach. These applications start with the definition of "parts" and "targets". Then the user describes which robots can do with parts and targets. The concepts of the locations, paths, IO are all derived from the top down approach.

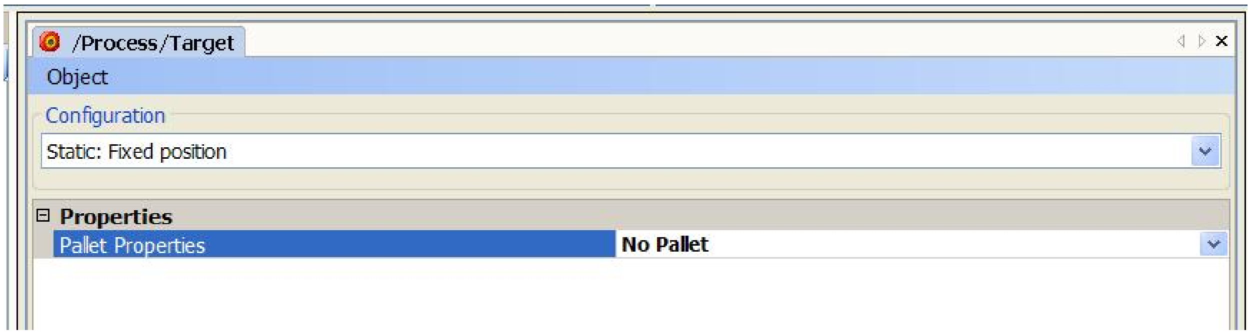
Top Down Approach: An Application Framework

In order to correctly synchronize instances between multiple robots and controllers, the software takes a "top down" approach to defining the application. Since there is a lot of infrastructure required to track and synchronize instance information, the process manager provides a large amount of pre-packaged software and behaviors. Many of the default software behaviors can be customized.

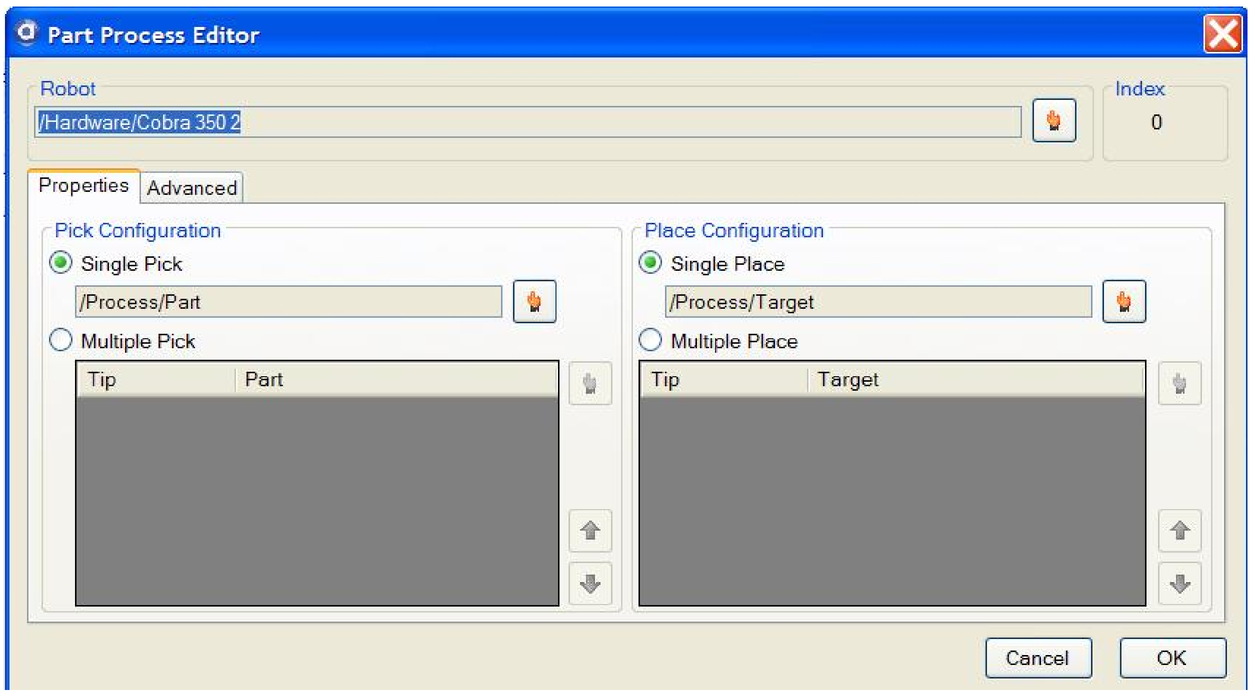
Assuming you have a workspace that contains a controller and a robot object, an application starts with the definition of a part that needs to be picked and a belt object the part may be presented on:



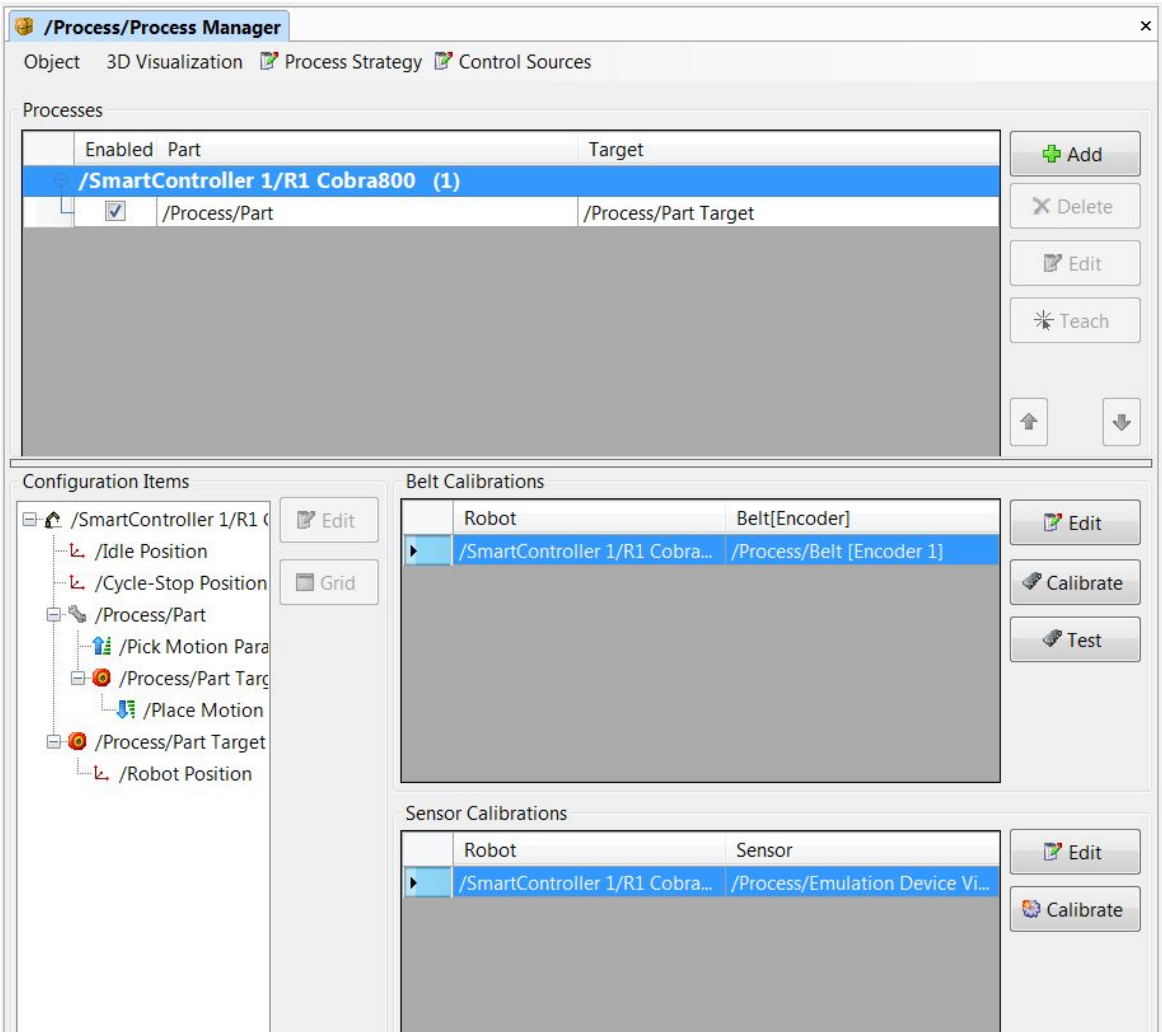
Once the part is created, the target where the part will be placed must likewise be defined:



After defining the part and targets, the user will create a processes in the process manager



Once all the parts, targets, and processes are define, the software creates a series of data points that are required by the framework: robot configuration data, robot-to-belt calibrations, and robot-to-sensor/camera/latch calibrations. This data can be directly accessed in the process manager editor:



Hardware Sources: Provide Instances

When a process manager application is run, the framework uses the part and targets referenced in the process list to create many tracking structures. In general terms, each part and target has a tracking object that identifies instances that are available for processing. Sometimes these objects are called hardware sources and they are represented in the software as a ***BaseSource***.

Each **BaseSource** will identify what instances are available for processing. For a given part, a different kind of ***BaseSource*** will be created depending on the configurations of the part.

In the example above, the source object used to manage the belt based part is a ***BeltSource***. The ***BeltSource*** is responsible for taking pictures using the camera and matching the vision coordinates with latch information sent from the controllers for any robots that can pick that part.

The source object used to manage the static target is the ***StaticPartTargetSource.***

Often times, the ***BaseSource*** will communicate with programs running in V+ in order to manage the availability of instances. In the case of a BeltSource, each source will have a V+ belt program that is responsible for monitoring the belt encoders and will send messages to the ***BeltSource*** on the PC when pictures need to be taken and latches are detected.

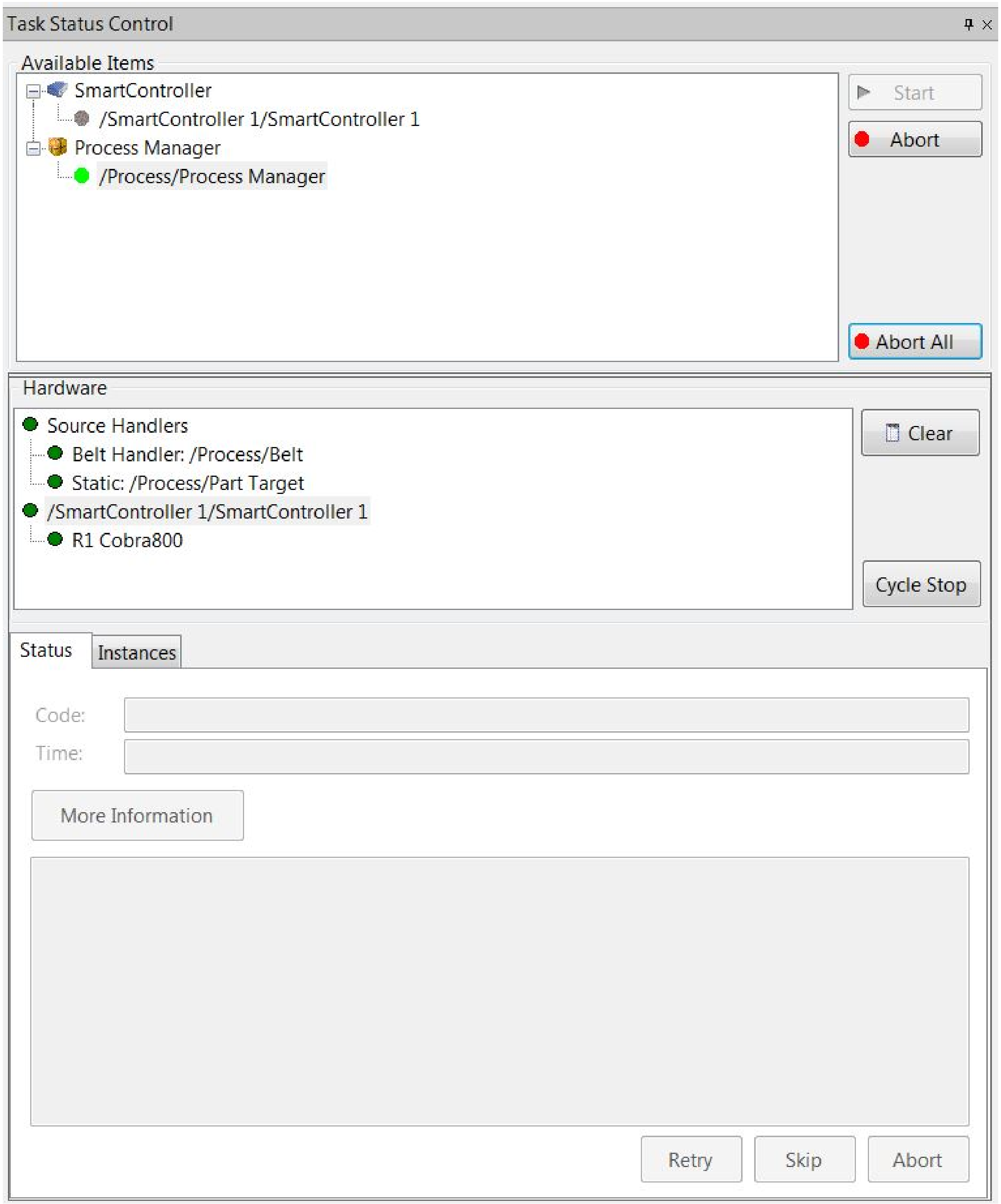
Controller Queue: Process Instances

When a process manager application is run, the framework uses the process list to identify all the robots that are capable of processing parts and targets. Each ***RobotStation*** is capable of consuming the instances that are provided by the ***BaseSource*** objects.

Process Manager Runtime Control: Hardware Display

The runtime control for the process manager is located in the ACE Task Status Control. Each process manager will have an entry under the process manager section. When a process manager is selected, the runtime control and status for that process manager is displayed in the Task Status Control.

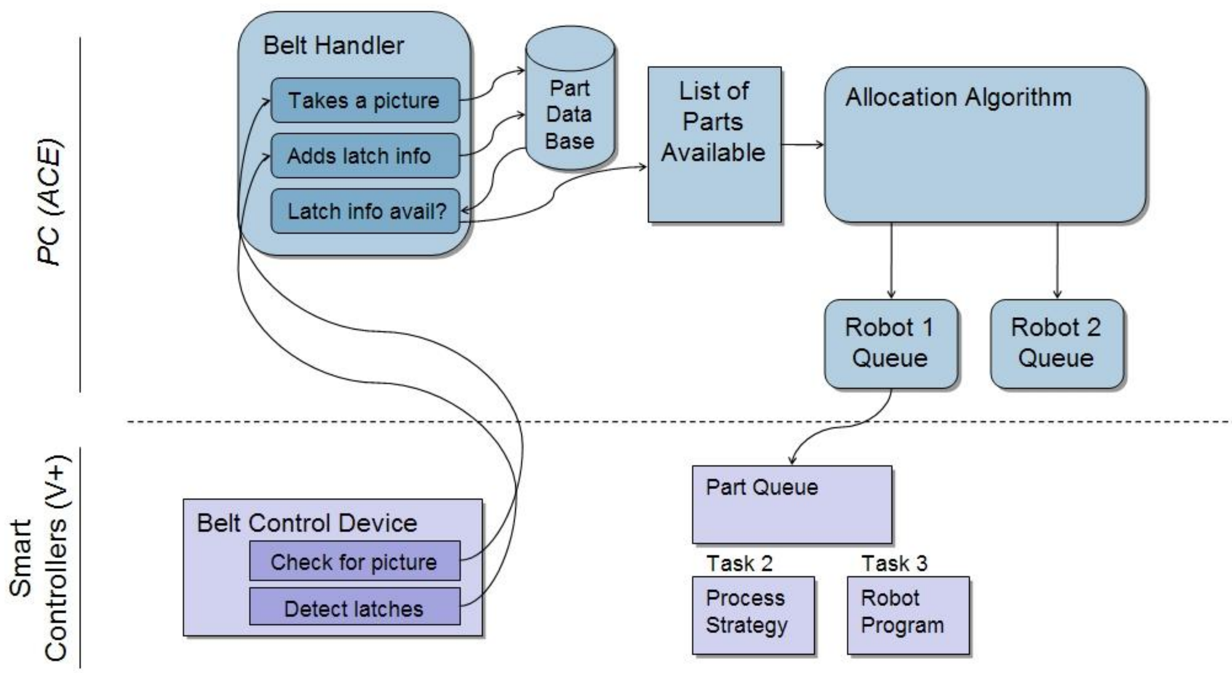
The process manager runtime control displays the ***BaseSoruce*** and ***RobotStation*** objects:


Task Status Control
Available Items
SmartController
/SmartController 1/SmartController 1
Process Manager
/Process/Process Manager
Start
Abort
Abort All
Hardware
Source Handlers
Belt Handler: /Process/Belt
Static: /Process/Part Target
/SmartController 1/SmartController 1
R1 Cobra800
Clear
Cycle Stop
Status
Instances
Code:
Time:
More Information
Retry
Skip
Abort

You can use the Instances tab on the runtime display to view instances associated with a specific hardware source.

Process Strategy: Data Models

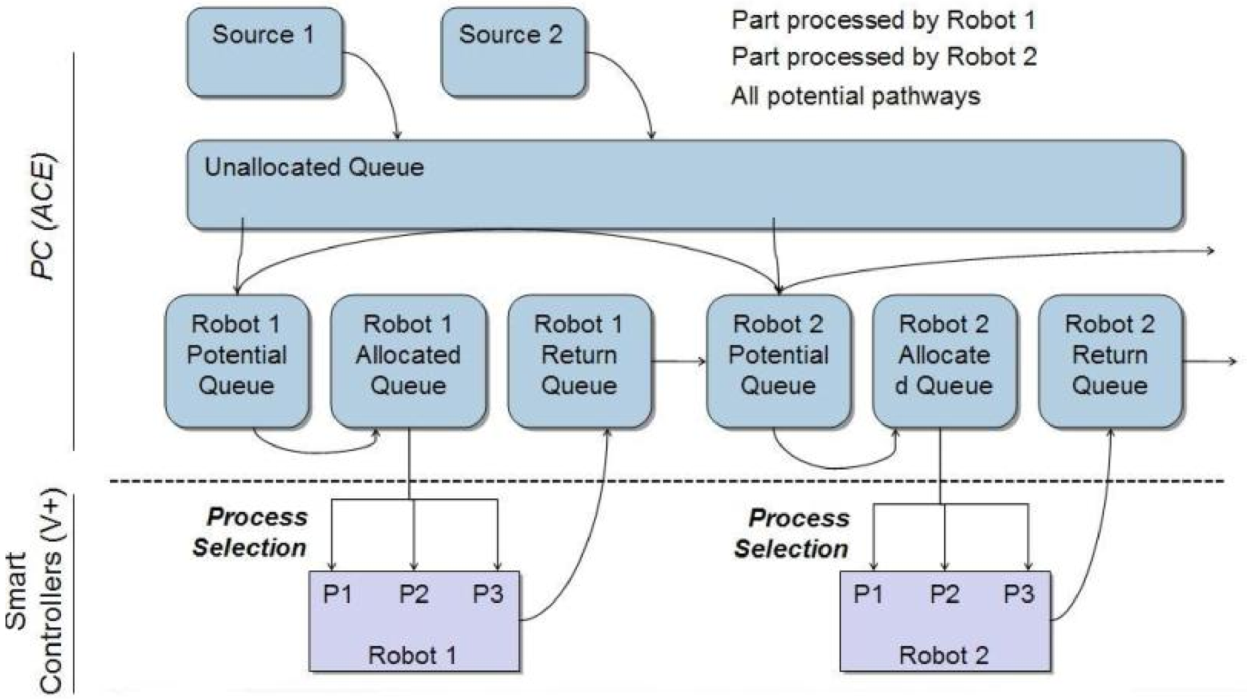
The process strategy is the object that is responsible for associating instances supplied by the hardware sources with robots that can process those instances. As an example, here is a diagram showing the data flow for the vision-located belt part:



When the application runs, a V+ belt control program is launched on the robot controller. The belt control program monitors for conveyor motion and input latchs. As the conveyor advances, it will send requests to the belt handler to take a picture with the camera. Additionally, the belt control program sends the latch information to the belt handler as they are detected.

The belt handler will take the instances located by a camera vision operation and associate it with latches received from the controller. Once it identifies instances for processing, it adds it to the list of parts that are available.

The allocation algorithm is notified when instances are made available for any hardware source. At that point, it will locate the farthest upstream robot that can process the instance. It will allocate that instance to that robot. This diagram shows the data flow for how RobotStation queues are managed:



When an instance is identified by a hardware source, it is unallocated. The process strategy code will identify the first robot capable of processing the instance. For belt based instances, this will be the farthest upstream robot that can pick the part.

When the instance is associated with a given robot, it is put in a "potential queue." These are instances that are allocated to a robot but have not been sent to the controller for processing by the robot controller. If space is available in the queue of instances on the robot controller, the part is moved to the "allocated queue" and sent to the robot controller.

When the robot controller is finished processing an instance, it will send a message to the process strategy indicating if the instance was processed or if the part was skipped. If the part was processed, the instance is removed from all tracking structures on the PC. If the part was not processed, the instance is moved to the "potential queue" of the next robot that can process the instance. If there are no more robots capable of processing the instance, the instance is removed from all tracking structures.

Custom Allocation Scripts

The process manager allows for customization of the allocation logic if the default allocation strategy does not fit your needs.

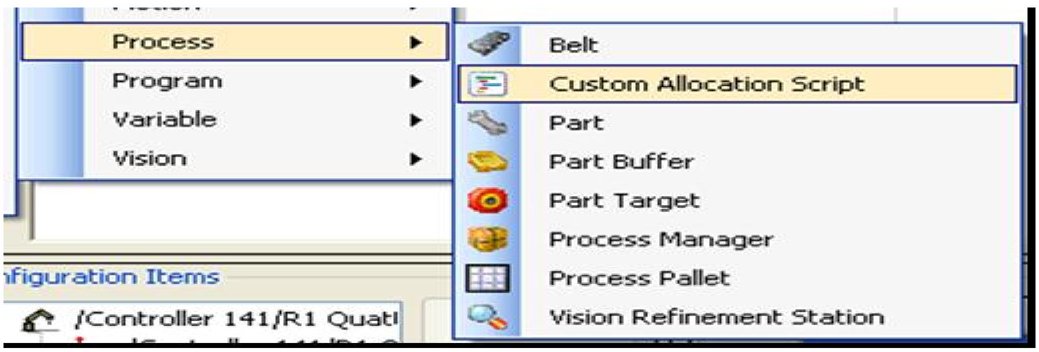
The default allocation strategy is designed so that each robot will pick as many instance as possible and instances are ordered from farthest downstream to farthest upstream. Any instances not processed by a given robot are placed in the next robot queue that is capable of processing the instance.

Some examples of situations that would require a custom allocation script:

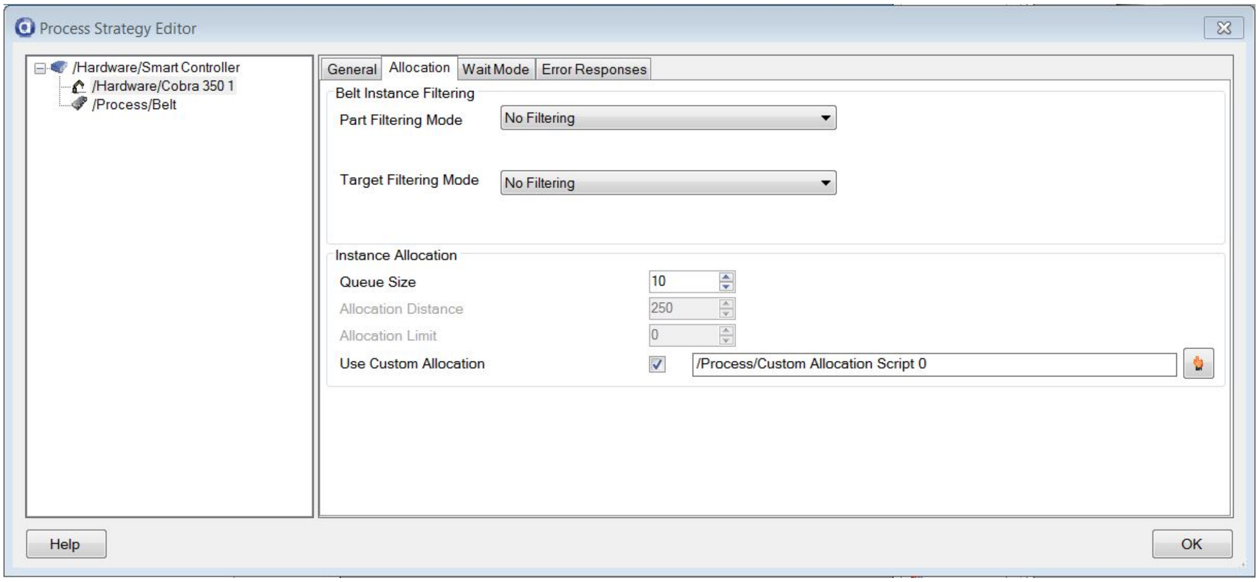
- Instances need to be picked up in rows perpendicular to the flow of parts. In this case, instances may need to be grouped into rows then sorted left-to-right.
- Instances need to be "load balanced" between a certain number of robots.

Creating an Allocation Script

An allocation script can be added to the workspace as such:



Once created, the script must be associated with each robot that will use the script:



Each robot maintains a queue size which defines how many instances of a given kind can be send to a robot controller. This queue size parameter is not the total number of instances, it is the total number of instances of each part or target that can be sent. The robot controller maintains a different queue for each possible part or target that a robot can process. This parameter defines the queue size for each of these queues.

Default Allocation Script

When a default allocation script is created, it is initialized to use the logic of the default allocation algorithm. There are 4 different methods that can be customized:

- Initialize
- Allocate Non-Belt Instances
- Allocate Belt Instances
- Notification as Instances are Processed

Default Allocation Script: Initialize Method

The Initialization method is called when the process manager is started. Each robot that is associated with the script will used to call the initialize method:

```
/// <summary>
/// Initializes the allocation algorithm at the start of runtime processing.
/// </summary>
/// <param name="station">The station.</param>
public void Initialize(RobotStation station) {
    Trace.WriteLine("Initializing for: "+station.Robot.FullPath);
}
```

Default Allocation Script: AllocateNonBeltInstances

The data requirements and allocation logic for belt based instances is very different than for non-belt based instances. As such, each class of instances is handled by a separate method. The non-belt instance method is called to allocate static and non-belt camera instances. The default behavior is to allocate the instances to the robot as long as there is space in the queues.

The non-belt instances method is passed two different arrays of note:

- ***availableInstances***: Input parameter containing the list of all instances that could be allocated.
- ***instancesToAllocate***: Output parameter containing the instances that should be sent to the robot controller.

It is the job of the ***AllocateNonBeltInstances*** to move instances from the ***availableInstances*** list to the ***instancesToAllocate*** list.

```
        /// <summary>
        /// Allocates the non-belt relative instances to a robot station.
        /// </summary>
        /// <param name="station">The station.</param>
        /// <param name="processType">The type of object being allocated.</param>
        /// <param name="queueSize">The queue size of the robot station.</param>
        /// <param name="availableInstances">The available instances.</param>
        /// <param name="instancesToAllocate">The instances to allocate to the station.</param>
        public void AllocateNonBeltInstances(RobotStation station, IProcessType processType, int queueSize, Locat

            // If no instances, do not process
            if (availableInstances.Count == 0)
                return;

            // Get the number of items in the robot queue for that type
            int numberInQueue = station.GetInstancesInProcess(processType);

            // Add as many items as we can to the queue
            foreach (LocatedInstance instance in availableInstances) {

                // Make sure there is room to add
                if (numberInQueue >= queueSize)
                    break;

                // Allocate the part to the station
                instancesToAllocate.Add(instance);

                // Note that one more has been added
                numberInQueue++;

            }
        }
```

Default Allocation Script: AllocateBeltInstances

The default behavior for allocation of belt-based instances is similar to non-belt instance handling from the standpoint that instances are allocated to a robot as long as the queue can take more instances.

The belt instances allocation method is passed three different arrays of note:

- ***availableInstances***: Input parameter containing the list of all instances that could be allocated.
- ***instancesToAllocate***: Output parameter containing the instances that should be sent to the robot controller.

- ***instancesToAllocate***: Output parameter containing the instances that should be sent to the next down-stream robot capable of processing the instance.

It is the job of the ***AllocateBeltInstances*** to move instances from the ***availableInstances*** either the ***instancesToAllocate*** list or the ***instancesToSkip*** list. The ***availableInstances*** list is an ordered list where the first instance in the list is the farthest downstream instance that is still capable of being processed by the robot station.

If an instance is never allocated by the ***AllocateBeltInstances*** method, the instance will automatically be moved to the queue of the next downstream robot once the instance passes out of the robot belt window.

```
        /// <summary>
        /// Allocates the belt-relative instances to a robot station.
        /// </summary>
        /// <param name="station">The station.</param>
        /// <param name="queueSize">Size of the queue.</param>
        /// <param name="availableInstances">The available instances.</param>
        /// <param name="instancesToAllocate">The instances to allocate.</param>
        /// <param name="instancesToSkip">The instances to skip and send to the next downstream robot.</param>
        public void AllocateBeltInstances(RobotStation station, int queueSize, LocatedInstanceCollection availabl

            // If no instances to process, skip
            if (availableInstances.Count == 0)
                return;

            // Assign parts to the robot
            int count;
            Dictionary<string, int> allocatedCount = new Dictionary<string, int>();

            foreach (LocatedInstance locatedInstance in availableInstances) {

                if (allocatedCount.ContainsKey(locatedInstance.ObjectType) == false) {
                    count = station.GetInstancesInProcess(ace[locatedInstance.ObjectType] as IProcessType);
                    allocatedCount.Add(locatedInstance.ObjectType, count);
                }

                // Get the number of parts in or allocated to the station
                count = allocatedCount[locatedInstance.ObjectType];

                // If there are too many parts in the queue
                if (count >= queueSize)
                    continue;

                instancesToAllocate.Add(locatedInstance);
                allocatedCount[locatedInstance.ObjectType]++;

            }
        }
```

Default Allocation Script: NotifyInstancesProcessed

After an instance is sent to a ***RobotStation***, the robot will either respond indicating that the instance was processed sucessfully or the instance was not processed. For example, if a belt-based instance was picked by the robot, it would respond to the PC that the instance was sucessfully processed. However, there are times where the robot was not able to pick all the instances on the conveyor because there are too many given the belt speed. In this case, the instances are robot responds to the PC that those instances were not processed.

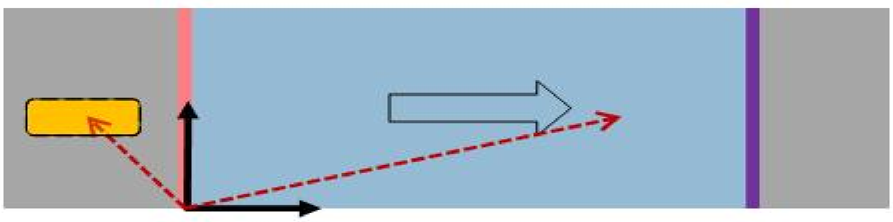
As robots report which instances are processed and not processed, the ***NotifyInstanceProcessed*** method is called:

```
/// <summary>
/// Method called notifying that instances sent to a RobotStation have been processed.
/// </summary>
/// <param name="station">The station.</param>
/// <param name="instancesProcessed">The instances sucessfully processed.</param>
/// <param name="instancesNotProcessed">The instances that were not processed by the station.</param>
public void NotifyInstancesProcessed(RobotStation station, LocatedInstanceCollection instancesProcessed,

}
```

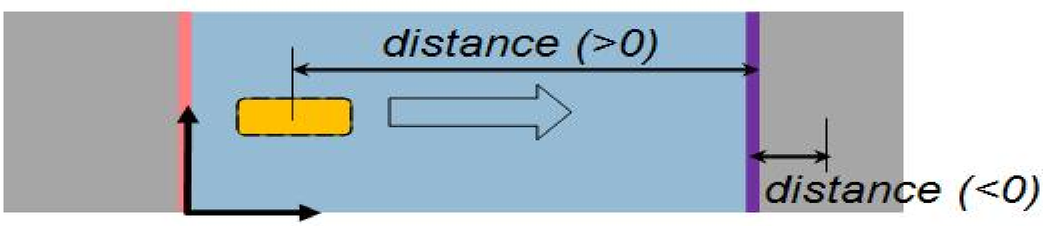
Default Allocation Script: LocatedInstance

The ***LocatedInstance*** class represents an instance that can be processed by a robot. There are many properties which can be useful when defining if an instance should be processed:

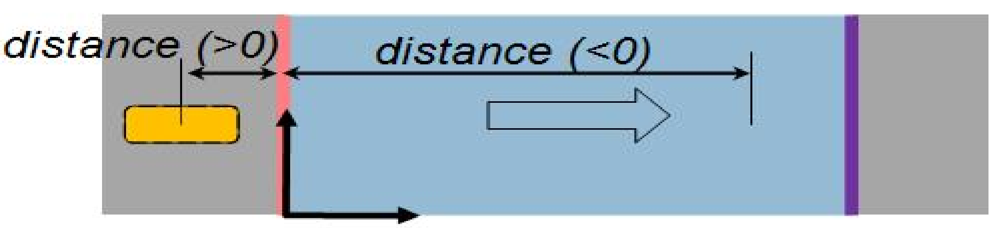
- BeltRelativePosition: Gives the position of the instance relatively to the belt frame and varies with time



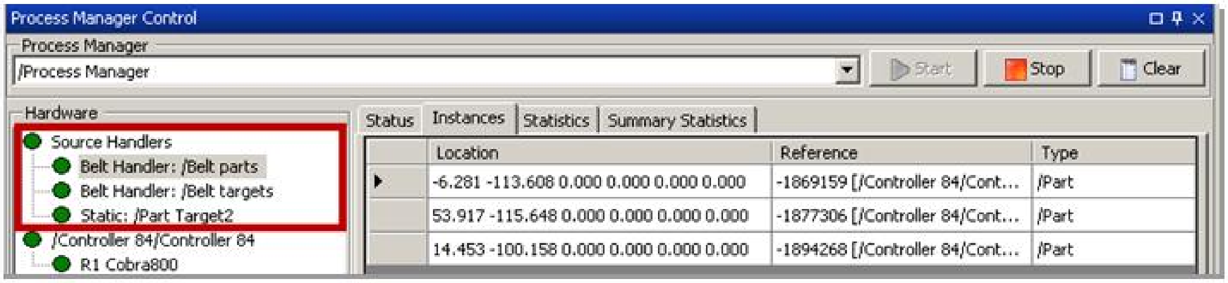
- DistanceToPickLine: Gives the distance between the instance and the pick line limit and varies with time



- DistanceToUpstreamWindow: Gives the distance the upstream line and the instance and varies with time



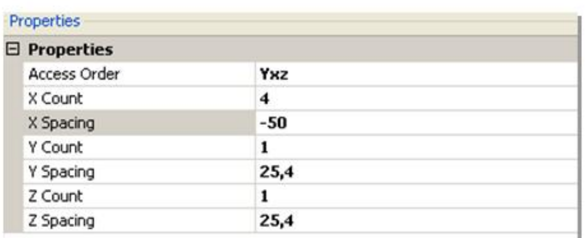
- Handler: Gives the source handler that generated the instance

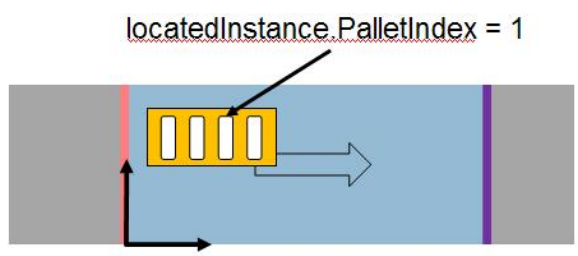


- Location: Location of the instance in its local frame
- ObjectType: Gives the "name" of the object type as a string

```
Part myPart = (Part) ace["/process1/Part"];
PartTarget myTarget = (PartTarget) ace["/process1/Part Target"];

if (locatedInstance.ObjectType == myPart.FullPath)
    ace.AppendToLog("this is my part");

if (locatedInstance.ObjectType == myTarget.FullPath)
    ace.AppendToLog("this is my target");
```

- PalletIndex: used only if the instance is pallet relative and defines the index of the pallet slot.





- PalletInstance: used only if the instance is pallet relative. unique identifier (Guid) of the pallet associated with this instance

There are also several methods that can be used in the process of allocating instances:

- GetControllerLatchPosition: Get the latched value of the instance (reference)

Default Allocation Script: LocatedInstance and the Tag property

There is one additional property that can be very useful when synchronizing a custom allocation script with a C# custom vision tool and/or robot behavior in V+. When a custom vision tool executes, a ***VisionTransform*** collection is returned as an output of the ***Main*** method. The ***VisionTransform*** class has a **Tag** property which is of type **object**. You can use this property to tag a vision instance in the custom vision tool and retrieve this tag from an allocations script.

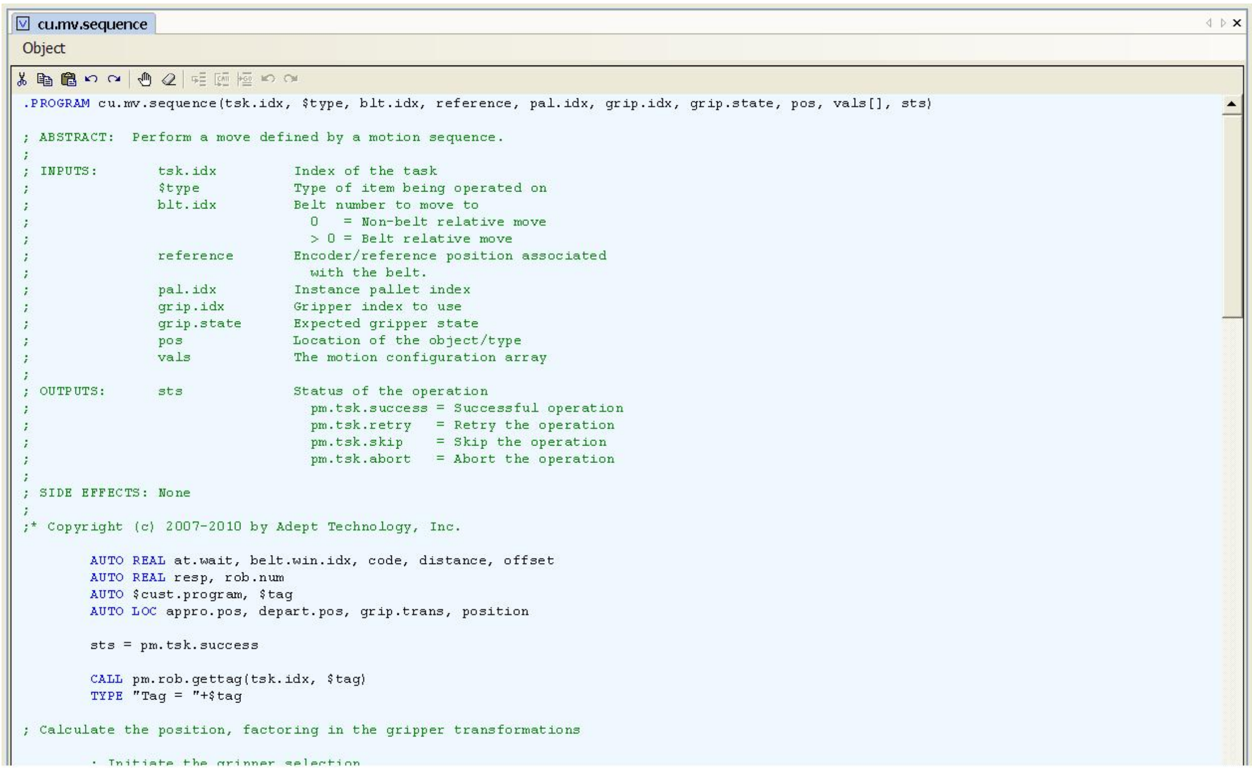
For example, your custom vision tool might look like this:

```
foreach (BlobResult res in myBlob.Results){

    // declare a List used to tag the instance
   List<float> myTag = new List<float>();

   // fill that List
   myTag.Add(res.Area);
   myTag.Add(res.GreyLevelMean);

   // tag the vision result
   res.Position.Tag = myTag;

   // add the vision result in Results of the custom vision tool
   results.Add(res.Position);
}
```

From within the allocation script, you can access the information as such:

```
List<float> tagInstance = (List<float>) locatedInstance.Tag;
float instanceArea = tagInstance[0];
float instanceMeanGrey = tagInstance[1];
```

Additionally, if the Tag is defined for a LocatedInstance, the Tag will be converted to a string representation and send to V+ as part of the instance information. From within a custom motion sequence program, the tag can be extracted as follows:




```
.PROGRAM cu.mv.sequence(tsk.idx, $type, blt.idx, reference, pal.idx, grip.idx, grip.state, pos, vals[], sts)

; ABSTRACT:  Perform a move defined by a motion sequence.
;
; INPUTS:       tsk.idx         Index of the task
;               $type           Type of item being operated on
;               blt.idx         Belt number to move to
;                                 0   = Non-belt relative move
;                                 > 0 = Belt relative move
;               reference       Encoder/reference position associated
;                                 with the belt.
;               pal.idx         Instance pallet index
;               grip.idx        Gripper index to use
;               grip.state      Expected gripper state
;               pos             Location of the object/type
;               vals            The motion configuration array
;
; OUTPUTS:      sts             Status of the operation
;                                 pm.tsk.success = Successful operation
;                                 pm.tsk.retry   = Retry the operation
;                                 pm.tsk.skip    = Skip the operation
;                                 pm.tsk.abort   = Abort the operation
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2010 by Adept Technology, Inc.

        AUTO REAL at.wait, belt.win.idx, code, distance, offset
        AUTO REAL resp, rob.num
        AUTO $cust.program, $tag
        AUTO LOC appro.pos, depart.pos, grip.trans, position

        sts = pm.tsk.success

        CALL pm.rob.gettag(tsk.idx, $tag)
        TYPE "Tag = "+$tag

; Calculate the position, factoring in the gripper transformations

        ; Initiate the gripper selection
```

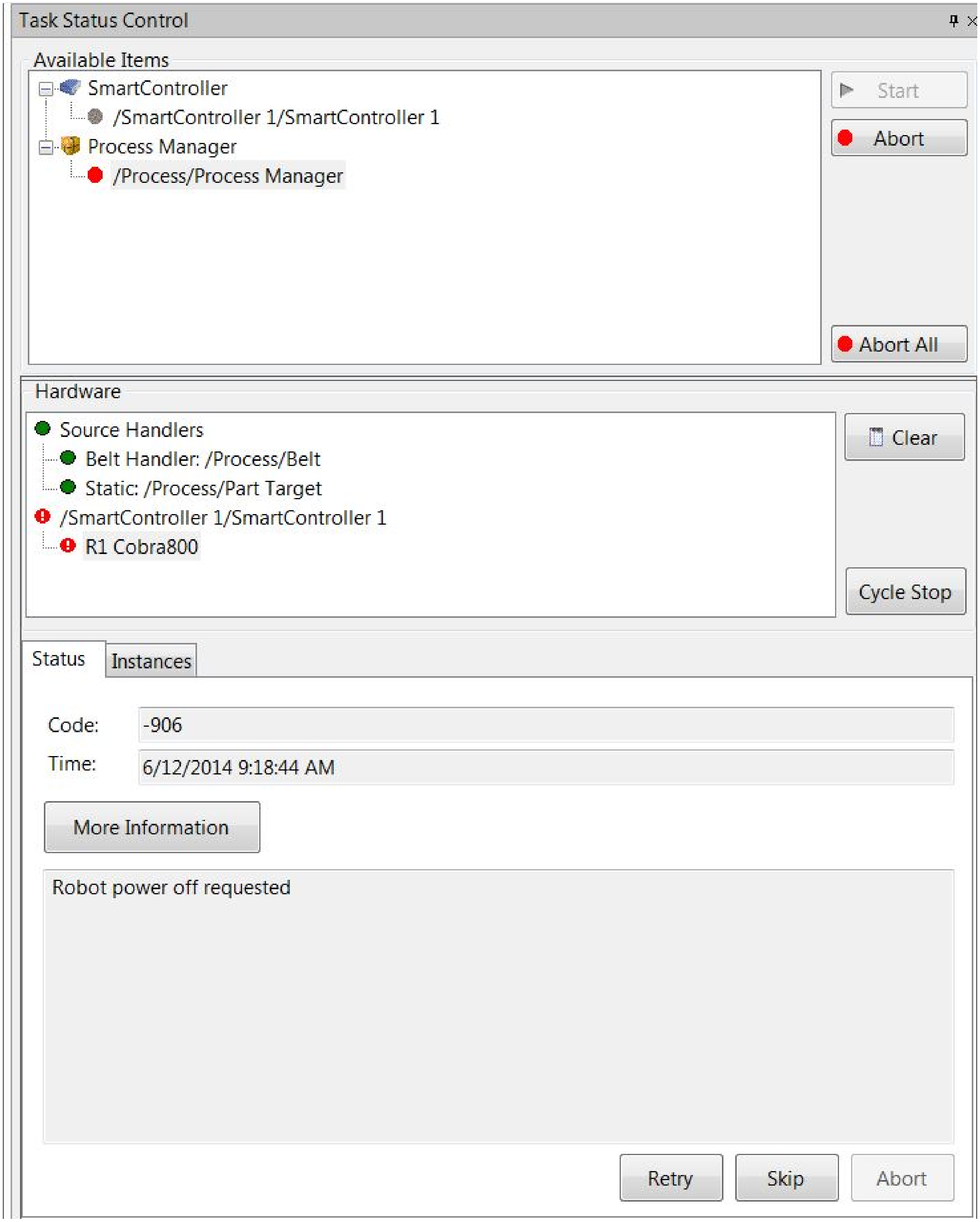
Custom Error Programs

The process manager runtime control will display errors that are reported during the execution of a process manager application. Many of these errors are generated from the V+ programs that control the robot and monitor other aspects of the hardware on the Adept controller.

For all errors generated on the controller, the user can customize how errors are handled and which errors are reported to the user interface.

Error Display

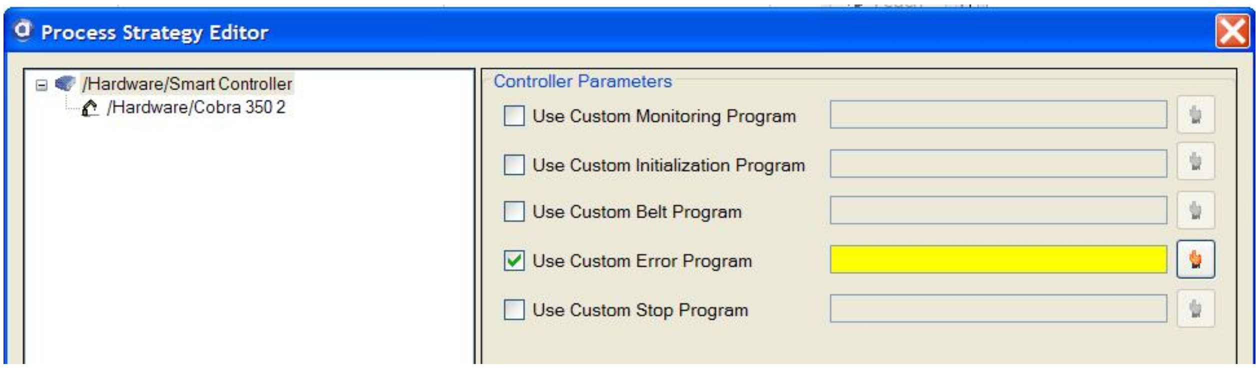
The process manager runtime display will show errors for all hardware sources, controllers, and robot stations:


Task Status Control
Available Items
SmartController
/SmartController 1/SmartController 1
Process Manager
/Process/Process Manager
Start
Abort
Abort All
Hardware
Source Handlers
Belt Handler: /Process/Belt
Static: /Process/Part Target
/SmartController 1/SmartController 1
R1 Cobra800
Clear
Cycle Stop
Status
Instances
Code:
-906
Time:
6/12/2014 9:18:44 AM
More Information
Robot power off requested
Retry
Skip
Abort

V+ Error Customization

When an error is generated on a controller, a single program on the robot is called to handle the error. This program makes a determination if an error is to be send to the PC to display or if the error should be automatically handled without user notification.

This program can be customized on a controller by controller basis:



When the program is customized, the user will start with a copy of the default error handling program. Here is a copy of the default program:

pm.psp.error

Object

```
.PROGRAM pm.psp.error(tsk.idx, code, code2, resp.mask, $stack[], resp)

; ABSTRACT:  Report errors to the PC. This program can filter
;       errors and change the reported code or handle errors
;       without reporting to the user interface.
;
; INPUTS:       tsk.idx         Task index for error reporting
;               code            Error code/status of the operation
;               code2           Variable portion of error code.
;               resp.mask       Available responses
;               $stack[]        V+ program stack information.  Indices
;                                 of 0 and 1 are valid.
;
; OUTPUTS:      resp            Possible error responses:
;                                 pm.tsk.success = The operation has
;                                     completed successfully.
;                                 pm.tsk.skip = Stop processing current
;                                     instance and move to next operation.
;                                 pm.tsk.retry = Retry processing of the
;                                     current instance.
;                                 pm.tsk.abort = Stop processing the
;                                     current process and clear the gripper.
;                                 pm.tsk.next = Move to the next instance
;                                     in the queue.
;                                 pm.tsk.fail = The operation has
;                                     not completed due to an error.
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2011 by Adept Technology, Inc.

        AUTO REAL is.grip.error

; If a belt window violation occurs, return with the response
; indicating the operation will be aborted.

        IF code == pm.err.blt.viol THEN
            resp = pm.tsk.abort

            ; To supress the movement to the idle position
            ; and clearing of the gripper, set the gripper
            ; clear state to FALSE by uncommenting the following
            ; line. When uncommented, the robot will stay at the
            ; current location and the gripper I/O will not be changed.
            ; CALL pm.rob.clear.st(tsk.idx, FALSE)

            GOTO 100
        END

; If an end effector error occurs, return with the response
; indicating the operation failed.

        is.grip.error = (code == pm.err.no.gr.cl) OR (code == pm.err.no.gr.op) OR (code == pm.err.inv.grip)
        is.grip.error = is.grip.error OR (code == pm.err.no.gr.ex) OR (code == pm.err.no.gr.rt)

        IF (is.grip.error == TRUE) THEN

        ; In the case of a gripper error, we report the pm.tsk.fail response.
        ; Part or target on which the error occured will be marked as failed.

            resp = pm.tsk.fail
            GOTO 100
        END


; Report the error

        CALL pm.error(tsk.idx, code, code2, pm.tsk.default, $stack[], resp)

   100  RETURN
.END
```

The error program will be called and passed the task number that has the error, the error code, and other details of the error. The program must pass back a status code the describes the response on how the error is to be handled in the ***resp*** parameter. The possible values are:

Response Code	Description
pm.tsk.success	The operation has completed successfully.
pm.tsk.skip	Stop processing current instance as if it was properly processed and move to the next operation.
pm.tsk.retry	Retry processing of the current instance.
pm.tsk.abort	Stop processing the current process and clear the gripper.
pm.tsk.next	Stop processing the current instance and move to the next instance in the queue.
pm.tsk.fail	The operation failed.

The default program will check to see if an error code indicates a belt window violation was detected. If this is true, the error program will automatically return a ***pm.tsk.abort*** response. This will cause the robot to abort the current process, clear the gripper, and look for a new process.

The default program will also check to see if an error code indicates an error from the gripper operation. If this is true, the error program will automatically return a ***pm.tsk.fail*** response. Doing so, the part or target that was being processed will be marked as failed causing the system to remove it from queues and not pushing it to the next robot on the conveyor : indeed we assume it may have moved when we tried to process it.

If the error is not a belt window violation, the default program will call ***pm.error,*** which will send the error to display on the process manager runtime control. When called, the error is presented to the user and any response will be returned.

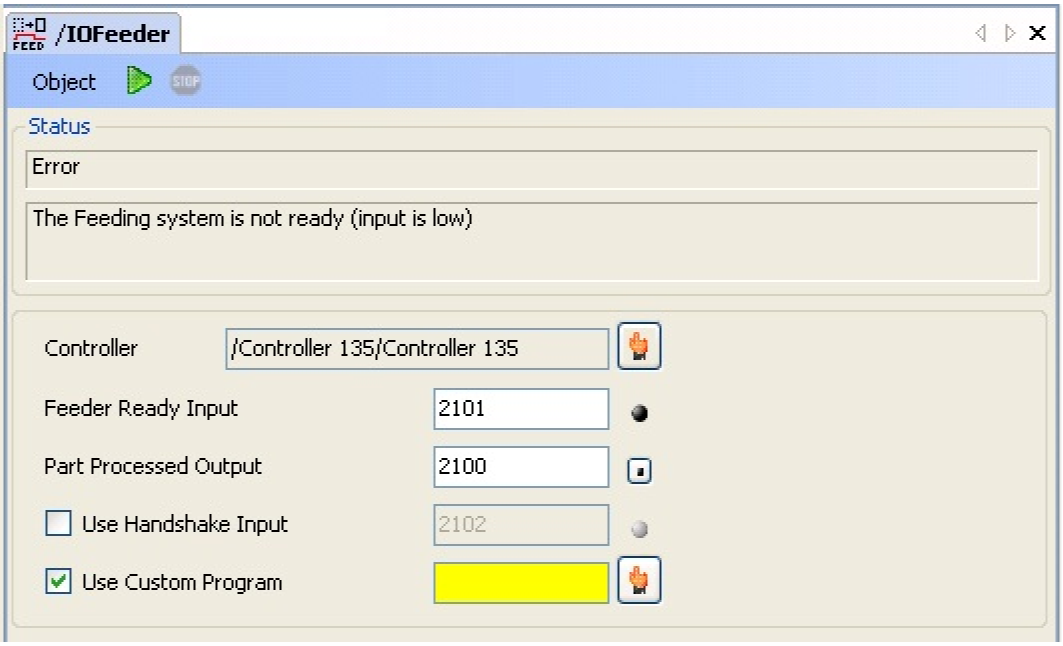
Some applications may want to display errors in a different way. For example, if a user wants to display an error using an IO panel with lights and monitor for button presses, the application should not call the ***pm.error*** program. Rather, it should turn on IO signals to indicate an error has been detected and then monitor for the press of a button through digital input points.

Custom IO Feeder Programs

When an IO Feeder is created it comes with a standard program managing a sequence of input and output signals and performing the feeding operation of the device. If the default sequence does not meet the requirements, the program that governs the behavior of the IO Feeder can be customized.

Customizing The Feeding Sequence

The IO Feeder editor defines all the parameters used when feeding parts or targets. The program managing the sequence can be overridden:



Use Custom Program checkbox allows replacing the standard program by user's one. When checked, the custom program selection box is available and user can select which program to use.

When the feeding program is customized and user wants to start from the standard one, the system starts by creating a copy of the default feeding sequence. The default feeding program allows 2 operations :

- Managing the input and output sequence performing the feeding operation
- Defining a set of instruction to be executed at initialization (called at IOFeeder start)

The definition of the default feeding program header looks like this:

```
.PROGRAM fdr.std.feed(fdr.idx, init.flag)

; ABSTRACT:  Executes a standard feed action.
;       This is the basic Feeding operation. Ready input is not read from here.
;       We consider it was checked before calling this method.
;       fdr.task.num[] array defined in this program allows user to indicate
;       one or several tasks usable to perform the feeding action.
;       - Number of tasks will set the number of IOFeeder that can run at the same time
;       - If this array is defined an available task is picked inside and used during feeding
;       - Task numbers are not related to a specific feeder. First available task is
;       picked and released once feeding action is complete
;       - If this array is undefined, a fdr.err.no.task error will occur
;       - If the array is defined but all tasks are used, a fdr.err.bsy.tsk error will occur
;       - This program can be used to initialize the tasks array using the init.flag input.
;       In this case, array is initialized and program exits.
;       - By default, tasks 2 to 20 can be selected
;
; INPUTS:         fdr.idx      Index of the device we want to feed with
;                 init.flag    if true, we will simply initialize fdr.task.num[] values and return
;
; OUTPUTS:        None
;
; SIDE EFFECTS:  This program is intended to be duplicated as user can customize it.
;       As a result there can be a different program for each iofeeder.
;       fdr.task.num[] values can be different for each feeder as we will use
;       this function each time in order to initialize the array.
;       If there are several iofeeders and several custom feed programs, as each program will
;       initialize fdr.task.num[] user will have to ensure there are no interference
;       between these initializations.
;
;* Copyright (c) 2010 by Adept Technology, Inc.
```

The feed sequence will be passed the IO Feeder number and the flag indicating the method is called to perform the initialization or to perform the feeding sequence.

The first step of the default program is to declare variables and to perform the initialization sequence when ***init.flag*** is true.

```
        AUTO REAL is.off, is.on, use.hdshke, debounce.sec, dwell.sec
        AUTO REAL sig.hdshke, sig.part.proc
        AUTO REAL i, sts

; Portion used to initialize the array containing the task numbers we can use.
; User can modify the array content and add more task numbers if necessary
; Tasks 0 and 1 reseved for user robot

        IF (init.flag) THEN
            FOR i = 0 TO 18
                fdr.task.num[fdr.idx,i] = i+2
            END
            RETURN
        END
```

During this step the variables are declared and the case ***init.flag*** is true is handled. This is called when the IO Feeder is started. A 2D array is used to indicate for each IO Feeder the Controller tasks it is allowed to use. Default will indicate tasks 2 to 20. At start, this array is scanned and the first available task is used to run the IO Feeder main program.

If user is running his own programs on dedicated tasks he can then ensure the IO Feeder will never try to use it. He can also fill it with only one value to force the IO Feeder running on a known task.

When called with the ***init.flag*** value to true, this will be the only operation performed by this program.

The rest of the program contains the Feeding sequence.

First step of feeding sequence is its initialization :

```
; Starting a command : say it to pc

        CALL fdr.pc.op.start(fdr.idx, sts)

        fdr.command[fdr.idx] = fdr.cmd.null

; Memorize IO settings (cannot change during the feed action then)

        debounce.sec = rm.obj.rvals[fdr.idx,fdr.dlay.dbnce]/1000
        dwell.sec = rm.obj.rvals[fdr.idx,fdr.dlay.dwell]/1000
        sig.hdshke = rm.obj.rvals[fdr.idx,fdr.in.hdshke]
        sig.part.proc = rm.obj.rvals[fdr.idx,fdr.out.proc]
        use.hdshke = rm.obj.rvals[fdr.idx,fdr.use.hdshke]

        fdr.status[fdr.idx] = fdr.sts.run

; If handshake used it has to be the correct state

        IF rm.obj.rvals[fdr.idx,fdr.use.hdshke] THEN

            CALL sv.sig.is.equal(rm.obj.rvals[fdr.idx,fdr.in.hdshke], FALSE, is.off)

            IF NOT is.off THEN
                fdr.status[fdr.idx] = fdr.err.hshke
                GOTO 100
            END
        END
```

The call to ***fdr.pc.op.start*** indicates the PC that the IO Feeder identified by ***fdr.idx*** started running. The ***sts*** value is simply the result of this instruction and is not used here.

fdr.command[fdr.idx] is then switched to a neutral value indicating the IO Feeder main task the requested operation is being performed.

Then all the parameters used to perform the operation are stored in local variables : if they are modified by user during the sequence it then has no impact on current operation. Feeder status is set to 'running'.

The last part of initialization is checking the handshake input is low in the case it is used. If not, feeding is not possible and an error will be returned.

Note : *Reading a signal value is done using **sv.sig.is.equal** program. This program reads the input and tells if it is at the expected state. Any customization should use this program to read inputs as it manages the case we are running on an emulator. Signals can be read using the SIG keyword but in this case it may not work properly with the emulator.*

Once the initialization and checking are performed, the feeding can start.

```
; Indicate part has been processed and dwell for the specified amount of time

        SIGNAL (sig.part.proc)
        IF (dwell.sec > 0) THEN
            CALL fdr.dwell(fdr.idx, dwell.sec)
        END

; If handshake input specified, wait for it to go on.

        IF (use.hdshke AND (fdr.command[fdr.idx] <> fdr.cmd.abort)) THEN

        ; Init signal management data  and Wait signal is stable long enough

            CALL fdr.sig.init(sig.hdshke)
            DO
                CALL fdr.sig.wait(sig.hdshke, debounce.sec, is.on)
                WAIT
            UNTIL (is.on OR (fdr.command[fdr.idx] == fdr.cmd.abort))
        END
```

the ***sig.part.proc*** output is switched on and activates the IO Feeder. Once it is activated we are waiting for the specified dwell time before continuing.

If an handshake signal is to be used the program is waiting for it to be on using the ***debounce.sec*** value to check it remains stable long enough.

After these operations are completed, the feeding is done and we have to terminate the sequence:

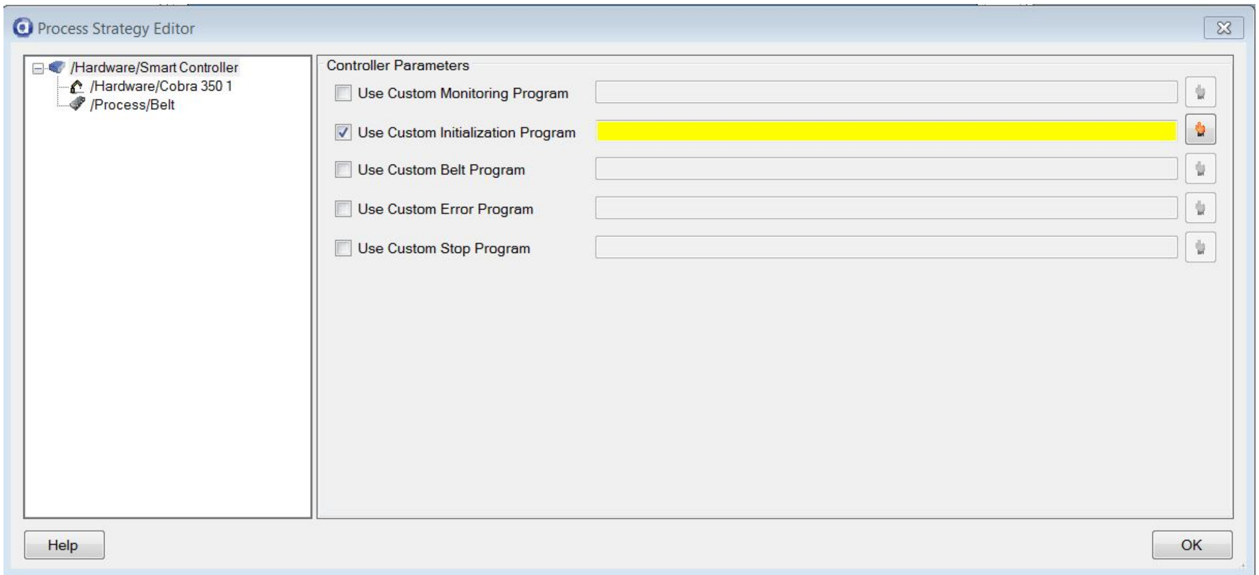
```
; Turn off the processed output

        SIGNAL (-sig.part.proc)

; If handshake input specified, wait for it to go off.

        IF (use.hdshke AND (fdr.command[fdr.idx] <> fdr.cmd.abort)) THEN

        ; Init signal management data and wait signal is stable long enough

            CALL fdr.sig.init(sig.hdshke)
            DO
                CALL fdr.sig.wait(-sig.hdshke, debounce.sec, is.off)
                WAIT
            UNTIL (is.off OR (fdr.command[fdr.idx] == fdr.cmd.abort))
        END
        fdr.status[fdr.idx] = fdr.sts.avail

   100
    ; Feeding is done. Send information to the PC

        CALL fdr.pc.op.end(fdr.idx, sts)

        RETURN
.END
```

The processed output is switched off and program waits for handshake signal to go off.

fdr.status[fdr.idx] is set to 'available' and ***fdr.pc.op.end*** indicates to the PC that the IO Feeder operation is finished. This instruction will report the operation result stored in ***fdr.status[fdr.idx]*** to the PC : this can be a success value or an error code that will be managed on PC side.

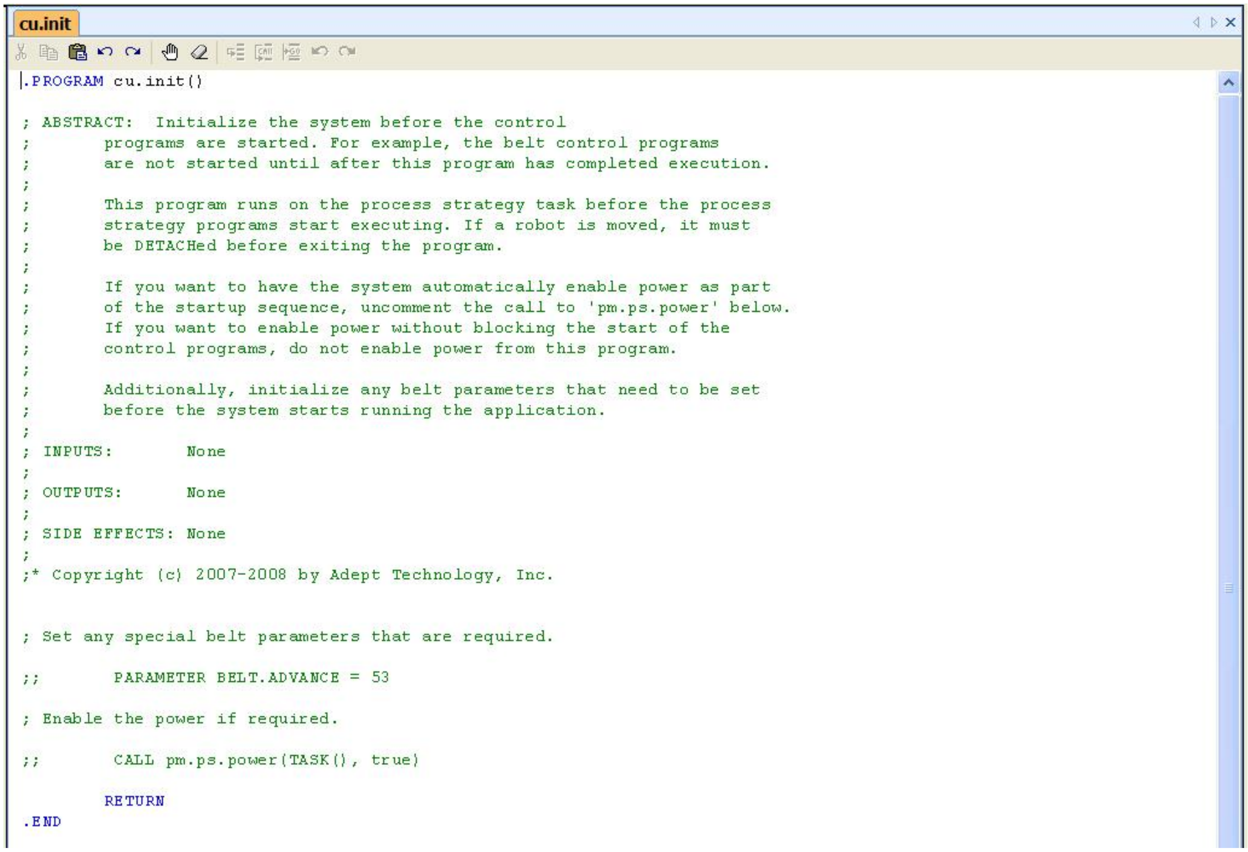
Customizing System Initialization

When the process manager establishes communications with a controller in runtime, it will go through a data initialization and startup phase. The user can define a V+ program that is called when the process manager starts the initialization process on a controller through the following menu:



Custom Initialization Program

The custom initialization program looks like this:



```
cu.init
.PROGRAM cu.init()

; ABSTRACT:  Initialize the system before the control
;       programs are started. For example, the belt control programs
;       are not started until after this program has completed execution.
;
;       This program runs on the process strategy task before the process
;       strategy programs start executing. If a robot is moved, it must
;       be DETACHed before exiting the program.
;
;       If you want to have the system automatically enable power as part
;       of the startup sequence, uncomment the call to 'pm.ps.power' below.
;       If you want to enable power without blocking the start of the
;       control programs, do not enable power from this program.
;
;       Additionally, initialize any belt parameters that need to be set
;       before the system starts running the application.
;
; INPUTS:       None
;
; OUTPUTS:      None
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2008 by Adept Technology, Inc.


; Set any special belt parameters that are required.

;;       PARAMETER BELT.ADVANCE = 53

; Enable the power if required.

;;       CALL pm.ps.power(TASK(), true)

        RETURN
.END
```

By default, the standard stop program does nothing. It can be used to initialize variables, parameters or tasks used by an application. For example, if you wanted to start a custom V+ program in a background task, this program could be used to execute the program task.

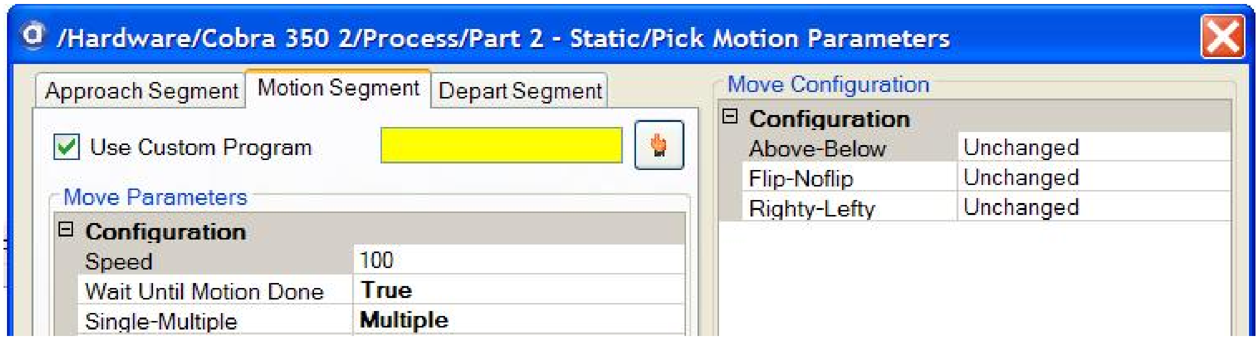
Custom Pick and Place Motion Programs

When a process is created, the process specifies that a robot must pick a set of parts and should place those parts at a set of targets. The process manager will supply a default set of motions used to acquire the parts and place at the targets. If the default motions do not meet the requirements, the program that governs the behavior of the motions can be customized.

In addition to defining parts and targets, a process may also reference a refinement station that locates parts in the gripper before placement at targets. The program that defines the motions of a refinement operation can also be overridden if the default motions are not sufficent.

Customizing The Motion Sequence

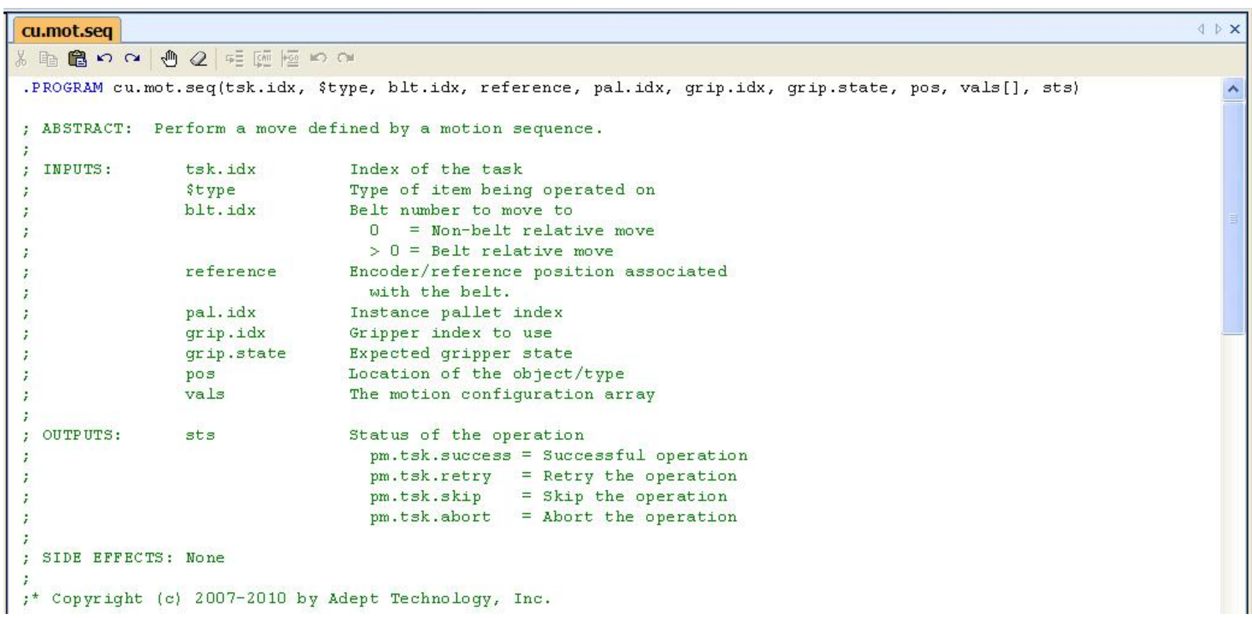
The motion sequence configuration item defines all the parameters used when moving to pick a part and place at a target. The motion sequence logic can be overridden:



When the motion sequence program is customized, the system starts by creating a copy of the default motion sequence. The default motion sequence can be broken down into several steps:

- Calculate Approach, Move, and Depart Positions
- Wait for belt based objects to track into the belt window
- Perform the Approach Sequence
- Perform Move to Destination
- Perform the Depart Sequence

The definition of the default motion program header looks like this:



cu.mot.seq

```
.PROGRAM cu.mot.seq(tsk.idx, $type, blt.idx, reference, pal.idx, grip.idx, grip.state, pos, vals[], sts)

; ABSTRACT:  Perform a move defined by a motion sequence.
;
; INPUTS:       tsk.idx         Index of the task
;               $type           Type of item being operated on
;               blt.idx         Belt number to move to
;                                 0   = Non-belt relative move
;                                 > 0 = Belt relative move
;               reference       Encoder/reference position associated
;                                 with the belt.
;               pal.idx         Instance pallet index
;               grip.idx        Gripper index to use
;               grip.state      Expected gripper state
;               pos             Location of the object/type
;               vals            The motion configuration array
;
; OUTPUTS:      sts             Status of the operation
;                                 pm.tsk.success = Successful operation
;                                 pm.tsk.retry   = Retry the operation
;                                 pm.tsk.skip    = Skip the operation
;                                 pm.tsk.abort   = Abort the operation
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2010 by Adept Technology, Inc.
```

The motion sequence will be passed the task umber and the name of the type of product being processed in the ***$type*** variable. The ***$type*** variable can be used to determine what part or target is being called. For example, if your process manager references a target named "/Process/Target", the following code is valid:

IF $target == "/Process/Target" THEN

The first step of the motion sequence program is to calculate the approach, move, and depart positions:

```
; Calculate the position, factoring in the gripper transformations

        ; Initiate the gripper selection

        CALL pm.gr.select(tsk.idx, grip.idx)

        ; Get the gripper transformation

        CALL pm.gr.trans(tsk.idx, grip.idx, grip.trans, sts)

        ; The position must be offset by the gripper transform
        ; do not use the TOOL instruction because that causes a stop
        ; in motion when applied

        SET position = pos:INVERSE(grip.trans)

        ; Calculate the approach position

        IF (vals[pm.fmv.ahtabs] == 0) THEN
            SET appro.pos = position:TRANS(0,0,-vals[pm.fmv.aheight])
        ELSE
            SET appro.pos = position:TRANS(0,0,-(vals[pm.fmv.aheight]-DZ(position)))
        END

        ; Calculate the depart position

        IF (vals[pm.fmv.dhtabs] == 0) THEN
            SET depart.pos = position:TRANS(0,0,-vals[pm.fmv.dheight])
        ELSE
            SET depart.pos = position:TRANS(0,0,-(vals[pm.fmv.dheight]-DZ(position)))
        END
```

The default program will account for the transform associated with the gripper tip by applying the INVERSE of the offset to the position so the robot will not stop when a TOOL command is executed.

After the positions are define, the software will wait for any belt based objects to track into the belt window:

```
; If a belt part, move to approach
; and wait until part tracks into the window

        IF (blt.idx <> 0) THEN

            at.wait = FALSE

            rob.num = SELECT(ROBOT)
            belt.win.idx = pm.rob.belt.idx[rob.num,blt.idx]
            SETBELT %pm.belt[rob.num,blt.idx,belt.win.idx] = reference
            distance = WINDOW(%pm.belt[rob.num,blt.idx,belt.win.idx]:position,0,1)
            WHILE (distance < 0) DO

                ; Move to the waiting position

                IF (at.wait == FALSE) THEN
                    offset = pm.dynamic.off[rob.num,blt.idx]
                    CALL pm.mv.wait(tsk.idx, blt.idx, offset, position, vals[])
                    at.wait = TRUE
                END

                ; Since the instance is not in-range, monitor the queues to make sure
                ; other instances that go out of range are properly deallocated

                CALL pm.rob.monque(tsk.idx, -1, -1)

                ; Check for task errors

                CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, code, resp)
                IF (resp <> pm.tsk.success) THEN
                    CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[TASK()])
                    sts = resp
                    GOTO 100
                END

                distance = WINDOW(%pm.belt[rob.num,blt.idx,belt.win.idx]:position,0,1)
                WAIT
            END

            ; If the part is beyond the belt window, report the error

            IF (distance > 0) THEN
                CALL pm.ps.error(tsk.idx, pm.err.blt.viol, pm.tsk.default, resp)
                sts = resp
                GOTO 100
            END
        END
```

The call to ***pm.rob.monque*** is very important. That program will check the part and target queues associated with the robot. If any parts or targets can no longer be processed, the instances are marked so they can be returned to the PC for processing for a downstream robot.

Once the instance can be accessed by the robot, the software will then process the approach sequence:

```
; Approach the point

        CALL pm.mv(blt.idx, appro.pos, pm.ms.aoffset, vals[])
        CALL pm.mv.chkbrk(pm.ms.aoffset, vals[])

        ; Check for errors

        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, code, resp)
        IF (resp <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[TASK()])
            sts = resp
            GOTO 100
        END

; Check the gripper selection

        CALL pm.gr.select.ck(tsk.idx, grip.idx, resp)
        IF (resp <> pm.tsk.success) THEN
            sts = resp
            GOTO 100
        END
```

The approach sequence is relatively straightforward. The motion primitive ***pm.mv*** will issue the move instruction to the robot and ***pm.mv.chkbrk*** will check for any BREAK settings on the approach sequence. Lastly, the program will make sure the correct gripper tip is selected.

After the approach is processed, the software will perform the move to the destination:

```
; Perform the motion

        ; Issue the motion

        CALL pm.mv(blt.idx, position, pm.ms.poffset, vals[])

        ; Monitor motion for gripper actuation

        CALL pm.mv.mongrip(tsk.idx, TRUE, appro.pos, position, grip.idx, grip.state, vals[], code, resp)
        IF (resp <> pm.tsk.success) THEN
            sts = resp
            GOTO 100
        END

        ; Check for break conditions

        CALL pm.mv.chkbrk(pm.ms.poffset, vals[])

        ; Check for errors

        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, code, resp)
        IF (resp <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[TASK()])
            sts = resp
            GOTO 100
        END

; Wait for gripper to complete

        DO
            CALL pm.gr.state(tsk.idx, grip.idx, grip.state, TRUE, sts)
            IF (sts < 0) THEN
                CALL pm.ps.error(tsk.idx, sts, pm.tsk.default, resp)
                IF (resp <> pm.tsk.success) THEN
                    sts = resp
                    GOTO 100
                END
            END
        UNTIL (sts >= 0)
```

The move to the destination is very similar to the approach and uses some of the same primitive functions. Once the call to ***pm.mv.chkbrk*** has completed, the software will ensure the gripper is in the correct state before continuing.

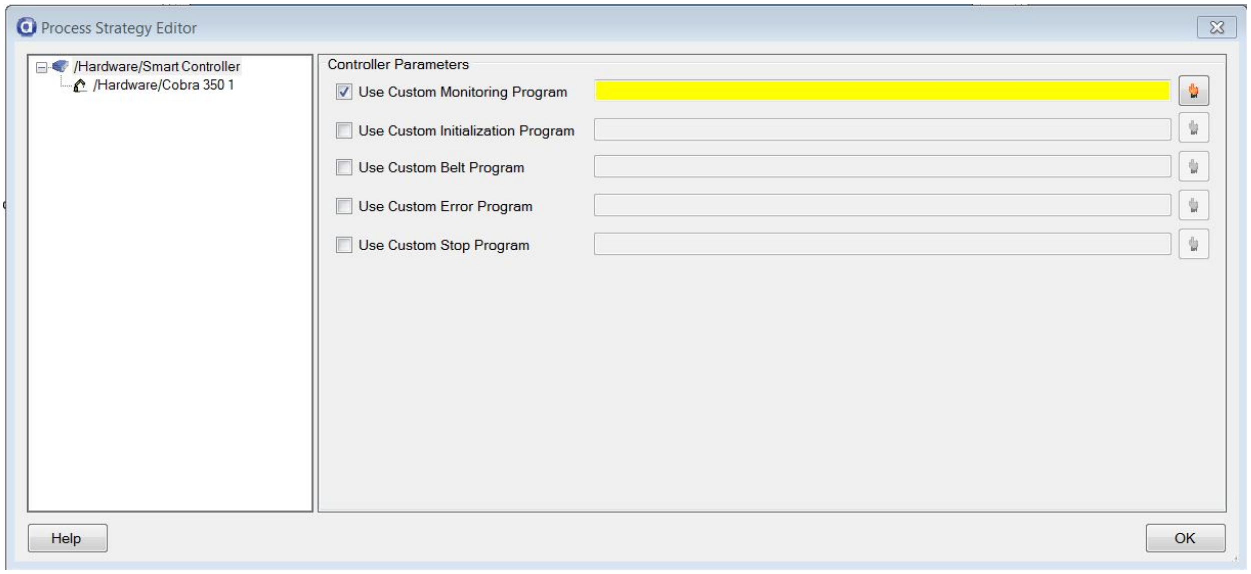
Once the move has been processed and the gripper is in the correct state, the software will perform the depart motion sequence:

```
; Perform the depart

   80       ; Start the motion

        CALL pm.mv(blt.idx, depart.pos, pm.ms.doffset, vals[])

        ; Monitor motion for gripper actuation

        CALL pm.mv.mongrip(tsk.idx, FALSE, position, depart.pos, grip.idx, grip.state, vals[], code, resp)
        IF (resp <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[TASK()])
            sts = resp
            GOTO 100
        END

        ; Check for any breaks

        CALL pm.mv.chkbrk(pm.ms.doffset, vals[])

        ; Check for errors

        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, code, resp)
        IF (resp <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[TASK()])
            sts = resp
            GOTO 100
        END

  100   RETURN
```

The depart sequence is very similar to the approach sequence.

Custom Process Strategy Monitoring Programs

The process manager application software will run a background program on each controller that is responsible for managing part and target queues and triggering belt monitoring programs. It is highly unlikely one would need to customize this program, but it is available in the process strategy editor here:



The process strategy monitoring program is broken into two different sections:

- Variable Initialization
- Control Loop

Variable Initialization

The default program first initializes variables that are used to monitor initialization of data structures used by the task. These should not be modified:

```
.PROGRAM my.pstrategy()

; ABSTRACT:  The main program implementing the packaging process strategy.
;
; INPUTS:        None
;
; OUTPUTS:       None
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2009 by Adept Technology, Inc.

        AUTO REAL run, tsk.idx

; Initialize Variables

        ; Mark that the strategy has not been initialized
        ; Initialization done in first call to 'pm.psp.init'

        psp.blt.init = FALSE
        psp.rbw.init = FALSE
        tsk.idx = TASK()
```

Control Loop

After variable initialization has completed, the software will then enter a continuous loop as long as the process manager is running. The program simply calls other programs to ensure the servicing of the following:

- Process strategy data structures (pm.psp.init)
- Belt control monitoring (pm.psp.belt)
- Instance processing (pm.ps.monitor)

Great care should be taken when modifying this program. The servicing of these three functions is critical to the normal operation of the system. These programs affect how the robots process the instance queues.

```
; Control loop

        DO

            ; Check for updates to process strategies

            CALL pm.psp.init(tsk.idx)

            ; Handle belt monitoring

            CALL pm.psp.belt(tsk.idx, TRUE)

            ; Monitor parts and targets

            CALL pm.ps.monitor(tsk.idx)

            ; Check for server running

            CALL pm.chk.stat(run)

            WAIT.EVENT , 0.1

        UNTIL run == FALSE

        RETURN
.END
```

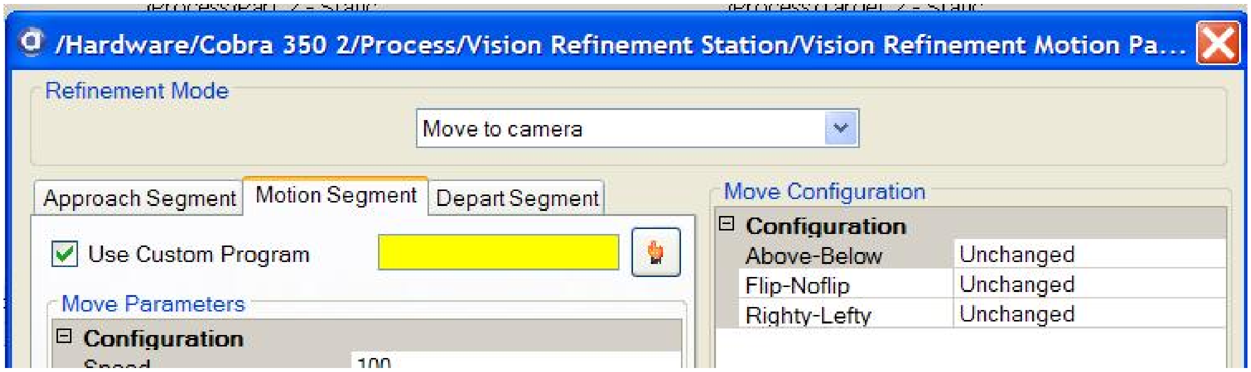
Custom Refinement Motion Programs

When a process is created, the user can define a refinement operation. When the refinement is specified, the instance will be located with a camera after the pick operation and before the placement operation.

The program that defines the motions of a refinement operation can also be overridden if the default motions are not sufficient.

Customizing The Refinement Sequence

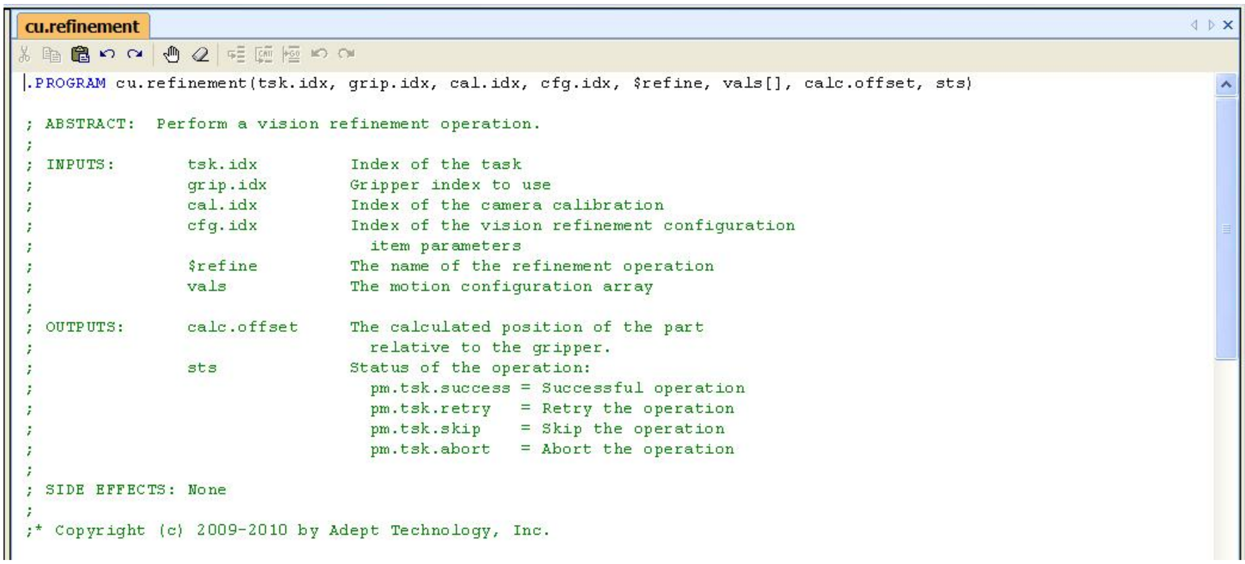
The refinement configuration item defines all the parameters used when refining parts in the gripper before placing at a target. The motion sequence logic can be overridden:



When the refinement program is customized, the system starts by creating a copy of the default refinement sequence. The default refinement sequence can be broken down into several steps:

- If normal camera refinement is enabled
  - Calculate the Approach, Move, and Depart Positions
  - Perform the Approach Sequence
  - Perform Move to Destination
  - Issue Camera Refinement Operation
  - Perform the Depart Sequence
- If Vision-on-the-Fly refinement is enabled
  - Calculate the Start and End Positions
  - Move to the Start Position
  - Start Moving to the End Position
  - Trigger the Refinement Operation

The definition of the default refinement program header looks like this:

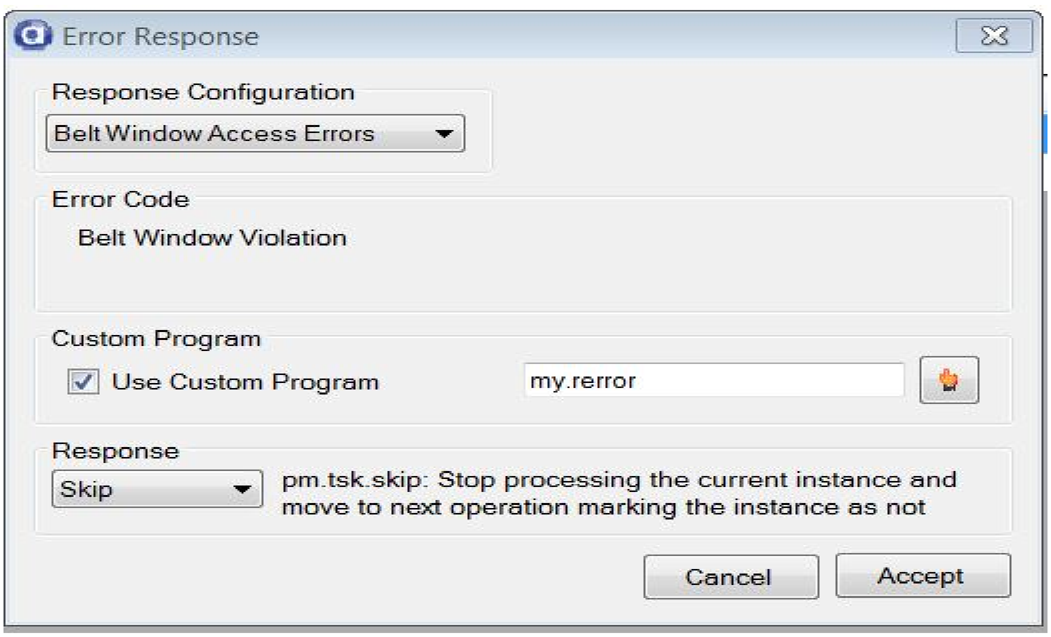


```
.PROGRAM cu.refinement(tsk.idx, grip.idx, cal.idx, cfg.idx, $refine, vals[], calc.offset, sts)

; ABSTRACT:  Perform a vision refinement operation.
;
; INPUTS:       tsk.idx         Index of the task
;               grip.idx        Gripper index to use
;               cal.idx         Index of the camera calibration
;               cfg.idx         Index of the vision refinement configuration
;                                 item parameters
;               $refine         The name of the refinement operation
;               vals            The motion configuration array
;
; OUTPUTS:      calc.offset     The calculated position of the part
;                                 relative to the gripper.
;               sts             Status of the operation:
;                                 pm.tsk.success = Successful operation
;                                 pm.tsk.retry   = Retry the operation
;                                 pm.tsk.skip    = Skip the operation
;                                 pm.tsk.abort   = Abort the operation
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2009-2010 by Adept Technology, Inc.
```

Many of the primitives are the same as what was seen in the default motion sequence program, so the program will not be reviewed with the same level of detail.

Custom Robot Error Response Programs

The process manager allows you to specify how a robot handles a specific error or class of error. If custom operations need to happen when the error is being handled, the user can specify a custom V+ program that will be called:



Customizing The Error Response Program

The default error response program does not perform any operations:

```
.PROGRAM my.error(tsk.idx, code, resp)

; ABSTRACT:  Custom error program called when a specific error
;        condition occurs.
;
; INPUTS:        tsk.idx         Task index for error reporting
;                code            Error code/status of the operation
;
; OUTPUTS:       resp            Possible error responses:
;                                  pm.tsk.success = The operation has
;                                      completed successfully.
;                                  pm.tsk.skip = Stop processing current
;                                      instance and move to next operation.
;                                  pm.tsk.retry = Retry processing of the
;                                      current instance.
;                                  pm.tsk.abort = Stop processing the
;                                      current process and clear the gripper.
;                                  pm.tsk.next = Move to the next instance
;                                      in the queue.
;                                  pm.tsk.fail = The operation has
;                                      not completed due to an error.
;                                  pm.tsk.cancel = Cancel processing of instance
;                                      and stop processing the current process.
;
; SIDE EFFECTS: None
;
; REMARKS:  To supress the movement to the idle position and
;        clearing of the gripper, set the gripper
;        clear state to FALSE by CALLing the following program.
;        When called, the robot will stay at the current location
;        and the gripper I/O will not be changed.
;
;               CALL pm.rob.clear.st(tsk.idx, FALSE)
;
;* Copyright (c) 2014 by Adept Technology, Inc.

        RETURN
.END
```

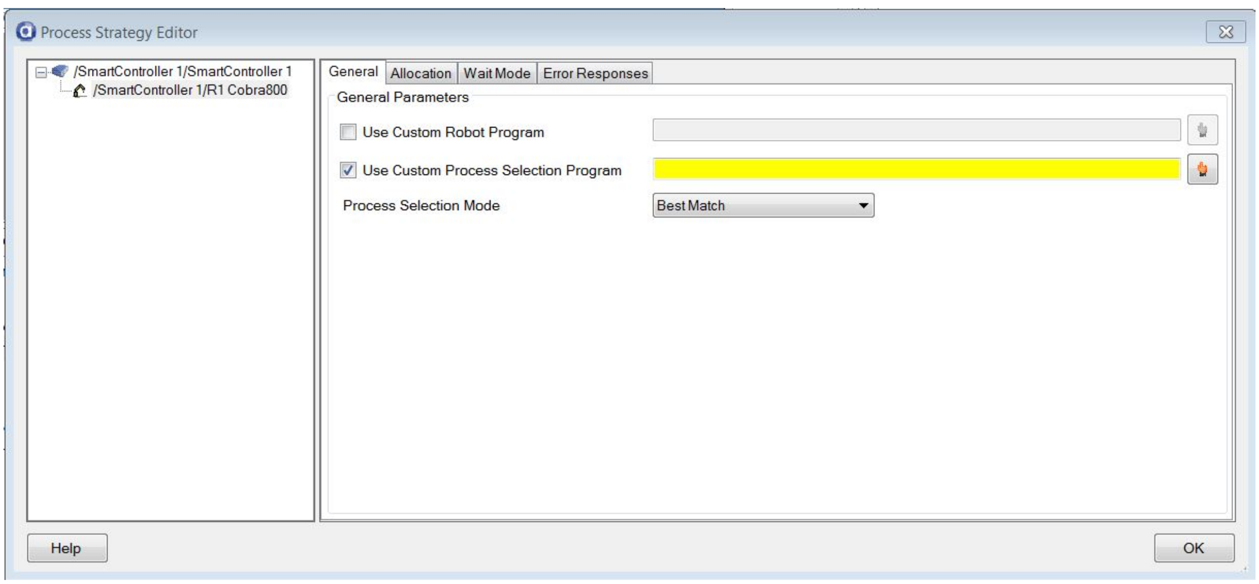
This program should be modified to implement any custom operations that are required. Typically, this might involve setting application specific variables or triggering other custom operations.

As an example, here is custom code that shows a typical customization that might take place. When the error program is called, it will use digital I/O to communicate with an external PLC. After continuing, it checks the system power state:

```


; Example: Set a signal and wait for confirmation

        SIGNAL o.error
        WHILE NOT SIG(i.error) DO
            WAIT
        END
        SIGNAL -o.error

; If power is not enabled, enable it and delay for it to take effect. Override the
; default error response in that case to explicitly issue a skip command.

        IF NOT SWITCH(POWER) THEN
            ENABLE POWER
            WAIT.EVENT , 1
            resp = pm.tsk.skip
        END

        RETURN
.END

```

Custom Robot Process Selection Program

Some applications may only need to customize the process that is selected for a given robot without customizing the entire robot program. If this is the case, the process selection can be customized here:



The default process selection program looks like this:

cu.pselect

```
.PROGRAM cu.pselect(rob.tsk, proc.idx)

; ABSTRACT:  Gets the process index for the robot to operate on.
;
; INPUTS:       rob.tsk         Current robot task
;
; OUTPUTS:      proc.idx        Desired process index
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2010 by Adept Technology, Inc.

        AUTO REAL first.ena.idx, num.enabled, ok
        AUTO REAL prt.pct.d, prt.pct.u, tgt.pct.d, tgt.pct.u
        AUTO REAL sel.prt.pct, sel.tgt.pct, temp.idx

        proc.idx = pm.ps.idx.none
        first.ena.idx = pm.ps.idx.none
        num.enabled = 0

; Handle the special case of cycle stop

        IF (pm.tsk.cycl.stp[rob.tsk]) GOTO 100

; Handle the special case of only 1 process is available

        ; Get the index of the current list of process we can operate with

        temp.idx = pm.ps.rob.midx[rob.tsk,pm.ps.rob.crix]

        ; If there are no "next" processes, then it is the only
        ; process at this level.

        IF (pm.ps.rob.map[temp.idx,pm.ps.rob.nxt] == pm.ps.rob.nopx) THEN

            ; If the process is not enabled, then do not assign any process.

            IF NOT pm.ps[temp.idx,pm.ps.ena] GOTO 100

            ; If the process can only be selected if it is in range
            ; then defer the check to the full selection logic in the
            ; code selection below.

            IF NOT pm.ps[temp.idx,pm.ps.inrng] THEN
                proc.idx = temp.idx
                GOTO 100
            END
        END
```

The first section will check to see if only one process exists. If that is the case, the process is automatically selected, regardless of the availability of any parts or targets. If this is the case, no other logic is performed for process selection.

```
; Handle the general case of multiple processes that could be used.
; We should find the set of process in the current set that
; has all parts and targets available. From within that set, pick
; the process based on the position of the product relative to
; the belt window (assuming a belt driven product).
; If only one process is enabled, then always select that process.

        DO

            ; Make sure the process is enabled

            IF NOT pm.ps[temp.idx,pm.ps.ena] GOTO 50

            ; Keep track of how many processes are enabled and
            ; the first enabled process.

            num.enabled = num.enabled+1
            IF (first.ena.idx == pm.ps.idx.none) THEN
                first.ena.idx = temp.idx
            END

            ; See if all required parts are available

            IF pm.ps[temp.idx,pm.ps.pk.excl] THEN
                CALL pm.ps.chk.partx(rob.tsk, temp.idx, prt.pct.d, prt.pct.u, ok)
            ELSE
                CALL pm.ps.chk.part(rob.tsk, temp.idx, prt.pct.d, prt.pct.u, ok)
            END

            ; If so, see if all required targets are available
```

If multiple process are define, the program will go through all of the processes and check to see if all parts for a given process are available in the queue by calling the ***pm.ps.chk.part*** and ***pm.ps.chk.partx*** programs. Those programs will scan the queues and ensure the total number of instances required by the process are available and return the farthest upstream and downstream position of the required instances.

```
            ; If so, see if all required targets are available

            IF ok THEN

                ; If in-range checking is enabled, see if the position
                ; is out of range. If so, skip the selection.

                IF pm.ps[temp.idx,pm.ps.inrng] AND ((prt.pct.u < 0) OR (prt.pct.d >= 100)) GOTO 50

                ; See if targets exist. If so, record the relative
                ; position of the target and the priority of the process
                ; if this is the first process that matches.

                IF pm.ps[temp.idx,pm.ps.tr.excl] THEN
                    CALL pm.ps.chk.targx(rob.tsk, temp.idx, tgt.pct.d, tgt.pct.u, ok)
                ELSE
                    CALL pm.ps.chk.targ(rob.tsk, temp.idx, tgt.pct.d, tgt.pct.u, ok)
                END
```

If all parts are available, the program will next check to see if all targets for the given process are available in the queue by calling the ***pm.ps.chk.targ*** and ***pm.ps.chk.targx*** programs. Those programs will scan the queues and ensure the total number of instances required by the process are available and return the farthest upstream and downstream position of the required instances.

```
            IF ok THEN

                ; If in-range checking is enabled, see if the
                ; position is out of range. If so, skip the
                ; selection.

                IF pm.ps[temp.idx,pm.ps.inrng] AND ((tgt.pct.u < 0) OR (tgt.pct.d >= 100)) GOTO 50

                ; If this is the first process that was found
                ; then record the position of the farthest product
                ; downstream. Otherwise, see if the new process
                ; is farther downstream from the process that was
                ; previously located

                IF (proc.idx == pm.ps.idx.none) THEN
                    proc.idx = temp.idx
                    sel.prt.pct = prt.pct.d
                    sel.tgt.pct = tgt.pct.d
                ELSE

                    ; See if the part or target percent of the new
                    ; process is further downstream. If the part or
                    ; target percent is equal, base the decision on
                    ; the instance that is NOT equal. If neither the
                    ; part or target percent is equal, then the
                    ; decision is based on the part or target
                    ; furthest downstream.

                    IF (sel.prt.pct == prt.pct.d) OR (sel.tgt.pct == tgt.pct.d) THEN
                        IF (sel.prt.pct == prt.pct.d) AND (sel.tgt.pct < tgt.pct.d) THEN
                            proc.idx = temp.idx
                            sel.prt.pct = prt.pct.d
                            sel.tgt.pct = tgt.pct.d
                        ELSE
                            IF (sel.tgt.pct == tgt.pct.d) AND (sel.prt.pct < prt.pct.d) THEN
                                proc.idx = temp.idx
                                sel.prt.pct = prt.pct.d
                                sel.tgt.pct = tgt.pct.d
                            END
                        END
                    ELSE
                        IF ((tgt.pct.d > sel.tgt.pct) AND (tgt.pct.d > sel.prt.pct)) OR ((prt.pct.d > sel.tgt.pct
                            proc.idx = temp.idx
                            sel.prt.pct = prt.pct.d
                            sel.tgt.pct = tgt.pct.d
                        END
                    END
                END
            END
        END
```

If all parts and targets are available for a process, the program will decide if this process should be marked for selection. If no other processes have been selected, then it will select the process. If another process had already been selected, it will pick the process that has the farthest downstream instance.

```
            ; Get the next process in the list

    50      temp.idx = pm.ps.rob.map[temp.idx,pm.ps.rob.nxt]

        UNTIL temp.idx == pm.ps.rob.nopx

   100

; If no process was selected, check to see if there is only 1 enabled
; process. If there is only 1, then select it.

        IF (proc.idx == pm.ps.idx.none) THEN
            IF (num.enabled == 1) AND (first.ena.idx <> pm.ps.idx.none) THEN
                IF NOT pm.ps[first.ena.idx,pm.ps.inrng] THEN
                    proc.idx = first.ena.idx
                END
            END
        END

; If a node was selected, set the pointer for the next set of processes
; to select. If a child process exists, then it is the entry point for the
; next set of processes. If no child process exists, the robot starting
; process is the next process set to select from.

        IF (proc.idx <> pm.ps.idx.none) THEN
            IF (pm.ps.rob.map[proc.idx,pm.ps.rob.chld] == pm.ps.rob.nopx) THEN
                pm.ps.rob.midx[rob.tsk,pm.ps.rob.crix] = pm.ps.rob.midx[rob.tsk,pm.ps.rob.stix]
            ELSE
                pm.ps.rob.midx[rob.tsk,pm.ps.rob.crix] = pm.ps.rob.map[proc.idx,pm.ps.rob.chld]
            END
        END

        RETURN
```

The program will iterate through all of the process. There is an additional check performed after all the processes have been checked. If only one process is enabled, then that process will automatically be selected.

Using a custom process selection program, you can change the selection criteria for a process. For example, the following code will force the selection of a process based on the part position, ignoring the status of any targets.

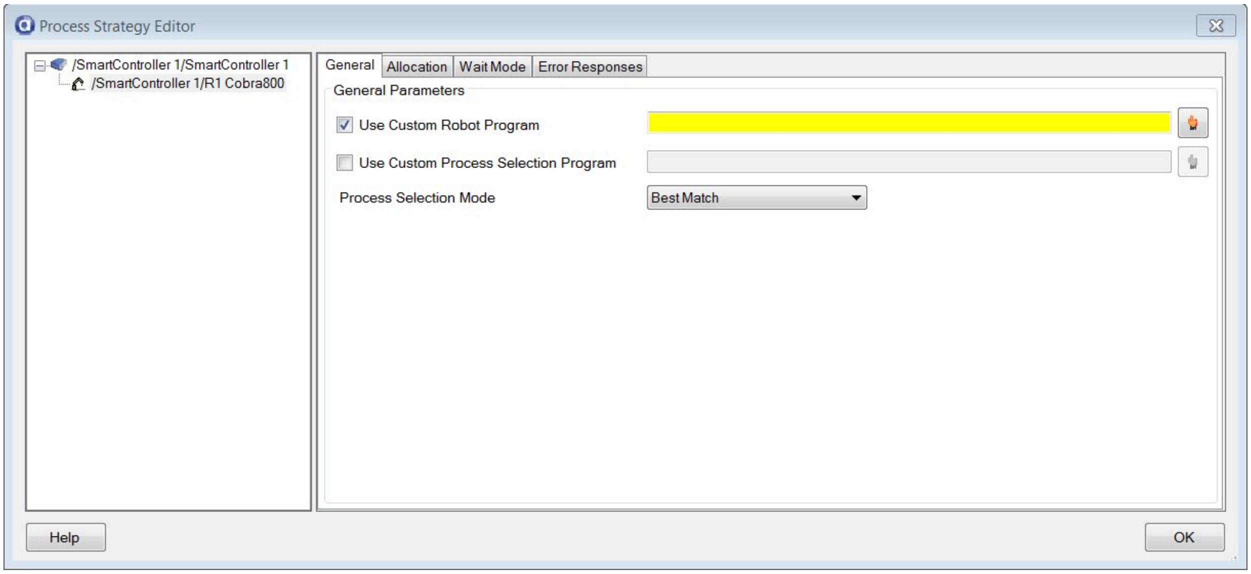
```
                ; Ignore the target availability and force process selection based on part position only

;                IF pm.ps[temp.idx,pm.ps.tr.excl] THEN
;                    CALL pm.ps.chk.targx(rob.tsk, temp.idx, tgt.pct.d, tgt.pct.u, ok)
;                ELSE
;                    CALL pm.ps.chk.targ(rob.tsk, temp.idx, tgt.pct.d, tgt.pct.u, ok)
;                END
                ok = TRUE
                tgt.pct.d = 0
                tgt.pct.u = 0

                IF ok THEN

                    ; If in-range checking is enabled, see if the
                    ; position is out of range. If so, skip the
                    ; selection.
```

Custom Robot Program: Overview

The process manager application software allows you to take complete control over the behaviors of a robot by customizing the main program that governs the robot top level behaviors. When a custom robot program is created, a copy of the default program is used as the basis for the custom program.

There default main robot program will call a process selection program and then execute that process. However, some applications may benefit from customizing the main robot program to force the selection of a process directly based on some custom condition or bypass the concept of processes entirely and command the robot to pick directly from the part and target queues. These two possibilities are detailed below.

The robot program can be customized in the process strategy editor:



Specification of Processes

One possibility for customization is to change the selection of processes. Below is an example of a program that does the following:

- Initializes the robot program moving it to an idle position
- Looks up the process index for 2 processes defined in the process manager
- Command the robot to execute the first process 3 times followed by 1 execution of process two.
- Check to see if the stop button was pressed

```
.PROGRAM cu.robot2()

; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

        AUTO REAL task.idx, i, sts
        AUTO REAL proc.idx.1, proc.idx.2

        tsk.idx = TASK()

        ; Initialize the robot
        CALL pm.rob.init(tsk.idx)
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Get the index associated with the first two processes in the process manager
        CALL pm.ps.map.idx(0, proc.idx.1)
        CALL pm.ps.map.idx(1, proc.idx.2)

        ; Control loop
        DO

            ; Execute the first process 3 time
            FOR i = 1 TO 3
                CALL pm.rob.process(tsk.idx, proc.idx.1, sts)
            END

            ; Execute the second process
            CALL pm.rob.process(tsk.idx, proc.idx.2, sts)

            ; Check process status
            CALL pm.chk.stat(is.running)

        UNTIL is.running == FALSE

        RETURN
.END
```

The process index used in the custom robot program in the call to ***pm.ps.map.idx*** is the process index associated with the process on the PC. This process index can be located in the part process editor in the process manager editor:

The program ***pm.ps.map.idx*** converts the process index on the PC to an index relative to the current robot task. It is this converted index that must be used when calling ***pm.rob.process***.

Accessing Queues

There is an alternative to using process selection to customize the robot behaviors. A custom application can directly access the part and target queues without using processes. There are times where the options allowed in the confines of a process are to restrictive. Using this technique, you can create more custom logic without having to specify low level motions of the robot.

Below is an example of a program that does the following:

- Initializes the robot program moving it to an idle position
- Looks up the queue numbers for a part and a part target.
- Command the robot to pick a part and then perform a placement of that part at a part target.
- Check to see if the stop button was pressed

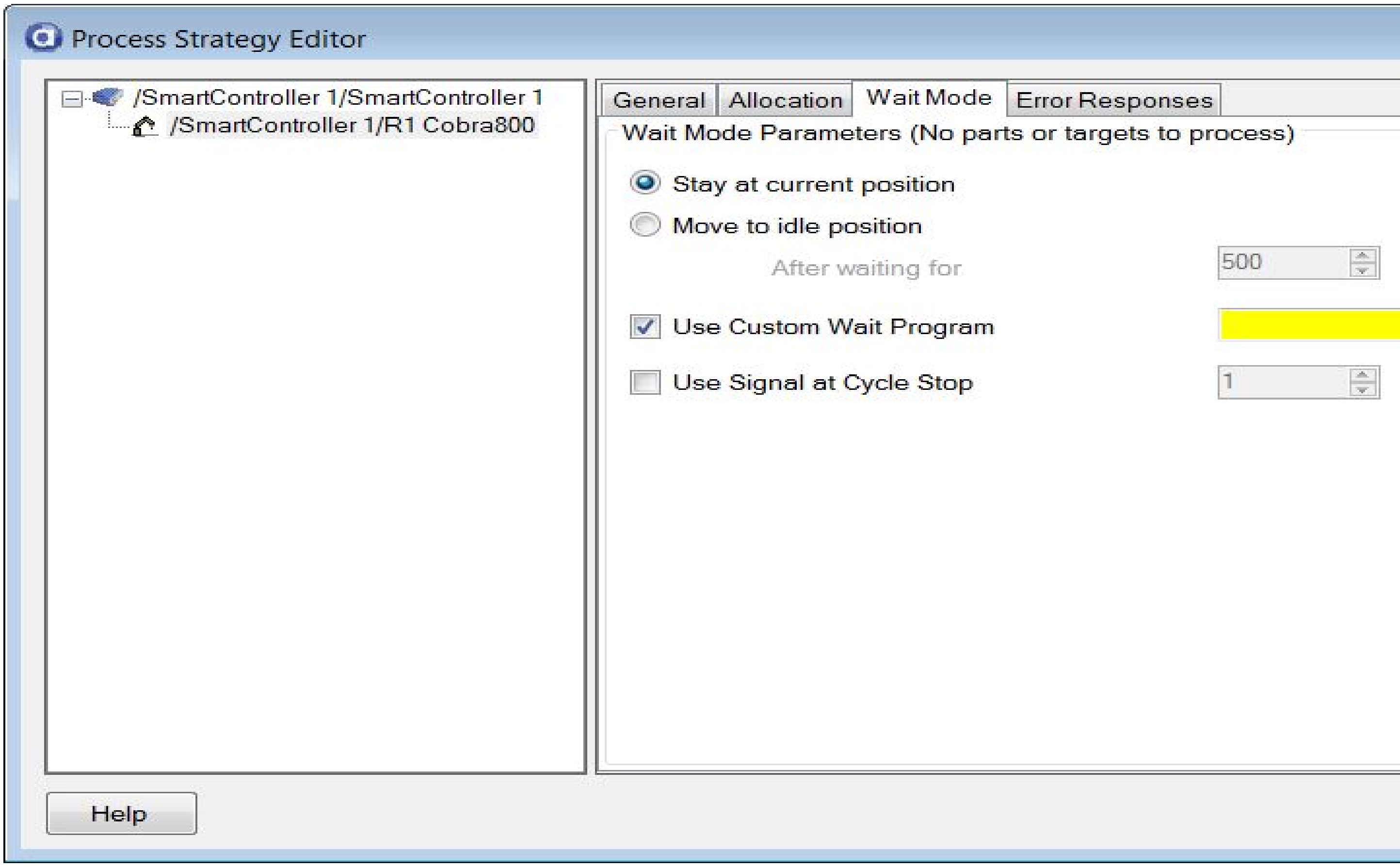
cu.robot3

Object

```
.PROGRAM cu.robot3()

; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

        AUTO REAL tsk.idx, is.reset, sts
        AUTO REAL part.idx, target.idx

        tsk.idx = TASK()

; Initialize the robot

        CALL pm.rob.init(tsk.idx)
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

; Get the part and target indexes

        CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
        CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

; Control loop

        DO

            ; Perform a pick
            CALL pm.rob.pick(tsk.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

            ; Perform a place
            CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

            ; Check to see if the process manager is still running
            CALL pm.chk.stat(is.running)

        UNTIL is.running == FALSE

        CALL pm.rob.clear(tsk.idx, sts)

        RETURN
.END
```

Custom Robot Wait Programs

When the robot does not have product available, it enters idle mode. The process strategy allows you to select the behavior of the robot when it enters an idle waiting mode:



By default you have the option to leave the robot at the current position or move it to the idle position after waiting a certain amount of time. If you want, you can customize the robot wait program. When it is customized, you start with a copy of the default program:

cu.rwait

```
.PROGRAM cu.rwait(rob.idx, $type, time, done)

; ABSTRACT:  Method called when the robot does not have a part or target
;        available. This program checks to see if the robot needs to be
;        moved to the idle location or should stay at the current location.
;
;        This can be called if no process is available -or- a process is
;        selected but the expected part or target is not in the robot queue.
;
;        This program will be called iteratively. This program should do a
;        check to see if the robot should be moved then return. If it issues
;        the move, it should set the done parameter to TRUE.
;
; INPUTS:        rob.idx         Index of the robot
;                $type           Type of object the robot is waiting for
;                                  or empty ("") if no process has been selected
;                time            Time that has elapsed. Units are in ms.
;
; OUTPUTS:       done            Has the robot been moved? Setting this to TRUE
;                                  will indicate the robot has been moved to the
;                                  waiting position.
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2009 by Adept Technology, Inc.
```

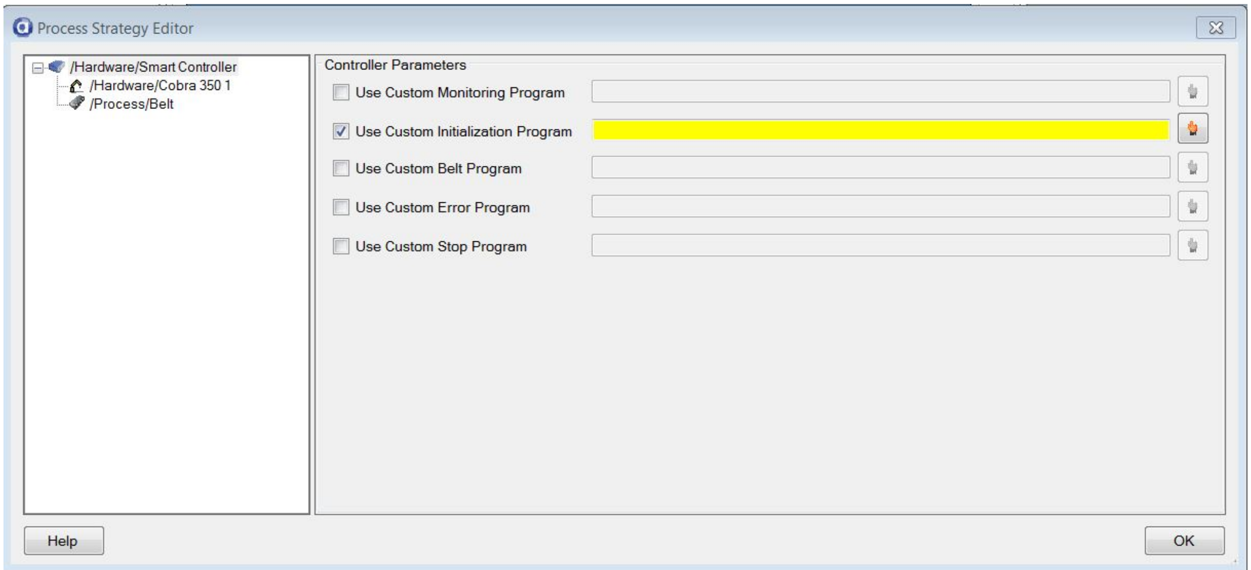
The default program gets called while the robot is waiting. By default this program implements the logic described above. If Move to Idle is enabled, it will monitor the elapsed time and move the robot to the idle position as needed. This can be seen in the body of the program:

```
        AUTO REAL move.to.idle, values[4]

; Extract the parameters for the waiting

        CALL pm.psp.rob.pars(rob.idx, values[])

; See if we need to move to idle

        IF (values[psp.rob.mvidle]) THEN
            IF (time > values[psp.rob.delay]) THEN
                move.to.idle = TRUE
            ELSE
                move.to.idle = FALSE
            END
        ELSE
            move.to.idle = FALSE
        END

; Perform the move to idle position, if needed

        IF (move.to.idle) THEN
            CALL pm.mv.idle(rob.idx)
            done = TRUE
        END

        RETURN
```

Customizing Stop Behavior

When the process manager application is stopped, the system will perform a stop procedure to stop the V+ tasks used by the runtime.

The user can define a V+ program that is called when the process manager is in the process of stopping through the following menu:



Custom Stop Program

The default stop program looks like this:

cu.stop

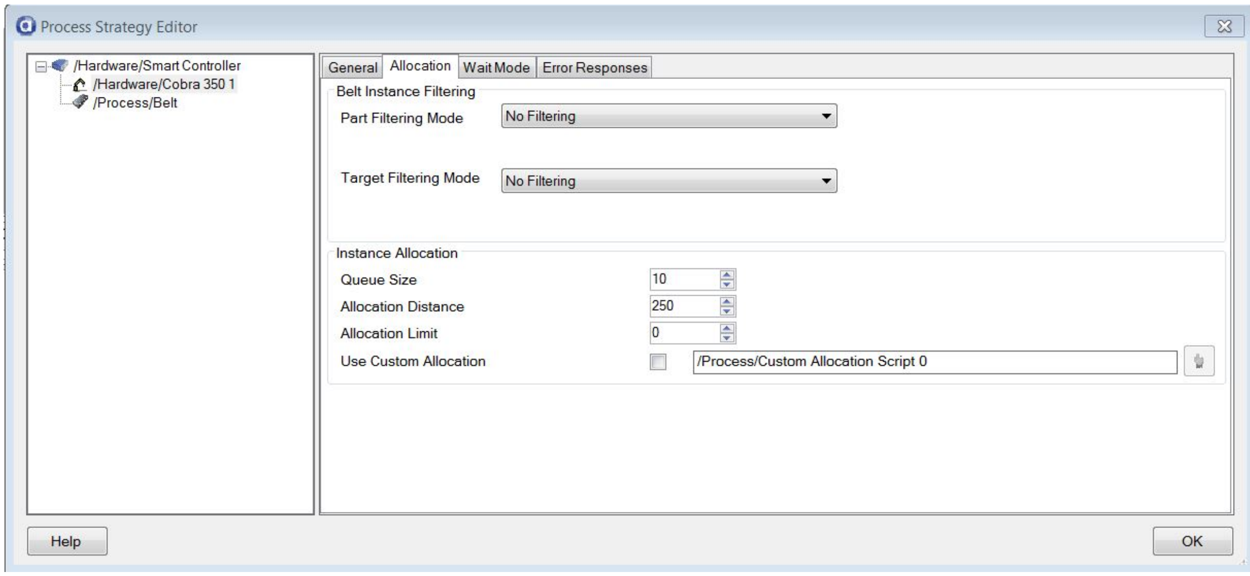
```
.PROGRAM cu.stop(tasks.stopped)

; ABSTRACT:  Perform operations required when the application is stopped.
;       This method will be called before tasks have been stopped and
;       after the tasks have been stopped.
;
;       This program will be allowed to run for at most 500 ms. It should only
;       perform operations that will be completed within that 500 ms time limit.
;
; INPUTS:       tasks.stopped   Have the tasks been stopped?
;
; OUTPUTS:      None
;
; SIDE EFFECTS: None
;
;* Copyright (c) 2007-2008 by Adept Technology, Inc.


; Check the belts to see if they need to stop running

        IF (tasks.stopped) THEN
            CALL pm.psp.belt(pm.ps.tsk, FALSE)
        END

        RETURN
.END
```

The stop program will be called twice. It will first be called before the V+ process manager tasks are stopped. After the V+ tasks are stopped, the program will be called a second time. When the custom program is executed, the program will be allowed to run for 500 ms each time. If the program has not completed execution in the allowed time period, the program will be aborted.

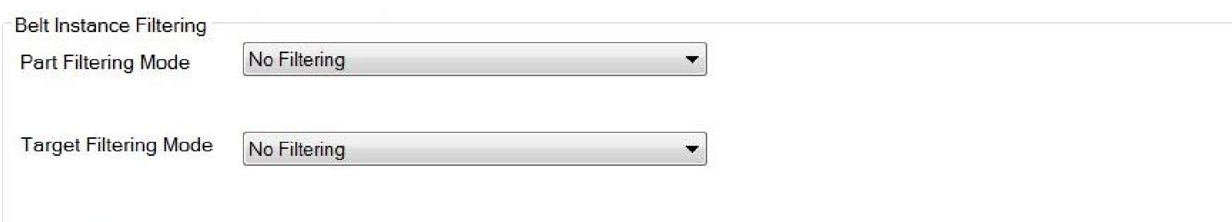
The default stop program will perform any belt stop operations that are configured in the process manager.

Part and Target Queues

Parts and target instances are generated by hardware sources. These instances are allocated to robots by the packaging software based on a number of factors. The basic parameters that dictate when and how parameters are allocated are located in the Process Strategy Editor under the Allocation tab for each robot:



Belt Instance Filtering



For a given robot, you can specify how you want it to manage belt based part and target instances. When instances are first created, these settings dictate how the flow of instances will be managed for the given robot. You have the following modes of instance filtering to choose from:

No Filtering	The robot will attempt to pick all instances that are available.
Pick/Skip Instances	Robot will be allocated ‘m’ instances then skip ‘n’ instances. This is interpreted as a strict requirement.
Percentage of	The robot will be allocated ‘x’ percentage of instances spread across the range of instances.

Instances	
Skip Rate	Robot will attempt to skip instances to achieve the specified rate of instances for downstream robots. The system calculates the instantaneous instances/minute on the belt and set the allocation % based on desired skip rate.
Relative Belt Position	The instances are spaced out based on the minimum distance between instances in the direction of the conveyor travel. This can be used if you have a much larger number of instances that can be processed by the robot.
Pick/Skip Pallets	The robot will be allocated ‘m’ pallets then skip ‘n’ pallets. This is interpreted as a strict requirement.

For many applications, these parameters are not needed. However, for applications that have a large number of instances or have multiple robots, these settings may be useful. For example, in a multi-robot applications, you may want to configure the first robot to use a ***Skip Rate*** allocation so the downstream robots are supplied a constant rate of product. This may allow the system to smooth out the flow of instances and achieve higher throughputs and more even distribution of work.

Instance Allocation

After the ***Belt Instance Filtering*** has been applied, a collection of instances will have been identified for the robot to process. The ***Instance Allocation*** parameters identify when those instances will be sent to the robot.

Queue Size	Identifies how many instances a robot can actively process and track. If there are more instances available than can fit in the queue, those instances will be sent as instances are process by the robot. If unsent instances pass out of the belt window of the robot, they will automatically be moved for processing by the next downstream robot.
Allocation Distance	The distance from the upstream belt window an instance must be before allocating to the robot.
Allocation Limit	The distance from the belt window pick limit an instance must be before allocating to the robot. This can be used to push the allocation further upstream of the pick limit. It is most often

	used for very fast belts where the robot needs extra time to position itself for picking an instance.
Use Custom Allocation/Custom Allocation Script	Should a custom allocation script be used. Details about custom allocation scripts can be located in the ***Custom Allocation Scripts*** section of this guide.

If an instance meets all the parameters for instance allocation, it will be sent to the robot controller for runtime processing and tracking. The instances sent to the robot continue to be tracked on the PC, but the actual processing of each instance is managed by the robotic controller.

Representation on Controller

Rotary Buffer

When an instance is sent to the robotic controller for processing, it is placed in a rotary buffer. The data structures and tables used to track parts and part targets are completely separate. Each part or target type is kept in a separate area of the respective data tables.

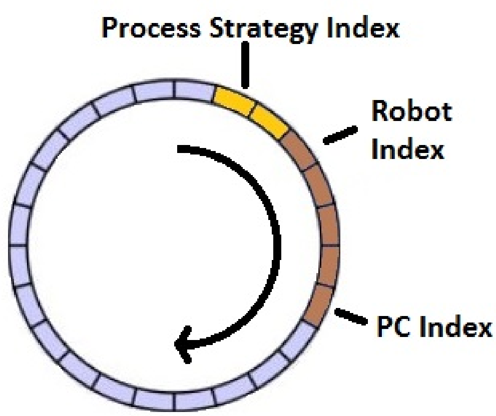
The rotator buffer has three different pointers:

- Robot Index: Identifies the next instance that will be processed by the robot.
- PC Index: Identifies the position in the buffer where new instances will be added.
- Process Strategy Index: Identifies the position in the buffer of the last instance whose status has been sent to the controller.

If no instances are in the queue, then all three indexes will be at the same position. When the PC sends instances for processing by the robot, the instance information is placed into the rotary buffer and the PC index is incremented so it points to the last instance. When there are instances in the queue, the robot will start processing the instances in the queue. As instances are processed, the robot index will be incremented so it is closer and closer to the PC index. If all instances are processed by the robot, the robot index and PC index will be equal.

The robot tasks are not responsible for communicating the status of instances processed. That task is handled by the V+ process strategy task. It monitors the Process Strategy Index. When the Robot Index is incremented, the process strategy task recognizes that instances have been processed. It will then send instance information back to the PC. As details of the instances are sent to the PC, it will increment the Process Strategy Index. When the Process Strategy Index catches up to the Robot Index, it means the status of all processed instances have been sent to the PC for processing.

This graphic illustrates the relationship between the three indexes:



There are 5 instances in the queue for processing by the robot. There are 2 instances that the robot has processed but the PC has not been notified of their status. Any new instances will be added at the end of the robot queue at the PC Index location. The "direction" of processing of the queue is represented by the arrow in the graphic.

V+ API Calls

There are several V+ programs that can be used by an application to interact with the part and target queues.

pm.prt.get.idx(rob.idx, $part, part.idx) pm.trg.get.idx(rob.idx, $target, target.idx)	Retrieve the index of the instance queue for a robot associated with a part or target.
pm.prt.avail(rob.idx, part.idx, count) pm.trg.avail(rob.idx, trg.idx, count)	Retrieve the number of instances for processing by the robot for a part or target queue.
pm.prt.get(rob.idx, part.idx, obj.idx, $id, position, reference, pal.idx) pm.trg.get(rob.idx, trg.idx, obj.idx, $id, position, reference, pal.idx)	Get details concerning a part or target instance in the robot queue.
pm.prt.getloc(rob.idx, part.idx, obj.idx, pct) pm.trg.getloc(rob.idx, trg.idx, obj.idx, pct)	Returns the relative position on the belt of a part or target instance in the robot queue.
pm.prt.move.ptr(ptr) pm.trg.move.ptr(ptr)	Increment the pointer to the next instance in the part or target queue.

Refer to the documentation on these programs for examples for how each of these can be used within a V+ application.

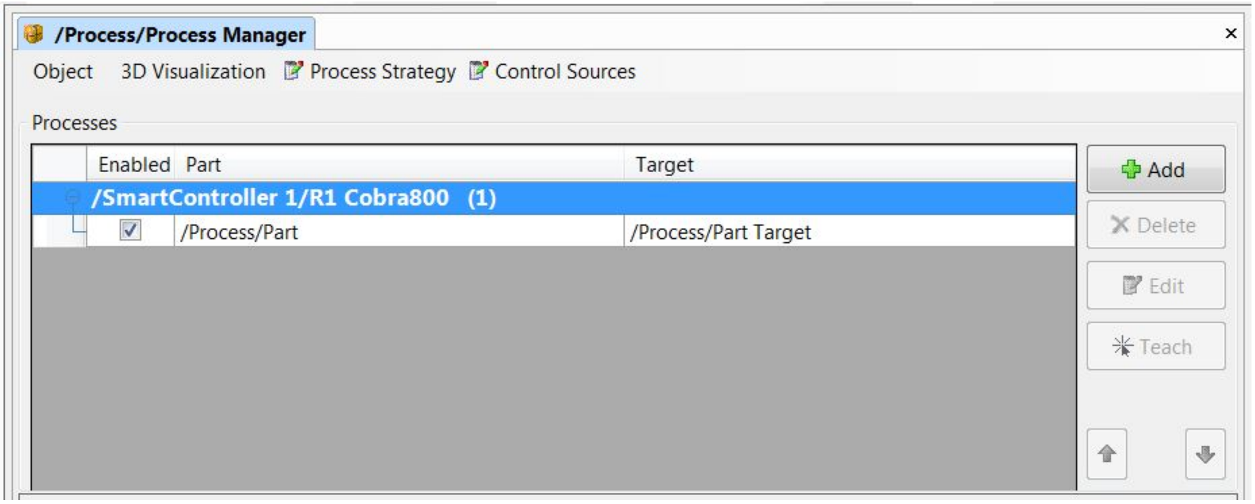
Recipe Management

The requirements for recipe management varies from one application to another. There are several features of the packaging and ACE software that facilitate different application requirements:

- Calibration sharing
- Separation of motion parameters
- C# Scripting

Calibration Sharing

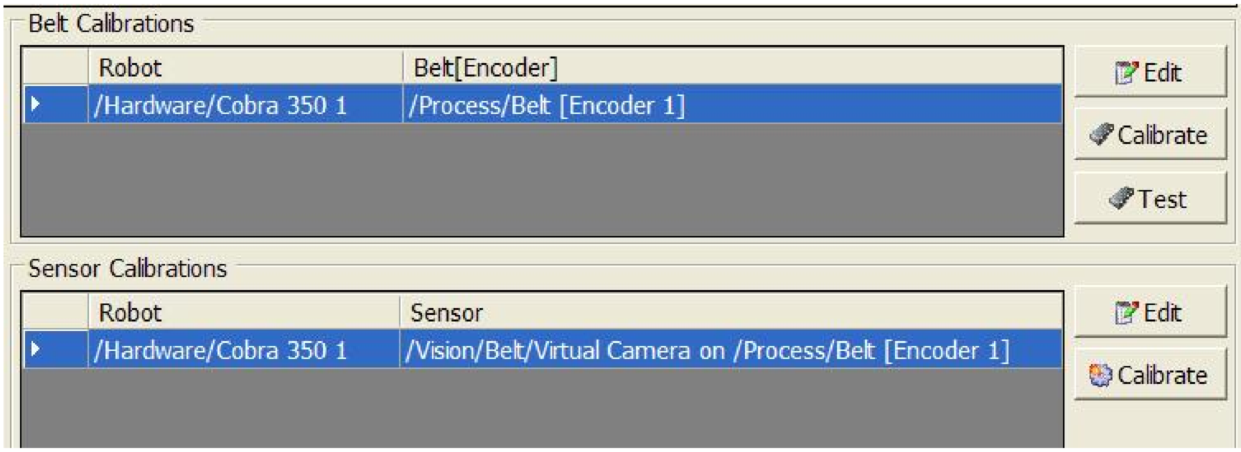
When a user defines a process, the software detects what kinds of robot-to-hardware calibrations are required. For example, if a robot is configures to pick a part from a belt based camera like this:



The process manager automatically understands that two calibrations are required:

- Robot-to-belt calibration
- Robot-to-belt camera calibration

These calibrations will appear automatically in the process manager display:



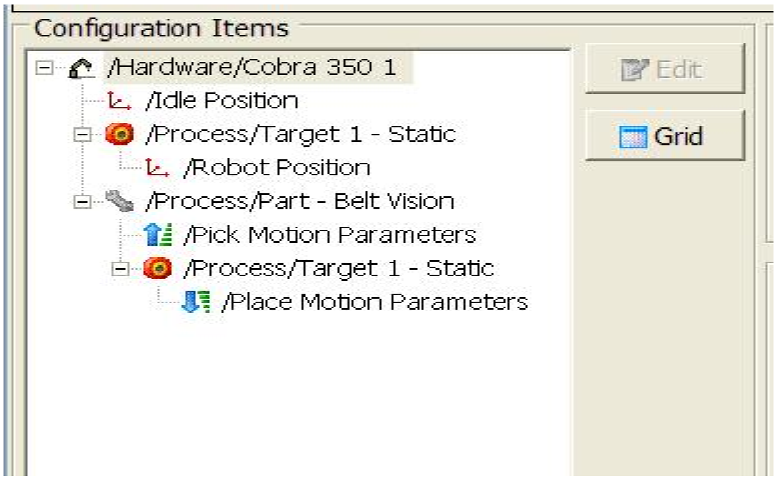
It is important to understand that the calibration is between the robot and the specified hardware source. If you have multiple processes with different parts and targets, but the parts are targets are associated with the set of cameras and belts, only 1 calibration is required for each hardware resource. So, if we have 2 different parts located by the same belt camera, we would only have one robot-to-belt and one robot-to-belt camera calibrations.

When multiple process managers are in the workspace, each one will show their own list of robot-to-hardware calibrations based on the process configuration. If a process in one process manager references the same hardware resources as a process in a different calibration, they will share the same robot-to-hardware calibration. For example, if you have a process manager with a part that is located using a camera, it will have a robot-to-camera calibration for that camera. If there is another process manager in the workspace with a different part that is located using the same camera, it will also have a robot-to-camera calibration for that camera and it will be the same as the camera calibration as the first process manager.

It is important to note that the sharing of calibrations is handled through a process of replication. Each process manager will maintain a local copy of the robot-to-hardware calibrations, but when a calibration is performed in one process manager, the newly updated values will overwrite all other matching robot-to-hardware calibrations referenced in other process managers in the workspace.

Separation of motion parameters

Whereas hardware calibrations are shared between process managers, the configuration items that define the locations, offsets, timing, and other settings associated with robot operations are different in each process manager in the workspace:



If you have two process managers in the workspace with identical processes, they will share the robot-to-hardware calibrations, but will maintain completely separate motion settings. As such, each process manager represents a different product run configuration. For many applications, this is sufficient to account for recipe management.

C# Scripting

In some applications, a user may want a more customized solution to recipe management. For example, perhaps an application only requires a different pallet layout and having separate process managers with completely separate motion settings would be considered burdensome. In this case, a C# script can be used to make targeted changes. For example, a script can be created for each product configuration and the appropriate script can be run at product changeover time.

It is important to note that many parameters cannot be changed while a process manager is running. You typically will need to stop running a process manager, execute the product change-over script, then restart the process manager.

V+ Module Documentation

This section details many of the methods that are available for use by a custom V+ applicaiton.

See Also

See "V+ AceServer Module Documentation" on page 724

See "V+ ACE Sight Module Documentation" on page 746

See "V+ End-Effector Program Documentation" on page 750

See "V+ Process Manager Documentation" on page 762

See "V+ Remote Library Documentation" on page 814

V+ AceServer Module Documentation

The V+ AceServer library documents the V+ programs a user might use in the process of customizing an application leveraging ACE.

sv.create_msg(msg_num, handle, ptr)

Abstract

Creates a new message, initializing the header for writing.

Input Parameters

Parameter	Description
msg_num	Message number to create

Output Parameters

Parameter	Description
handle	Handle to new message
ptr	Pointer into the new message

sv.read_adouble(handle, ptr, offset, value[], n_items)

Abstract

Read an array of double-precision floats from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at
offset	Offset into output array for first item. 0 is default

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value[]	Array filled with data read
n_items	Number of items read

sv.read_ashort(handle, ptr, offset, value[], n_items)

Abstract

Read an array of 16-bit integers from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at
offset	Offset into output array for first item. 0 is default

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value[]	Array filled with data read
n_items	Number of items read

sv.read_asingle(handle, ptr, offset, value[], n_items)

Abstract
Read an array of single-precision float variables from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at
offset	Offset into output array for first item. 0 is default

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value[]	Array filled with data read
n_items	Number of items read

sv.read_astring(handle, ptr, offset, $value[], n_items)

Abstract
Read an array of strings from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description

handle	Index of the message to read from
ptr	Position in the message buffer to read at
offset	Offset into output array for first item. 0 is default

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
$value[]	Array filled with data read
n_items	Number of items read

sv.read_bool(handle, ptr, value)

Abstract

Read a boolean from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer
   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
```

```
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.read_byte(handle, ptr, value)

Abstract

Reads a byte from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer
   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val
   ; Save the original position of the pointer
   saved.ptr = ptr
   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
```

```
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)
   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.read_byte(handle, ptr, value)

Abstract

Reads a byte from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer
   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val
   ; Save the original position of the pointer
   saved.ptr = ptr
   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
```

```
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)
   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.read_double(handle, ptr, value)

Abstract

Read a double-precision float from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                 ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
```

```
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)

.END
```

sv.read_short(handle, ptr, value)

Abstract

Read a 16-bit integer from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;              ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
```

```
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)

.END
```

sv.read_single(handle, ptr, value)

Abstract

Read a single-precision float from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
value	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;              ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
```

```
    CALL sv.write_byte(8, handle, ptr)
    CALL sv.write_double(100.1, handle, ptr)
    CALL sv.write_short(2, handle, ptr)
    CALL sv.write_single(121.2, handle, ptr)
    CALL sv.write_string("Text", handle, ptr)
    CALL sv.write_trans(NULL, handle, ptr)

    ; Read the data that was written
    CALL sv.read_bool(handle, saved.ptr, bool.val)
    CALL sv.read_byte(handle, saved.ptr, byte.val)
    CALL sv.read_double(handle, saved.ptr, double.val)
    CALL sv.read_short(handle, saved.ptr, short.val)
    CALL sv.read_single(handle, saved.ptr, single.val)
    CALL sv.read_string(handle, saved.ptr, $string.val)
    CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.read_string(handle, ptr, $string)

Abstract

Read a string from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
$string	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                ptr: Position in the message buffer

    AUTO REAL saved.ptr
    AUTO REAL bool.val, byte.val, double.val, short.val
    AUTO REAL single.val
    AUTO $string.val
    AUTO LOC trans.val

    ; Save the original position of the pointer
    saved.ptr = ptr

    ; Write some data
    CALL sv.write_bool(TRUE, handle, ptr)
```

```
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.read_trans(handle, ptr, trans)

Abstract

Read a transform from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message
trans	Value of the parameter read from the message stream

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;              ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
```

```
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.stop_collect(robot.num)

Abstract
Stop the data collection operations on the specified robot.

Input Parameters

Parameter	Description
robot.num	The robot number to stop data collection on.

Output Parameters

None

sv.sig.is.equal(sig.num, expected.val, is.equal.res)

Abstract
Compares a signal to an expected value. This method is employed when a program checks a signal is at the expected state. It encapsulates the comparison process but it also manages the fact a program is running on an emulator or not. In the case of an emulator, reading inputs can be a problem as in emulator mode external system generating the input signals is not available. In emulator mode, this method will always consider the signal is at the expected state and will then always return ***true***.

Input Parameters

Parameter	Description
sig.num	the signal number we want to check
expected.val	The expected signal value

Output Parameters

Parameter	Description
is.equal.res	The result value, indicating if signal is at the expected value

Example

```
.PROGRAM ex1()
; ABSTRACT: Demonstrate comparing a signal to an expected value
;
; INPUTS:

   AUTO REAL my.signal1 = 1001
   AUTO REAL my.signal2 = 1002
   AUTO REAL is.off, is.on

   ; Check a signal is not at an incorrect value. This is checking an error case
   ; In emulator mode, is.off will always be true and program will not generate errors
   CALL sv.sig.is.equal(my.signal1, FALSE, is.off)

   IF NOT is.off THEN
      TYPE "input is not at the correct state : it is on and should be off."
   END

   ; wait a signal is at the expected value. If emulation mode is active, true will be returned and program can continue.
   ; In normal mode, the signal is read periodically and is compared to expected value.
   DO
      CALL sv.sig.is.equal(my.signal2, TRUE, is.on)
   UNTIL is.on

.END
```

sv.write(handle, ptr)

Abstract

Write a message to the client.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

None

sv.write_abyte($string[], offset, count, handle, ptr)

Abstract

Read an array of 16-bit integers from the message buffer. The message pointer is incremented after the read.

Input Parameters

Parameter	Description

$string[]	Array of strings to write
offset	1-based starting byte offset in the string array
count	Number of bytes to transfer
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

sv.write_adoubl(first_index, value[], n_items, handle, ptr)

Abstract

Write an array of doubles to the output message at the specified offset.

Input Parameters

Parameter	Description
first_index	First index in the array to write
value[]	Array of data values to write
n_items	Number of items in value[] to write into the buffer
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

sv.write_ashort(first_index, value[], n_items, handle, ptr)

Abstract

Write an array of shorts to the output message at the specified offset.

Input Parameters

Parameter	Description
first_index	First index in the array to write
value[]	Array of data values to write
n_items	Number of items in value[] to write into the buffer

handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

sv.write_asingl(first_index, value[], n_items, handle, ptr)

Abstract
Write an array of floats to the output message at the specified offset.

Input Parameters

Parameter	Description
first_index	First index in the array to write
value[]	Array of data values to write
n_items	Number of items in value[] to write into the buffer
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

sv.write_astrin(first_index, $value[], n_items, handle, ptr)

Abstract
Write an array of strings to the output message at the specified offset.

Input Parameters

Parameter	Description
first_index	First index in the array to write
$value[]	Array of data values to write
n_items	Number of items in value[] to write into the buffer
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

sv.write_bool(value, handle, ptr)

Abstract

Write a boolean to the output message at the specified offset.

Input Parameters

Parameter	Description
value	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                    ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
```

```
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.write_byte(value, handle, ptr)

Abstract

Write a byte to the output message at the specified offset.

Input Parameters

Parameter	Description
value	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
```

```
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.write_short(value, handle, ptr)

Abstract

Write a 16-bit integer to the output message at the specified offset.

Input Parameters

Parameter	Description
value	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                 ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
```

```
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.write_hdr(handle, ptr)

Abstract
Write the header for an output message.

Input Parameters

Parameter	Description
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

None

sv.write_single(value, handle, ptr)

Abstract
Write a float to the output message at the specified offset.

Input Parameters

Parameter	Description
value	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
```

```
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.write_string($string, handle, ptr)

Abstract

Write a string to the output message at the specified offset.

Input Parameters

Parameter	Description
$string	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
```

```
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

sv.write_trans(trans, handle, ptr)

Abstract

Write a string to the output message at the specified offset.

Input Parameters

Parameter	Description
trans	Value to write
handle	Index of the message to read from
ptr	Position in the message buffer to read at

Output Parameters

Parameter	Description
ptr	Index pointing to the next data item in the message

Example

```
.PROGRAM ex1.msg(handle, ptr)
; ABSTRACT:   Demonstrate writing then reading to a message buffer
;
; INPUTS:      handle: Index of the message to write into
;                  ptr: Position in the message buffer

   AUTO REAL saved.ptr
   AUTO REAL bool.val, byte.val, double.val, short.val
   AUTO REAL single.val
```

```
   AUTO $string.val
   AUTO LOC trans.val

   ; Save the original position of the pointer
   saved.ptr = ptr

   ; Write some data
   CALL sv.write_bool(TRUE, handle, ptr)
   CALL sv.write_byte(8, handle, ptr)
   CALL sv.write_double(100.1, handle, ptr)
   CALL sv.write_short(2, handle, ptr)
   CALL sv.write_single(121.2, handle, ptr)
   CALL sv.write_string("Text", handle, ptr)
   CALL sv.write_trans(NULL, handle, ptr)

   ; Read the data that was written
   CALL sv.read_bool(handle, saved.ptr, bool.val)
   CALL sv.read_byte(handle, saved.ptr, byte.val)
   CALL sv.read_double(handle, saved.ptr, double.val)
   CALL sv.read_short(handle, saved.ptr, short.val)
   CALL sv.read_single(handle, saved.ptr, single.val)
   CALL sv.read_string(handle, saved.ptr, $string.val)
   CALL sv.read_trans(handle, saved.ptr, trans.val)
.END
```

V+ ACE Sight Module Documentation

The V+ ACE Sight library documents the default ACE Sight V+ programs.

as.save.image($filename, $ip, seq.idx, tool.idx, status)

Abstract

Save the image associated with the vision tool into a file.

Input Parameters

Parameter	Description
$filename	The name of the file to save into
$ip	The IP address of the PC
seq.idx	The index of the sequence
tool.idx	The index of the virtual camera ; or other image tool

Output Parameters

Parameter	Description
status	Status of the operation. 0 = Success

check_tracking(tracking_ok)

Abstract

This function is used to check if the robot is successfully tracking the belt

Input Parameters

None

Output Parameters

Parameter	Description
tracking_ok	If true, the robot is tracking the belt successfully; otherwise an error occured while tracking the belt

clear_queue(queue_index)

Abstract

Clears an ACE Sight queue

Input Parameters

Parameter	Description
queue_index	The index of the queue to clear

Output Parameters

None

getinstance(queue_index, flags, location, model, encoder, visionx, visiony, visionrot)

Abstract

This function is the main function called to retrieve the instance.

Input Parameters

Parameter	Description
queue_ index	This is the queue index in which we want to retrieve the instance
flags	Flag bits to control operation of the routine: • **Bit 1:** If set, routine waits for queue element to be defined; otherwise, "fail" if the element is not defined • **Bit 2:** If set, do not remove the instance from the queue; otherwise, remove the instance from the queue. • **Bit 3:** If set, we are sure there is an instance in the queue. When using this bit, the calling program is responsible for checking if an instance is ready in the queue; otherwise, getinstance will check if there is an instance before retrieving it from the queue.

Output Parameters

Parameter	Description
location	Location of the part found by iSight. The location can be relative to the belt reference frame or the robot reference frame depending on the option selected in the communication tool on iSight side.
model	Model Index (-1 if there was an error)
encoder	Contains the encoder value when the instance was found.
visionx	Countains the part X position in the vision coordinates reference frame
visiony	Countains the part Y position in the vision coordinates reference frame

visionrot	Countains the part Rotation in the vision coordinates reference frame

reset_seq($myip, seq_id)

Abstract
Reset an ACE Sight sequence.

Input Parameters

Parameter	Description
$myip	String containing the IP Address of the ACE Sight vision server
seq_id	Index of the sequence to reset

Output Parameters

None

set_as_exec_mod($myip, seq_id, mode)

Abstract
Set the ACE Sight execution mode for a given sequence.

Input Parameters

Parameter	Description
$myip	String containing the IP Address of the ACE Sight vision server
seq_id	Index of the sequence to reset
mode	Desired ACE Sight execution. 0 = Single execution mode; otherwise: Continuous execution mode

Output Parameters

None

V+ End-Effector Program Documentation

The V+ end-effector library documents the V+ programs a user might use to interact with the end-effector. These programs are a sub-set of the Ace Server program module.

End-Effector Library Error Codes

If a problem is encountered when executing an end-effector library method, the following error codes can be returned:

Error Code	Description
-20003	The end-effector will not close
-20004	The end-effector will not open
-20010	Invalid end-effector referenced
-20026	The end-effector will not extend
-20027	The end-effector will not retract
-20030	A part is detected in the end-effector
-20031	No part detected in the end-effector

ee.chk.er(ee.idx, tip.idx, state, sts)

Abstract

Checks the status of the end-effector for extend or release input signals. The gripper waits until the end-effector reaches the desired state before moving.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of operation

ee.chk.oc(tsk.idx, tip.idx, state, sts)

Abstract

Checks the status of the gripper for an open or close input signal. The robot waits until the gripper reaches the desired state.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of operation

ee.clroff(ee.idx, sts)

Abstract

Clears all refinement offsets associated with the end-effector.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector

Output Parameters

Parameter	Description
sts	Status of operation

ee.presence(tsk.idx, tip.idx, state, sts)

Abstract

Check if the part presence sensor is in the desired state. The robot waits until the gripper reaches the desired state.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
state	Desired state of the part presence sensors

Output Parameters

Parameter	Description
sts	Status of operation

ee.release(ee.idx, tip.idx, state, sts)

Abstract

Turns the release signals on the end-effector on or off.

Input Parameters

Parameter	Description
tip.idx	Zero-based index of the tip to access: -1 All tips 0...N The tip to operate with
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of operation

ee.select.ck(ee.idx, tip.idx, sts)

Abstract

Checks for any end-effector selection errors.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access

Output Parameters

Parameter	Description
sts	Status of operation

ee.select(ee.idx, tip.idx)

Abstract

Performs a tip selection, extending or retracting tips as needed. It also checks to see if a tip selection program is associated with the current robot end-effector. If one is specified it is executed.

Input Parameters

Parameter	Description	
ee.idx	Index of the end-effector	
tip.idx	Zero based index of the tip to access	
	-1	All tips
	0...N	The tip to operate with

Output Parameters

None

Example

```
.PROGRAM a.ex2.gripper(ee.idx)

   AUTO REAL sts
   AUTO LOC tool.trans

   ; Select the first tip
   CALL ee.select(ee.idx, 0)

   ; Select all tips
   CALL ee.select(ee.idx, -1)

   ; Unselect all tips
   CALL ee.unselect(ee.idx)

   ; Extend tip #1 and #2
   CALL ee.state(ee.idx, 0, TRUE, TRUE, sts)
   CALL ee.state(ee.idx, 1, TRUE, TRUE, sts)
```

```
   ; Set the offset associated with tip #2
   CALL ee.setoff(ee.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL ee.trans(ee.idx, 0, tool.trans, sts)

 .END
```

ee.setoff(ee.idx, tip.idx, tool.trans, sts)

Abstract

Associates an offset with a specific end-effector tip.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access
tool.trans	Gripper tip transform

Output Parameters

Parameter	Description
sts	Status of operation

Example

```
.PROGRAM a.ex2.gripper(ee.idx)

   AUTO REAL sts
   AUTO LOC tool.trans

   ; Select the first tip
   CALL ee.select(ee.idx, 0)

   ; Select all tips
   CALL ee.select(ee.idx, -1)

   ; Unselect all tips
   CALL ee.unselect(ee.idx)
```

```
; Extend tip #1 and #2
CALL ee.state(ee.idx, 0, TRUE, TRUE, sts)
CALL ee.state(ee.idx, 1, TRUE, TRUE, sts)

; Set the offset associated with tip #2
CALL ee.setoff(ee.idx, 0, NULL, sts)

; Get the offset associated with tip #2
CALL ee.trans(ee.idx, 0, tool.trans, sts)

.END
```

ee.gr.settle(ee.idx, tip.idx, state, sts)

Abstract

Waits for the required settling time for the end-effector.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of operation

ee.state(ee.idx, tip.idx, state, check.inputs, sts)

Abstract

Changes the state of the end-effector tip.

Input Parameters

Parameter	Description

ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access
state	State of the gripper
check.inputs	Check to see if the end-effector is waiting for an input

Output Parameters

Parameter	Description
sts	Status of operation

Example

```
.PROGRAM a.ex2.gripper(ee.idx)

   AUTO REAL sts
   AUTO LOC tool.trans

   ; Select the first tip
   CALL ee.select(ee.idx, 0)

   ; Select all tips
   CALL ee.select(ee.idx, -1)

   ; Unselect all tips
   CALL ee.unselect(ee.idx)

   ; Extend tip #1 and #2
   CALL ee.state(ee.idx, 0, TRUE, TRUE, sts)
   CALL ee.state(ee.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
   CALL ee.setoff(ee.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL ee.trans(ee.idx, 0, tool.trans, sts)

.END
```

ee.gr.tip.er(tsk.idx, tip.idx, state)

Abstract
Changes the extend or retract state of the gripper tip for the specified robot.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
state	State of the gripper

Output Parameters

None

ee.tip.erdw(ee.idx, tip.idx)

Abstract

Performs the required extend or retract dwell of the end-effector tip.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access

Output Parameters

None

ee.tip.oc(ee.idx, tip.idx, state, sts)

Abstract

Changes the open and closed state of the end-effector tip.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector
tip.idx	Zero based index of the tip to access
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of operation

pm.gr.trans(tsk.idx, tip.idx, tool.trans, sts)

Abstract

Associates an offset with a specific gripper tip.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access

Output Parameters

Parameter	Description
tool.trans	The gripper tip transformation
sts	Status of operation

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

   AUTO REAL tsk.idx, sts
   AUTO LOC tool.trans
   tsk.idx = TASK()

   ; Select the first tip
   CALL pm.gr.select(tsk.idx, 0)

   ; Select all tips
   CALL pm.gr.select(tsk.idx, -1)

   ; Unselect all tips
   CALL pm.gr.unselect(tsk.idx)

   ; Extend tip #1 and #2
   CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
```

```
   CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
   CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)

.END
```

ee.unselect(ee.idx)

Abstract

Retracts all of the end-effector tips.

Input Parameters

Parameter	Description
ee.idx	Index of the end-effector

Output Parameters

None

Example

```
.PROGRAM a.ex2.gripper(ee.idx)

   AUTO REAL sts
   AUTO LOC tool.trans

   ; Select the first tip
   CALL ee.select(ee.idx, 0)

   ; Select all tips
   CALL ee.select(ee.idx, -1)

   ; Unselect all tips
   CALL ee.unselect(ee.idx)

   ; Extend tip #1 and #2
   CALL ee.state(ee.idx, 0, TRUE, TRUE, sts)
   CALL ee.state(ee.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
```

```
    CALL ee.setoff(ee.idx, 0, NULL, sts)

    ; Get the offset associated with tip #2
    CALL ee.trans(ee.idx, 0, tool.trans, sts)

.END
```

V+ Process Manager Documentation

The V+ process manager library documents the V+ programs a user might use in the process of customizing a process manager application.

pm.ace.blt.cam(tsk.idx, $camera, remote, encoder)

Abstract
Indicate to ace that a camera needs to be triggered.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
$camera	Name of the camera that needs to be triggered
remote	Is the camera remote or local?
encoder	Encoder reference

Output Parameters

None

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use.

pm.ace.blt.ltch(blt.idx, enc, latch, value, $tag)

Abstract
Send a command to the PC that a latch has been detected.

Input Parameters

Parameter	Description
blt.idx	Index of the belt
enc	Encoder channel
latch	Latch number that was triggered
value	Latched encoder value defining the location
$tag	Tag to associate with instances created

Output Parameters

None

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use or refer to the Process Manager documentation under the Belt Monitoring section.

Note that if the user passes in a $tag, all instances created relative to that latch event will be assigned the specified tag.

pm.ace.blt.pos(blt.idx, encoder, encoder.value, encoder.vel)

Abstract

Send the current belt position to the PC.

Input Parameters

Parameter	Description
blt.idx	Index of the task that owns the belt being updated
encoder	Index of the encoder being updated
encoder.value	The encoder position
encoder.vel	The encoder velocity

Output Parameters

None

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use.

pm.ace.blt.sp(blt.idx, $spacing, value, $tag)

Abstract

Indicate to ace that a part/target belt spacing event has occurred.

Input Parameters

Parameter	Description
blt.idx	Index of the task that owns the belt being updated
$spacing	Name of the part/target to which the spacing is relative
value	Encoder value defining the location
$tag	Tag to associate with instances created

Output Parameters

None

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use or refer to the Process Manager documentation under the Belt Monitoring section.

Note that if the user passes in a $tag, all instances created relative to that spacing event will be assigned the specified tag.

pm.ace.pkstate($manager, state, time.out, sts)

Abstract

Change the run state of a process manager. This program is called by the startup program to start/stop a process manager application.

Input Parameters

Parameter	Description
$manager	Process manager name
state	The desired state where True = Start False = Stop
time.out	Time (in s) which rm.execute is allowed to run

Output Parameters

Parameter	Description
sts	Status of the operation.

pm.ace.ref(tsk.idx, $ref.name, sts)

Abstract

Sends a refinement request to the PC.

Input Parameters

Parameter	Description
tsk.idx	Index of the task requiring refinement
$ref.name	Name of the refinement operation

Output Parameters

Parameter	Description
sts	Status of the operation.

Example

This program is always used in the context of a refinement program. Create a custom refinement program to see an example.

pm.ace.ref.wt(tsk.idx, time.out, loc, sts)

Abstract

Wait for the refinement operation to complete.

Input Parameters

Parameter	Description
tsk.idx	Index of the task requiring refinement
time.out	Maximum amount of time to wait for the operation.

Output Parameters

Parameter	Description
sts	Status of the operation.

Example

This program is always used in the context of a refinement program. Create a custom refinement program to see an example.

pm.blt.convert(encoder)

Abstract

Convert a real world reference location to an encoder value in range between min/max.

Input Parameters

Parameter	Description
encoder	The encoder reference to convert

Output Parameters

Parameter	Description
encoder	The converted position

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use.

pm.blt.travel(point.1, point.2, distance)

Abstract

Determine the distance the belt has traveled between 2 encoder values.

Input Parameters

Parameter	Description
point.1	The first encoder position
point.2	The second encoder position

Output Parameters

Parameter	Description
distance	The distance between the 2 positions

Example

This program is always used in the context of a belt monitoring program. Create a custom belt program to see this in use.

pm.chk.run(run)

Abstract

Check to see if a process application is running.

Input Parameters

None

Output Parameters

Parameter	Description
run	Checks to see if a process manager application is running.

pm.chk.stat(run)

Abstract

Determine if both the process manager application is running and the ACE server is running. Process manager tasks uses this method to check if they should continue running.

Input Parameters

None

Output Parameters

Parameter	Description
run	True - ACE and a process manager application are running False - ACE and/or the process manager application are not running

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

  AUTO REAL tsk.idx, is.reset, , is.running, sts
  AUTO REAL part.idx, target.idx

  tsk.idx = TASK()

  ; Initialize the robot
  CALL pm.rob.init(tsk.idx)
  CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

  ; Get the part and target indexes
  CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
  CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

  ; Control loop
  DO

    ; Perform a pick
    CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

    ; Perform a place
    CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

    ; Indicate we are entering idle mode
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Move to the current destination to ensure
    ; we are no longer tracking a belt.
    CALL pm.mv.dest(tsk.idx)

    ; Wait for a period of time
    WAIT.EVENT , 3

    ; Check to see if any errors occured while we were delaying
    CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
    IF (sts <> pm.tsk.success) THEN
      CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
    END

    ; Indicate we are leaving idle mode
    CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

    ; Check to see if the process manager is still running
    CALL pm.chk.stat(is.running)

  UNTIL is.running == FALSE

  CALL pm.rob.clear(tsk.idx, sts)
  RETURN
.END
```

pm.chk.tskerr(tsk.idx, resp.mask, code, resp)

Abstract
Check for error conditions and display an error if one exists on the specified task.

Input Parameters

Parameter	Description
tsk.idx	Index of the task.
resp.mask	Possible responses to the error.

Output Parameters

Parameter	Description
code	Detected error code, or 0 if no error detected
resp	Response to the error

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

  AUTO REAL tsk.idx, is.reset, , is.running, sts
  AUTO REAL part.idx, target.idx

  tsk.idx = TASK()

  ; Initialize the robot
  CALL pm.rob.init(tsk.idx)
  CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

  ; Get the part and target indexes
  CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
  CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

  ; Control loop
  DO

    ; Perform a pick
    CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

    ; Perform a place
    CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

    ; Indicate we are entering idle mode
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Move to the current destination to ensure
    ; we are no longer tracking a belt.
    CALL pm.mv.dest(tsk.idx)
```

```
        ; Wait for a period of time
        WAIT.EVENT , 3

        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END

        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.def.btn.txt(tsk.idx, button, $text)

Abstract

Define the text associated with a response button for the next error reported on a given task. This method must be called before calling pm.error.

Input Parameters

Parameter	Description
tsk.idx	Index of the task.
button	Response code associated with the button
$text	Text associated with the button. "" will clear the text.

Output Parameters

None

pm.def.err.txt(tsk.idx, $text)

Abstract

Define a custom text message associated with an error for the next error reported on the specified task index. This method must be called before calling pm.error.

Input Parameters

Parameter	Description
tsk.idx	Index of the task.
$text	Text associated with the error. "" will clear the text.

Output Parameters

None

pm.def.err.txt(tsk.idx, $text)

Abstract
Define a custom text message associated with an error for the next error reported on the specified task index. This method must be called before calling pm.error.

Input Parameters

Parameter	Description
tsk.idx	Index of the task.
$text	Text associated with the error. "" will clear the text.

Output Parameters

None

pm.fdr.state($name, state)

Abstract
Updates the state of a feeder on the PC.

Input Parameters

Parameter	Description
$name	Name of the feeder part or target
state	State of the feeder: pm.fst.avail Is available pm.fst.busy Is not available pm.fst.cycle Is cycling pm.fst.hshk Waiting for handshake

Output Parameters

None

pm.get.free.tsk(first.tsk, last.tsk, tsk.num)

Abstract
Selects an available task that can be used Tasks from first.tsk to last.tsk are scanned and the first available one is returned

Input Parameters

Parameter	Description
first.task	The task number at which to start scanning
last.task	The task number at which we want to stop scanning

Output Parameters

Parameter	Description
task.num	The task number. If value is -1, no available task was found

pm.gr.clroff(tsk.idx, sts)

Abstract

Clears all refinement offsets associated with the gripper.

Input Parameters

Parameter	Description
tsk.idx	Index of the task

Output Parameters

Parameter	Description
sts	Status of the operation

pm.gr.release(tsk.idx, tip.idx, state, sts)

Abstract

Turn on/off the gripper release signals.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access -1 = All tips 0...N = The tip to operate with
state	State of the gripper

Output Parameters

Parameter	Description
sts	Status of the operation

pm.gr.select.ck(tsk.idx, tip.idx, resp)

Abstract
Check for any gripper selection errors.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access

Output Parameters

Parameter	Description
resp	User response to an error

pm.gr.select(tsk.idx, tip.idx)

Abstract
Perform a tip selection, extending or retracting tips as needed. It also checks if a tip selection program is associated with the current robot gripper. If one is specified it will be executed.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access -1 = All tips 0...N = The tip to operate with

Output Parameters

None

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

   AUTO REAL tsk.idx, sts
   AUTO LOC tool.trans

   tsk.idx = TASK()

   ; Select the first tip
   CALL pm.gr.select(tsk.idx, 0)
```

```
    ; Select all tips
    CALL pm.gr.select(tsk.idx, -1)

    ; Unselect all tips
    CALL pm.gr.unselect(tsk.idx)

    ; Extend tip #1 and #2
    CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
    CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

    ; Set the offset associated with tip #2
    CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

    ; Get the offset associated with tip #2
    CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)

.END
```

pm.gr.setoff(tsk.idx, tip.idx, tool.trans, sts)

Abstract

Associates an offset with a specific gripper tip.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
tool.trans	Gripper tip transform

Output Parameters

Parameter	Description
sts	Status of the operation

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

    AUTO REAL tsk.idx, sts
    AUTO LOC tool.trans

    tsk.idx = TASK()

    ; Select the first tip
    CALL pm.gr.select(tsk.idx, 0)

    ; Select all tips
    CALL pm.gr.select(tsk.idx, -1)
```

```
    ; Unselect all tips
    CALL pm.gr.unselect(tsk.idx)

    ; Extend tip #1 and #2
    CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
    CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

    ; Set the offset associated with tip #2
    CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

    ; Get the offset associated with tip #2
    CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)

.END
```

pm.gr.settle(tsk.idx, tip.idx, state, sts)

Abstract

Wait for the required settling time for the gripper.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
state	Desired state of the gripper

Output Parameters

Parameter	Description
sts	Status of the operation

pm.gr.state(tsk.idx, tip.idx, state, check.inputs, sts)

Abstract

Change the state of the gripper tip for the specified robot.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access
state	Desired state of the gripper
check.inputs	Do we wait for inputs

Output Parameters

Parameter	Description
sts	Status of the operation

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

   AUTO REAL tsk.idx, sts
   AUTO LOC tool.trans

   tsk.idx = TASK()

   ; Select the first tip
   CALL pm.gr.select(tsk.idx, 0)

   ; Select all tips
   CALL pm.gr.select(tsk.idx, -1)

   ; Unselect all tips
   CALL pm.gr.unselect(tsk.idx)

   ; Extend tip #1 and #2
   CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
   CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
   CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)

.END
```

pm.gr.trans(tsk.idx, tip.idx, tool.trans, sts)

Abstract
Associates an offset with a specific gripper tip.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
tip.idx	Zero based index of the tip to access

Output Parameters

Parameter	Description

tool.trans	The gripper tip transformation
sts	Status of the operation

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

   AUTO REAL tsk.idx, sts
   AUTO LOC tool.trans

   tsk.idx = TASK()

   ; Select the first tip
   CALL pm.gr.select(tsk.idx, 0)

   ; Select all tips
   CALL pm.gr.select(tsk.idx, -1)

   ; Unselect all tips
   CALL pm.gr.unselect(tsk.idx)

   ; Extend tip #1 and #2
   CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
   CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
   CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)
.END
```

pm.gr.unselect(tsk.idx)

Abstract

Retract all the tips.

Input Parameters

Parameter	Description
tsk.idx	Index of the task

Output Parameters

None

Example

This program is always used in the context of a robot program and should always be called directly from the robot task.

```
.PROGRAM a.ex2.gripper()

   AUTO REAL tsk.idx, sts
   AUTO LOC tool.trans

   tsk.idx = TASK()

   ; Select the first tip
   CALL pm.gr.select(tsk.idx, 0)

   ; Select all tips
   CALL pm.gr.select(tsk.idx, -1)

   ; Unselect all tips
   CALL pm.gr.unselect(tsk.idx)

   ; Extend tip #1 and #2
   CALL pm.gr.state(tsk.idx, 0, TRUE, TRUE, sts)
   CALL pm.gr.state(tsk.idx, 1, TRUE, TRUE, sts)

   ; Set the offset associated with tip #2
   CALL pm.gr.setoff(tsk.idx, 0, NULL, sts)

   ; Get the offset associated with tip #2
   CALL pm.gr.trans(tsk.idx, 0, tool.trans, sts)
.END
```

pm.log($text)

Abstract

Send text associated with the current task to the AceServer log file.

Input Parameters

Parameter	Description
$text	The text to append to the AceServer log on the PC

Output Parameters

None

Additional Notes

The ***pm.log*** method should only be called on a process manager task. The program will check to see if logging is enabled before sending the log string to the PC. The variable ***pm.log*** will enable logging on all V+ process manager tasks. If you want to enable logging for an individual V+ task, the variable ***pm.tsk.log [TASK()]*** can be set to ***TRUE***.

pm.mv.attach(tsk.idx, rob.num)

Abstract
Attach the robot to the current task.

Input Parameters

Parameter	Description
tsk.idx	Index of the task
rob.num	Robot number to attach

Output Parameters

None

Additional Notes
For a robot task, the global variable ***pm.tsk.robnum[]*** saves the robot number associated with the task.

Example
```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL tsk.idx, is.reset, , is.running, sts
    AUTO REAL part.idx, target.idx

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Get the part and target indexes
    CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
    CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

    ; Control loop
    DO

        ; Perform a pick
        CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

        ; Perform a place
        CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

        ; Indicate we are entering idle mode
        CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

        ; Move to the current destination to ensure
        ; we are no longer tracking a belt.
        CALL pm.mv.dest(tsk.idx)
```

```
        ; Wait for a period of time
        WAIT.EVENT , 3

        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END

        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.mv.chkbrk(offset, pars[])

Abstract
Check for a break in motion using the specified motion parameters.

Input Parameters

Parameter	Description
offset	Starting index in the motion parameters array
pars[]	Motion parameters array

Output Parameters

None

pm.mv.dest(rob.idx)

Abstract
Move to the current destination. Used to stop robot fram tracking a part.

Input Parameters

Parameter	Description
rob.idx	Index of the robot

Output Parameters

None

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL tsk.idx, is.reset, , is.running, sts
    AUTO REAL part.idx, target.idx

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Get the part and target indexes
    CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
    CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

    ; Control loop
    DO

        ; Perform a pick
        CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

        ; Perform a place
        CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

        ; Indicate we are entering idle mode
        CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

        ; Move to the current destination to ensure
        ; we are no longer tracking a belt.
        CALL pm.mv.dest(tsk.idx)

        ; Wait for a period of time
        WAIT.EVENT , 3

        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END

        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.mv(blt.idx, position, offset, pars[])

Abstract

Perform a move to the location using the specified motion parameters.

Input Parameters

Parameter	Description
blt.idx	Belt number to move relative to 0 = Non-belt relative move > 0 = Belt relative move
position	Position to move to
offset	Starting index in the motion parameters array
pars[]	Motion parameters array

Output Parameters

None

Additional Notes

The ***pars[]*** parameter uses the following offset values when applying the motion parameters:

Variable	Usage
pm.fmv.spd	The speed percentage
pm.fmv.spdmd	The Speed Mode
pm.fmv.accel	The acceleration percentage
pm.fmv.decel	The deceleration percentage
pm.fmv.dur	The move duration
pm.fmv.scv	The S-curve profile
pm.fmv.straigh	Is straight line motion requested
pm.fmv.motend	Motion end parameter
pm.fmv.stlpct	The settle percentage
pm.fmv.break	Should it wait until motion is done
pm.fmv.single	Single/multiple mode
pm.fmv.righty	Righty/Lefty mode
pm.fmv.above	Above/below mode
pm.fmv.flip	Flip/No-flip

For a standard motion sequence, the following values are used for the ***offset*** parameter.

Variable	Usage
pm.ms.aoffset	Approach motion segment
pm.ms.poffset	Move motion segment
pm.ms.doffset	Depart motion segment

For a standard vision refinement sequence, the following values are used for the ***offset*** parameter.

Variable	Usage
pm.vrf.aoffset	Approch motion segment
pm.vrf.poffset	Move motion segment
pm.vrf.doffset	Depart motion segment

pm.mv.idle(rob.idx)

Abstract

Move a robot to the idle location. The robot first retracts the Z axis (maintaining the current XY position) to the level of the idle position. Once retracted it moves to idle.

Input Parameters

Parameter	Description
rob.idx	Index of the robot

Output Parameters

None

Example

CALL pm.mv.idle(rob.idx)

pm.proc.enable($manager, proc.idx, state, sts)

Abstract

Enable or disable a process based on the process index.

Input Parameters

Parameter	Description
$manager	Name of the process manager
proc.idx	Name of the process manager
state	State of the process True = Enable the process False = Disable the process

Output Parameters

Parameter	Description
sts	Status of the operation: 0 = success < 0 = error

pm.prt.avail(rob.idx, part.idx, count)

Abstract

Gets the number of available parts for the robot.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
part.idx	Part index to access

Output Parameters

Parameter	Description
count	Number of items available in the queue

Example

```
.PROGRAM a.ex2.qlookup()

   AUTO REAL rob.idx, part.idx, target.idx, count
   AUTO $part, $target

   ; Identify the first robot task
   rob.idx = 3

   ; Look up the queue number for a part in the workspace
   $part = "/Process/Part - Spacing 2"
   CALL pm.prt.get.idx(rob.idx, $part, part.idx)

   ; Get the number of parts available
   CALL pm.prt.avail(rob.idx, part.idx, count)

   ; Look up the queue number for a target in the workspace
   $target = "/Process/Target 2 - Static"
   CALL pm.trg.get.idx(rob.idx, $target, target.idx)

   ; Get the number of partstargets available
   CALL pm.trg.avail(rob.idx, target.idx, count)
.END
```

pm.prt.done(rob.idx, part.idx, obj.idx, sts)

Abstract

Indicate a part has been processed.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
part.idx	Part index to access
obj.idx	Index of the instance/object to mark as done
sts	Was the part processed by the robot: TRUE = part was processed FALSE = part was not processed

Output Parameters

None

pm.prt.get(rob.idx, part.idx, obj.idx, $id, position, reference, pal.idx)

Abstract

Gets a part from the buffer.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
part.idx	Part index to access
obj.idx	Index of the object/instance to extract

Output Parameters

Parameter	Description
$id	Unique identifier of the part
position	Position of the part
reference	Reference position of the part
pal.idx	Pallet index associated with the instance

Example

```
.PROGRAM a.ex2.queues()

   AUTO REAL i, rob.idx, inst.idx
   AUTO REAL reference, pal.idx, sts, time, pct
   AUTO LOC position
   AUTO $id, $source
```

```
; Identify the first robot task
rob.idx = 3

; Go through all part queues
FOR i = 0 TO pm.prt.count[rob.idx]-1

   TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

   ; Go through all instances in the queue
   inst.idx = pm.prt.rob.idx[rob.idx,i]
   WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

      CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.prt.move.ptr(inst.idx)
   END
END

; Go through all target queues
FOR i = 0 TO pm.trg.count[rob.idx]-1

   TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

   ; Go through all instances in the queue
   inst.idx = pm.trg.rob.idx[rob.idx,i]
   WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

      CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.trg.move.ptr(inst.idx)
```

```
    END
  END
.END
```

pm.prt.get.idx(rob.idx, $part, part.idx)

Abstract

Get the index of the part buffer for a given robot task.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
$part	Part name to search for

Output Parameters

Parameter	Description
part.idx	Part index in the robot part buffer. -1 = No matching part was found

Example

```
.PROGRAM a.ex2.qlookup()

  AUTO REAL rob.idx, part.idx, target.idx, count
  AUTO $part, $target

  ; Identify the first robot task
  rob.idx = 3

  ; Look up the queue number for a part in the workspace
  $part = "/Process/Part - Spacing 2"
  CALL pm.prt.get.idx(rob.idx, $part, part.idx)

  ; Get the number of parts available
  CALL pm.prt.avail(rob.idx, part.idx, count)

  ; Look up the queue number for a target in the workspace
  $target = "/Process/Target 2 - Static"
  CALL pm.trg.get.idx(rob.idx, $target, target.idx)

  ; Get the number of targets available
  CALL pm.trg.avail(rob.idx, target.idx, count)
.END
```

pm.prt.getloc(rob.idx, part.idx, obj.idx, pct)

Abstract

Get the location of a given part in the belt window.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
part.idx	Part index to access
obj.idx	Index of the object/instance to access

Output Parameters

Parameter	Description
pct	Percentage of travel along the belt

Example

```
.PROGRAM a.ex2.queues()

  AUTO REAL i, rob.idx, inst.idx
  AUTO REAL reference, pal.idx, sts, time, pct
  AUTO LOC position
  AUTO $id, $source

  ; Identify the first robot task
  rob.idx = 3

  ; Go through all part queues
  FOR i = 0 TO pm.prt.count[rob.idx]-1

    TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

    ; Go through all instances in the queue
    inst.idx = pm.prt.rob.idx[rob.idx,i]
    WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

      CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.prt.move.ptr(inst.idx)

    END
  END
```

```
   ; Go through all target queues
   FOR i = 0 TO pm.trg.count[rob.idx]-1

      TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

      ; Go through all instances in the queue
      inst.idx = pm.trg.rob.idx[rob.idx,i]
      WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

         CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
         CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
         CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)

         TYPE " ID = ", $id
         TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
         TYPE " Reference = ", reference
         TYPE " Pallet Index = ", pal.idx
         TYPE " Status = ", sts
         TYPE " Time = ", time
         TYPE " Source = ", $source
         TYPE " Location % = ", pct
         TYPE " "

         CALL pm.trg.move.ptr(inst.idx)

      END
   END
.END
```

pm.prt.move.ptr(ptr)

Abstract

Increment the pointer to the next slot in the buffer.

Input Parameters

Parameter	Description
ptr	The pointer to increment

Output Parameters

Parameter	Description
ptr	The incremented pointer

Example

```
.PROGRAM a.ex2.queues()

   AUTO REAL i, rob.idx, inst.idx
   AUTO REAL reference, pal.idx, sts, time, pct
   AUTO LOC position
   AUTO $id, $source
```

```
; Identify the first robot task
rob.idx = 3

; Go through all part queues
FOR i = 0 TO pm.prt.count[rob.idx]-1

   TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

   ; Go through all instances in the queue
   inst.idx = pm.prt.rob.idx[rob.idx,i]
   WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

      CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.prt.move.ptr(inst.idx)

   END
END

; Go through all target queues
FOR i = 0 TO pm.trg.count[rob.idx]-1

   TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

   ; Go through all instances in the queue
   inst.idx = pm.trg.rob.idx[rob.idx,i]

   WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

      CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.trg.move.ptr(inst.idx)
```

```
    END
  END
.END
```

pm.ps.error(tsk.idx, code, resp.mask, resp)

Abstract

Report an error and wait for a response.

Input Parameters

Parameter	Description
tsk.idx	Index associated with the error report.
code	Error code/status of the operation
resp.mask	Mask of available response options

Output Parameters

Parameter	Description
resp	The user response to the error

pm.ps.map.idx(proc.idx, map.idx)

Abstract

Converts a process index from the PC to a robot-relative index.

Input Parameters

Parameter	Description
proc.idx	Process index from the PC

Output Parameters

Parameter	Description
map.idx	Robot relative index for the process

Example

```
.PROGRAM cu.robot2()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

  AUTO REAL task.idx, i, sts
  AUTO REAL proc.idx.1, proc.idx.2

  tsk.idx = TASK()

  ; Initialize the robot
  CALL pm.rob.init(tsk.idx)
  CALL pm.rob.idlemd(tsk.idx, TRUE)
```

```
    ; Get the index associated with the first two processes in the process manager
    CALL pm.ps.map.idx(0, proc.idx.1)
    CALL pm.ps.map.idx(1, proc.idx.2)

    ; Control loop
    DO

        ; Execute the first process 3 time
        FOR i = 1 TO 3
            CALL pm.rob.process(tsk.idx, proc.idx.1, sts)
        END

        ; Execute the second process
        CALL pm.rob.process(tsk.idx, proc.idx.2, sts)

        ; Check process status
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    RETURN
.END
```

pm.ps.power(tsk.idx, enable)

Abstract

Change the power status of the system. If power is enabled, issue a calibration request, if needed. If power cannot be enabled, it will report an error. Because of this, this method is not intended to be used from within a custom error program. It may lead to a recursive call and a stack overflow.

Input Parameters

Parameter	Description
tsk.idx	The task index for error reporting
enable	Enable the power.

Output Parameters

None

pm.rob.chkidle(rob.idx, part.type, grip.idx, excl[], is.reset, sts)

Abstract

Check the status when the robot is idle, including checking for the robot cycle stop condition.

Input Parameters

Parameter	Description
tsk.idx	Index of the robot task
start.time	Type of part to pick

in.idle	The reference time when the robot entered idle mode
at.wait	Is the robot at the waiting position

Output Parameters

Parameter	Description
in.idle	Is the robot in idle mode
at.wait	Is the robot at the waiting position

pm.rob.clear(rob.idx)

Abstract

Move the robot to the idle position and clear the robot gripper.

Input Parameters

Parameter	Description
rob.idx	Index of the robot

Output Parameters

None

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

   AUTO REAL tsk.idx, is.reset, , is.running, sts
   AUTO REAL part.idx, target.idx

   tsk.idx = TASK()

   ; Initialize the robot
   CALL pm.rob.init(tsk.idx)
   CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

   ; Get the part and target indexes
   CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
   CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

   ; Control loop
   DO

      ; Perform a pick
      CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

      ; Perform a place
      CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

      ; Indicate we are entering idle mode
      CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")
```

```
      ; Move to the current destination to ensure
      ; we are no longer tracking a belt.
      CALL pm.mv.dest(tsk.idx)

      ; Wait for a period of time
      WAIT.EVENT , 3

      ; Check to see if any errors occured while we were delaying
      CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
      IF (sts <> pm.tsk.success) THEN
         CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
      END

      ; Indicate we are leaving idle mode
      CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

      ; Check to see if the process manager is still running
      CALL pm.chk.stat(is.running)

   UNTIL is.running == FALSE

   CALL pm.rob.clear(tsk.idx, sts)
   RETURN
.END
```

pm.rob.gettag(rob.idx, $tag)

Abstract

Returns the tag associated with the part or target instance currently being operated on by the robot. Typically, this would be called from within a custom motion sequence program to access information that was associated with a VisionTransform Tag associated with a custom vision tool or the Tag associated with a LocatedInstance set by a custom allocation script.

Input Parameters

Parameter	Description
tsk.idx	Index of the robot

Output Parameters

Parameter	Description
$tag	Tag associated with the instance being processed

Example

```
.PROGRAM cu.mv.sequence(tsk.idx, $type, blt.idx, reference, pal.idx, grip.idx, grip.state, pos, vals[], sts)

   AUTO REAL at.wait, belt.win.idx, code, distance, offset
   AUTO REAL resp, rob.num
   AUTO $cust.program, $tag
   AUTO LOC appro.pos, depart.pos, grip.trans, position
```

```
    sts = pm.tsk.success

    CALL pm.rob.gettag(tsk.idx, $tag)
    TYPE "Tag = "+$tag

; Calculate the position, factoring in the gripper transformations

    ; Initiate the gripper selection
    CALL pm.gr.select(tsk.idx, grip.idx)

    ; Get the gripper transformation
    CALL pm.gr.trans(tsk.idx, grip.idx, grip.trans, sts)
```

pm.rob.idlemd(rob.tsk, in.idle, code, $info)

Abstract

Change the idle status of the robot. This program sets the status in global memory. It expects the process strategy background program to perform the sending of the status to the PC.

Input Parameters

Parameter	Description
rob.tsk	Index of the robot
in.idle	Is the robot in idle mode? True = Robot is idle False = Robot is running
code	Error code associated with the idle mode state. If non-zero, an error will be reported
$info	Additional information to send with the error code. This is used by 'pm.err.no.inst' to send the name of the part or target the robot is waiting for.

Output Parameters

None

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL tsk.idx, is.reset, , is.running, sts
    AUTO REAL part.idx, target.idx

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")
```

```
    ; Get the part and target indexes
    CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
    CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

    ; Control loop
    DO

        ; Perform a pick
        CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

        ; Perform a place
        CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

        ; Indicate we are entering idle mode
        CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

        ; Move to the current destination to ensure
        ; we are no longer tracking a belt.
        CALL pm.mv.dest(tsk.idx)

        ; Wait for a period of time
        WAIT.EVENT , 3
        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END

        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.rob.init(rob.idx)

Abstract

Initialize the robot by attaching the robot, clearing the TOOL, moving to the idle position, and turning off all gripper signals.

Input Parameters

Parameter	Description
rob.idx	Index of the robot

Output Parameters

None

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL tsk.idx, is.reset, , is.running, sts
    AUTO REAL part.idx, target.idx

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Get the part and target indexes
    CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
    CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

    ; Control loop
    DO

        ; Perform a pick
        CALL pm.rob.pick(tsk.idx, part.idx, -1, , is.reset, sts)

        ; Perform a place
        CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, , is.reset, sts)

        ; Indicate we are entering idle mode
        CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

        ; Move to the current destination to ensure
        ; we are no longer tracking a belt.
        CALL pm.mv.dest(tsk.idx)

        ; Wait for a period of time
        WAIT.EVENT , 3

        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END

        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.rob.monque(rob.tsk, part.idx, trg.idx)

Abstract

Process the active part and targets queues and mark any out-of-range instances as done. This method should only be called on a robot task when the robot is waiting for some operation to complete. This ensures any instances that go out of range of the robot are properly released and sent back to the PC.

If the robot is currently processing a part or target, the index of the associated queue should be passed as a parameter to the method.

Input Parameters

Parameter	Description
rob.tsk	Index of the robot
part.idx	Index of a part queue to not check. -1 indicates all queues will be checked
trg.idx	Index of a target queue to not check. -1 indicates all queues will be checked

Output Parameters

None

Example

```
CALL pm.rob.monque(rob.tsk, -1, -1)
```

pm.rob.pick(rob.idx, part.type, grip.idx, excl[], is.reset, sts)

Abstract

Perform a single pick operation with the robot in the current task.

Input Parameters

Parameter	Description
rob.tsk	Index of the robot
part.idx	Type of part to pick
grip.idx	The index of the gripper tip to use -1 = All tips

ms.type	The move segment type: pm.ms.mv.norm - Normal Move pm.ms.mv.iad - Pick / place motion with intra approach and intra depart pm.ms.mv.iand - Pick / place with intra approach and normal depart pm.ms.mv.naid - Pick / place with normal approach and intra depart
excl[]	The exclusions used when picking

Output Parameters

Parameter	Description
is.reset	Was the queue reset during the operation
sts	Status code: pm.tsk.success = Successful operation pm.tsk.abort = Abort the operation

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

  AUTO REAL tsk.idx, is.reset, , is.running, sts
  AUTO REAL part.idx, target.idx

  tsk.idx = TASK()

  ; Initialize the robot
  CALL pm.rob.init(tsk.idx)
  CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

  ; Get the part and target indexes
  CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
  CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

  ; Control loop
  DO

    ; Perform a pick
    CALL pm.rob.pick(tsk.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

    ; Perform a place
    CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

    ; Indicate we are entering idle mode
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")
```

```
      ; Move to the current destination to ensure
      ; we are no longer tracking a belt.
      CALL pm.mv.dest(tsk.idx)

      ; Wait for a period of time
      WAIT.EVENT , 3

      ; Check to see if any errors occured while we were delaying
      CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
      IF (sts <> pm.tsk.success) THEN
         CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
      END

      ; Indicate we are leaving idle mode
      CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

      ; Check to see if the process manager is still running
      CALL pm.chk.stat(is.running)

   UNTIL is.running == FALSE

   CALL pm.rob.clear(tsk.idx, sts)
   RETURN
.END
```

pm.rob.place(rob.idx, target.type, part.type, grip.idx, ms.type, excl[], is.reset, sts)

Abstract

Perform a single place operation with the robot in the current task.

Input Parameters

Parameter	Description
rob.tsk	Index of the robot
target.type	The type of target to place
part.type	The type of part being placed
grip.idx	The index of the gripper tip to use -1 = All tips
ms.type	The move segment type: pm.ms.mv.norm - Normal Move pm.ms.mv.iad - Pick / place motion with intra approach and intra depart pm.ms.mv.iand - Pick / place with intra approach and normal depart pm.ms.mv.naid - Pick / place with normal approach and intra depart

excl[]	The exclusions used when picking

Output Parameters

Parameter	Description
is.reset	Was the queue reset during the operation
sts	Status code: pm.tsk.success = Successful operation pm.tsk.abort = Abort the operation

Example

```
.PROGRAM cu.robot()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL tsk.idx, is.reset, , is.running, sts
    AUTO REAL part.idx, target.idx

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

    ; Get the part and target indexes
    CALL pm.prt.get.idx(tsk.idx, "/Process/Part - Spacing 2", part.idx)
    CALL pm.trg.get.idx(tsk.idx, "/Process/Target 2 - Static", target.idx)

    ; Control loop
    DO

        ; Perform a pick
        CALL pm.rob.pick(tsk.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

        ; Perform a place
        CALL pm.rob.place(tsk.idx, target.idx, part.idx, -1, pm.ms.mv.norm, , is.reset, sts)

        ; Indicate we are entering idle mode
        CALL pm.rob.idlemd(tsk.idx, TRUE, 0, "")

        ; Move to the current destination to ensure
        ; we are no longer tracking a belt.
        CALL pm.mv.dest(tsk.idx)

        ; Wait for a period of time
        WAIT.EVENT , 3

        ; Check to see if any errors occured while we were delaying
        CALL pm.chk.tskerr(tsk.idx, pm.tsk.retry, , sts)
        IF (sts <> pm.tsk.success) THEN
            CALL pm.mv.attach(tsk.idx, pm.tsk.robnum[tsk.idx])
        END
```

```
        ; Indicate we are leaving idle mode
        CALL pm.rob.idlemd(tsk.idx, FALSE, 0, "")

        ; Check to see if the process manager is still running
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE

    CALL pm.rob.clear(tsk.idx, sts)
    RETURN
.END
```

pm.rob.process(rob.idx, proc.idx, sts)

Abstract

Converts a process index from the PC to a robot-relative index.

Input Parameters

Parameter	Description
rob.idx	Index of the robot to drive
proc.idx	Process to operate with

Output Parameters

Parameter	Description
sts	Status code pm.tsk.success = Successful operation pm.tsk.skip = Skip the operation pm.tsk.abort = Abort the operation

Example

```
.PROGRAM cu.robot2()
; ABSTRACT: Custom robot program for the Process Manager
; Process Strategy.

    AUTO REAL task.idx, i, sts
    AUTO REAL proc.idx.1, proc.idx.2

    tsk.idx = TASK()

    ; Initialize the robot
    CALL pm.rob.init(tsk.idx)
    CALL pm.rob.idlemd(tsk.idx, TRUE)

    ; Get the index associated with the first two processes in the process manager
    CALL pm.ps.map.idx(0, proc.idx.1)
    CALL pm.ps.map.idx(1, proc.idx.2)

    ; Control loop
    DO
```

```
        ; Execute the first process 3 time
        FOR i = 1 TO 3
            CALL pm.rob.process(tsk.idx, proc.idx.1, sts)
        END

        ; Execute the second process
        CALL pm.rob.process(tsk.idx, proc.idx.2, sts)

        ; Check process status
        CALL pm.chk.stat(is.running)

    UNTIL is.running == FALSE
    RETURN
.END
```

pm.rob.refine(rob.idx, refine.idx, grip.idx, sts)

Abstract

Refine the part in the robot gripper using vision.

Input Parameters

Parameter	Description
rob.tsk	Index of the robot
refine.idx	Index of the refinement station
grip.idx	The index of the gripper tip to use -1 = All tips
excl[]	The exclusions used when picking

Output Parameters

Parameter	Description
sts	Status code: pm.tsk.success = Successful operation pm.tsk.retry = Retry the operation pm.tsk.abort = Abort the operation

pm.rob.settag(tsk.idx, $tag)

Abstract

Sets the tag associated with the part or target instance currently being operated on by the robot. Typically, this would be called from within a custom motion sequence program to set a string for the Tag associated with a LocatedInstance set by a custom allocation script.

Input Parameters

Parameter	Description
tsk.tsk	Index of the robot

$tag	Tag associated with the instance

Output Parameters

None

Example

```
.PROGRAM cu.mv.sequence(tsk.idx, $type, blt.idx, reference, pal.idx, grip.idx, grip.state, pos, vals[], sts)

   AUTO REAL at.wait, belt.win.idx, code, distance, offset
   AUTO REAL resp, rob.num
   AUTO $cust.program, $tag
   AUTO LOC appro.pos, depart.pos, grip.trans, position

   sts = pm.tsk.success
   $tag = "My Tag String"
   CALL pm.rob.settag(tsk.idx, $tag)

; Calculate the position, factoring in the gripper transformations
   ; Initiate the gripper selection
   CALL pm.gr.select(tsk.idx, grip.idx)

   ; Get the gripper transformation
   CALL pm.gr.trans(tsk.idx, grip.idx, grip.trans, sts)
```

pm.sig.check(sig, time, is.valid)

Abstract

Checks for the status of a digital signal ensuring the state is valid for the specified amount of time. This method does not block execution.

Input Parameters

Parameter	Description
sig	Signal to be checked
time	Amount of time the signal is expected to be in the state

Output Parameters

Parameter	Description
is.valid	Is the signal in the specified state

Example

```
.PROGRAM a.ex2.debounce()

   AUTO REAL sig, is.on, debounce.time

   ; Define the signal and the amount of time we want the signal
   ; to remain on before we recognize it is on
```

```
   sig = 2100
   debounce.time = 2.5
   is.on = FALSE

   ; Clear the tracking structures for the signal
   CALL pm.sig.clear(sig)

   ; Trun the signal on
   SIGNAL sig
   TYPE $TIME()

   ; Wait until the signal has been on for the required amount of time
   WHILE (is.on == FALSE) DO
      CALL pm.sig.check(sig, debounce.time, is.on)
   END

   TYPE $TIME()
   SIGNAL -sig
.END
```

pm.sig.clear(sig)

Abstract

Clears the variables used to track the status of a signal.

Input Parameters

Parameter	Description
sig	Signal to clear

Output Parameters

None

Example

```
.PROGRAM a.ex2.debounce()

   AUTO REAL sig, is.on, debounce.time

   ; Define the signal and the amount of time we want the signal
   ; to remain on before we recognize it is on
   sig = 2100
   debounce.time = 2.5
   is.on = FALSE

   ; Clear the tracking structures for the signal
   CALL pm.sig.clear(sig)

   ; Trun the signal on
   SIGNAL sig
   TYPE $TIME()

   ; Wait until the signal has been on for the required amount of time
   WHILE (is.on == FALSE) DO
```

```
      CALL pm.sig.check(sig, debounce.time, is.on)
   END

   TYPE $TIME()
   SIGNAL -sig
.END
```

pm.src.clear($manager, $source, sts)

Abstract
Clear the specified source

Input Parameters

Parameter	Description
$manager	Name of the process manager
$source	The name of the source to clear

Output Parameters

Parameter	Description
sts	Status of operation

Example

```
.PROGRAM a.ex2.srcclr()

   AUTO $manager
   AUTO REAL sts

   $manager = "/Process/PackXpert Process Manager"
   $source = "Belt Handler: /Process/Belt"

   ; Clear the source with the specified name
   CALL pm.src.clear($manager, $source, sts)

   ; Clear all source
   CALL pm.src.clearall($manager, sts)
.END
```

pm.src.clearall($manager, sts)

Abstract
Clears all sources for the specified process manager

Input Parameters

Parameter	Description
$source	The name of the source to clear

Output Parameters

Parameter	Description
sts	Status of operation

Example

```
.PROGRAM a.ex2.srcclr()

   AUTO $manager
   AUTO REAL sts

   $manager = "/Process/PackXpert Process Manager"
   $source = "Belt Handler: /Process/Belt"

   ; Clear the source with the specified name
   CALL pm.src.clear($manager, $source, sts)

   ; Clear all source
   CALL pm.src.clearall($manager, sts)
.END
```

pm.trg.avail(rob.idx, trg.idx, count)

Abstract

Gets the number of available targets for the robot.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
trg.idx	Target index to access

Output Parameters

Parameter	Description
count	Number of items available in the queue

Example

```
.PROGRAM a.ex2.qlookup()

   AUTO REAL rob.idx, part.idx, target.idx, count
   AUTO $part, $target

   ; Identify the first robot task
   rob.idx = 3

   ; Look up the queue number for a part in the workspace
   $part = "/Process/Part - Spacing 2"
   CALL pm.prt.get.idx(rob.idx, $part, part.idx)
```

```
    ; Get the number of parts available
    CALL pm.prt.avail(rob.idx, part.idx, count)

    ; Look up the queue number for a target in the workspace
    $target = "/Process/Target 2 - Static"
    CALL pm.trg.get.idx(rob.idx, $target, target.idx)

    ; Get the number of partstargets available
    CALL pm.trg.avail(rob.idx, target.idx, count)
.END
```

pm.trg.done(rob.idx, trg.idx, obj.idx, sts)

Abstract

Indicate a target has been processed.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
trg.idx	Target index to access
obj.idx	Index of the instance/object to mark as done
sts	Was the instance processed by the robot: TRUE = part was processed FALSE = part was not processed

Output Parameters

None

pm.trg.get(rob.idx, trg.idx, obj.idx, $id, position, reference, pal.idx)

Abstract

Gets a target from the buffer.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
trg.idx	Target index to access
obj.idx	Index of the object/instance to extract

Output Parameters

Parameter	Description
$id	Unique identifier of the instance

position	Position of the instance
reference	Reference position of the instance
pal.idx	Pallet index associated with the instance

Example

```
.PROGRAM a.ex2.queues()

  AUTO REAL i, rob.idx, inst.idx
  AUTO REAL reference, pal.idx, sts, time, pct
  AUTO LOC position
  AUTO $id, $source

  ; Identify the first robot task
  rob.idx = 3

  ; Go through all part queues
  FOR i = 0 TO pm.prt.count[rob.idx]-1

    TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

    ; Go through all instances in the queue
    inst.idx = pm.prt.rob.idx[rob.idx,i]
    WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

      CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

      TYPE " ID = ", $id
      TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
      TYPE " Reference = ", reference
      TYPE " Pallet Index = ", pal.idx
      TYPE " Status = ", sts
      TYPE " Time = ", time
      TYPE " Source = ", $source
      TYPE " Location % = ", pct
      TYPE " "

      CALL pm.prt.move.ptr(inst.idx)
    END
  END

  ; Go through all target queues
  FOR i = 0 TO pm.trg.count[rob.idx]-1

    TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

    ; Go through all instances in the queue
    inst.idx = pm.trg.rob.idx[rob.idx,i]
    WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

      CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
      CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
      CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)
```

```
        TYPE " ID = ", $id
        TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
        TYPE " Reference = ", reference
        TYPE " Pallet Index = ", pal.idx
        TYPE " Status = ", sts
        TYPE " Time = ", time
        TYPE " Source = ", $source
        TYPE " Location % = ", pct
        TYPE " "

        CALL pm.trg.move.ptr(inst.idx)
      END
   END
.END
```

pm.trg.get.idx(rob.idx, $target, target.idx)

Abstract

Get the index of the target buffer for a given robot task.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
$target	Target name to search for

Output Parameters

Parameter	Description
target.idx	Target index in the robot target buffer. -1 = No matching target was found

Example

```
.PROGRAM a.ex2.qlookup()

   AUTO REAL rob.idx, part.idx, target.idx, count
   AUTO $part, $target

   ; Identify the first robot task
   rob.idx = 3

   ; Look up the queue number for a part in the workspace
   $part = "/Process/Part - Spacing 2"
   CALL pm.prt.get.idx(rob.idx, $part, part.idx)

   ; Get the number of parts available
   CALL pm.prt.avail(rob.idx, part.idx, count)

   ; Look up the queue number for a target in the workspace
   $target = "/Process/Target 2 - Static"
   CALL pm.trg.get.idx(rob.idx, $target, target.idx)
```

```
    ; Get the number of targets available
    CALL pm.trg.avail(rob.idx, target.idx, count)
.END
```

pm.trg.getloc(rob.idx, trg.idx, obj.idx, pct)

Abstract

Get the location of a given target in the belt window.

Input Parameters

Parameter	Description
rob.idx	Index of the robot task
trg.idx	Target index to access
obj.idx	Index of the object/instance to access

Output Parameters

Parameter	Description
pct	Percentage of travel along the belt

Example

```
.PROGRAM a.ex2.queues()

    AUTO REAL i, rob.idx, inst.idx
    AUTO REAL reference, pal.idx, sts, time, pct
    AUTO LOC position
    AUTO $id, $source

    ; Identify the first robot task
    rob.idx = 3

    ; Go through all part queues
    FOR i = 0 TO pm.prt.count[rob.idx]-1

        TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

        ; Go through all instances in the queue
        inst.idx = pm.prt.rob.idx[rob.idx,i]
        WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

            CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
            CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
            CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

            TYPE " ID = ", $id
            TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
            TYPE " Reference = ", reference
            TYPE " Pallet Index = ", pal.idx
            TYPE " Status = ", sts
            TYPE " Time = ", time
            TYPE " Source = ", $source
```

```
        TYPE " Location % = ", pct
        TYPE " "

        CALL pm.prt.move.ptr(inst.idx)
      END
    END

    ; Go through all target queues
    FOR i = 0 TO pm.trg.count[rob.idx]-1

      TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

      ; Go through all instances in the queue
      inst.idx = pm.trg.rob.idx[rob.idx,i]

      WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

        CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
        CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
        CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)

        TYPE " ID = ", $id
        TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
        TYPE " Reference = ", reference
        TYPE " Pallet Index = ", pal.idx
        TYPE " Status = ", sts
        TYPE " Time = ", time
        TYPE " Source = ", $source
        TYPE " Location % = ", pct
        TYPE " "

        CALL pm.trg.move.ptr(inst.idx)
      END
    END
.END
```

pm.trg.move.ptr(ptr)

Abstract
Increment the pointer to the next slot in the buffer.

Input Parameters

Parameter	Description
ptr	The pointer to increment

Output Parameters

Parameter	Description
ptr	The incremented pointer

Example

```
.PROGRAM a.ex2.queues()
```

```
AUTO REAL i, rob.idx, inst.idx
AUTO REAL reference, pal.idx, sts, time, pct
AUTO LOC position
AUTO $id, $source

; Identify the first robot task
rob.idx = 3

; Go through all part queues
FOR i = 0 TO pm.prt.count[rob.idx]-1

    TYPE "Part Queue Name: ", $pm.prt.type[rob.idx,i]

    ; Go through all instances in the queue
    inst.idx = pm.prt.rob.idx[rob.idx,i]
    WHILE (inst.idx <> pm.prt.pc.idx[rob.idx,i]) DO

        CALL pm.prt.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
        CALL pm.prt.get.stat(rob.idx, i, inst.idx, sts, time, $source)
        CALL pm.prt.getloc(rob.idx, i, inst.idx, pct)

        TYPE " ID = ", $id
        TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
        TYPE " Reference = ", reference
        TYPE " Pallet Index = ", pal.idx
        TYPE " Status = ", sts
        TYPE " Time = ", time
        TYPE " Source = ", $source
        TYPE " Location % = ", pct
        TYPE " "

        CALL pm.prt.move.ptr(inst.idx)
    END
END

; Go through all target queues
FOR i = 0 TO pm.trg.count[rob.idx]-1

    TYPE "Target Queue Name: ", $pm.trg.type[rob.idx,i]

    ; Go through all instances in the queue
    inst.idx = pm.trg.rob.idx[rob.idx,i]
    WHILE (inst.idx <> pm.trg.pc.idx[rob.idx,i]) DO

        CALL pm.trg.get(rob.idx, i, inst.idx, $id, position, reference, pal.idx)
        CALL pm.trg.get.stat(rob.idx, i, inst.idx, sts, time, $source)
        CALL pm.trg.getloc(rob.idx, i, inst.idx, pct)

        TYPE " ID = ", $id
        TYPE " Position = ", DX(position), ", ", DY(position), ", ", DZ(position)
        TYPE " Reference = ", reference
        TYPE " Pallet Index = ", pal.idx
        TYPE " Status = ", sts
        TYPE " Time = ", time
        TYPE " Source = ", $source
        TYPE " Location % = ", pct
        TYPE " "
```

```
         CALL pm.trg.move.ptr(inst.idx)
      END
   END
.END
```

V+ Remote Library Documentation

The V+ remote library allows a V+ application to access functionality on the PC. These programs are a subset of the Ace Server program module.

Remote Library Error Codes

If a problem is encountered when executing a remote library method, the following error codes can be returned:

Error Code	Description
-1000	Unable to find the AceObject referenced in the remote command
-1001	A general error has occurred while processing the remote command. See the AceServer event log for specific details.
-1002	The method referenced in a remote library execute command could not be found
-1003	A general error has occurred while processing a remote execute invocation or property access. See the AceServer event log for specific details.
-10000	Server has stopped executing and the connection to the remote PC has been lost
-10001	The request has timed out
-10002	The referenced object does not exist
-10003	The field referenced in the remote object does not exist
-10004	The type of the referenced field is not compatible with the requested operation

rm.app.event(value, wait.time, status)

Abstract

Generates an application event on the PC associated with this controller.

Input Parameters

Parameter	Description
value	Numeric code representing the event.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.appevt()

    AUTO REAL status

    CALL rm.app.event(101, 1, status)
    IF (status < 0) THEN
        TYPE "Unable To Send Event Code: ", status
        PAUSE
    END
.END
```

rm.chk.server(is.alive)

Abstract

Check to see if the ACE server is running

Input Parameters

None

Output Parameters

Parameter	Description
is.alive	Is the server alive

Example

```
.PROGRAM ex1.rm.ex()

    AUTO REAL is.alive
    AUTO REAL args[0], status
    AUTO $object, $method

    ; Check to see if communications with the PC is active
    CALL rm.chk.server(is.alive)
    IF (is.alive == FALSE) THEN
        TYPE "Not Communicating"
        PAUSE
    END

    ; Execute a script on the server and wait for 3 seconds for it to complete
    $object = "/C# Program"
    $method = "Execute"
    CALL rm.execute($object, $method, 0, $args[], 3, status)
    IF (status < 0) THEN
        TYPE "Problem executing script: ", status
```

```
    PAUSE
  END

.END
```

rm.execute($object, $method, num.args, $args[], wait.time, status)

Abstract

Execute a method on the PC and optionally wait until it has completed. The number of arguments must match the arguments defined by the method signature of the specified method. Additionally, the arguments must be convertable to the type expected in the method signature using a standard TypeConverter on the PC.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$method	The name of the method to execute
num.args	The number of arguments to send
$args[]	The arguments to pass
wait.time	The amount of time to wait for the execution to complete. Units are in seconds. A value less than or equal to 0 indicates no waiting.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.ex()

  AUTO REAL is.alive
  AUTO REAL status
  AUTO $object, $method, $args[0]

  ; Check to see if communications with the PC is active
  CALL rm.chk.server(is.alive)
  IF (is.alive == FALSE) THEN
    TYPE "Not Communicating"
    PAUSE
  END

  ; Execute a script on the server and wait for 3 seconds for it to complete
  $object = "/C# Program"
  $method = "Execute"
  CALL rm.execute($object, $method, 0, $args[], 3, status)
  IF (status < 0) THEN
```

```
      TYPE "Problem executing script: ", status
      PAUSE
   END

.END
```

rm.pc.fapp($file, $line, wait.time, status)

Abstract

Append a single line to a file on the PC

Input Parameters

Parameter	Description
$file	The name of the file to append to on the PC.
$line	The line to append to the file.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

   AUTO REAL i, num.items, status
   AUTO $file, $to.file, $lines[3]
   AUTO $file.spec
   LOCAL REAL types[]
   LOCAL $items[]

   $file = "C:\log.txt"

   ; Write 3 lines in a file on the PC
   FOR i = 1 TO 3
      CALL rm.pc.fapp($file, $TIME(), 1, status)
      IF (status < 0) THEN
         TYPE "Failure to append line on file: ", status
      END
   END

   ; Append multiple lines in 1 call to the same file
   $lines[0] = "Line 1"
   $lines[1] = "Line 2"
   $lines[2] = "Line 3"
   $lines[3] = "Line 4"
   CALL rm.pc.fapps($file, 4, $lines[], 1, status)
   IF (status < 0) THEN
      TYPE "Failure to append line on file: ", status
   END
```

```
    ; Append multiple lines in 1 call specifying the start index
    CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Copy the file from the PC to the controller
    $to.file = "DISK>D:\log.txt"
    CALL rm.pc.fcopy($file, $to.file, 3, status)
    IF (status < 0) THEN
        TYPE "Unable to copy the file to the controller: ", status
    END

    ; List all files on the PC
    $file.spec = "C:\*.txt"
    CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
    IF (status < 0) THEN
        TYPE "Unable to delete the file from the PC: ", status
    END

    FOR i = 1 TO num.items

        TYPE $items[i-1]

        IF (types[i-1] == 0) THEN
            TYPE " Is a file"
        ELSE
            TYPE " Is a directory"
        END
    END

    ; Delete the file on the PC
    CALL rm.pc.fdel($file, 3, status)
    IF (status < 0) THEN
        TYPE "Unable to delete the file from the PC: ", status
    END

.END
```

rm.pc.fapps($file, num.lines, $lines[], wait.time, status)

Abstract

Append an array of lines to a file on the PC.

Input Parameters

Parameter	Description
$file	The name of the file to append to on the PC.
num.lines	The number of lines in the array to copy.
$lines[]	The lines to append to the file, starting at index 0.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

   AUTO REAL i, num.items, status
   AUTO $file, $to.file, $lines[3]
   AUTO $file.spec
   LOCAL REAL types[]
   LOCAL $items[]

   $file = "C:\log.txt"

   ; Write 3 lines in a file on the PC
   FOR i = 1 TO 3

      CALL rm.pc.fapp($file, $TIME(), 1, status)
      IF (status < 0) THEN
         TYPE "Failure to append line on file: ", status
      END
   END

   ; Append multiple lines in 1 call to the same file
   $lines[0] = "Line 1"
   $lines[1] = "Line 2"
   $lines[2] = "Line 3"
   $lines[3] = "Line 4"
   CALL rm.pc.fapps($file, 4, $lines[], 1, status)
   IF (status < 0) THEN
      TYPE "Failure to append line on file: ", status
   END

   ; Append multiple lines in 1 call specifying the start index
   CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
   IF (status < 0) THEN
      TYPE "Failure to append line on file: ", status
   END

   ; Copy the file from the PC to the controller
   $to.file = "DISK>D:\log.txt"
   CALL rm.pc.fcopy($file, $to.file, 3, status)
   IF (status < 0) THEN
      TYPE "Unable to copy the file to the controller: ", status
   END

   ; List all files on the PC
   $file.spec = "C:\*.txt"
   CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
   IF (status < 0) THEN
      TYPE "Unable to delete the file from the PC: ", status
   END
```

```
   FOR i = 1 TO num.items

      TYPE $items[i-1]

      IF (types[i-1] == 0) THEN
         TYPE " Is a file"
      ELSE
         TYPE " Is a directory"
      END
   END

   ; Delete the file on the PC
   CALL rm.pc.fdel($file, 3, status)
   IF (status < 0) THEN
      TYPE "Unable to delete the file from the PC: ", status
   END
.END
```

rm.pc.fapps2($file, start.idx, num.lines, $lines[], wait.time, status)

Abstract

Append an array of lines to a file on the PC.

Input Parameters

Parameter	Description
$file	The name of the file to append to on the PC.
start.idx	The starting index in the array
num.lines	The number of lines in the array to copy.
$lines[]	The lines to append to the file, starting at the start.index.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

   AUTO REAL i, num.items, status
   AUTO $file, $to.file, $lines[3]
   AUTO $file.spec
   LOCAL REAL types[]
   LOCAL $items[]

   $file = "C:\log.txt"
   ; Write 3 lines in a file on the PC
   FOR i = 1 TO 3
      CALL rm.pc.fapp($file, $TIME(), 1, status)
      IF (status < 0) THEN
```

```
      TYPE "Failure to append line on file: ", status
    END
  END

  ; Append multiple lines in 1 call to the same file
  $lines[0] = "Line 1"
  $lines[1] = "Line 2"
  $lines[2] = "Line 3"
  $lines[3] = "Line 4"
  CALL rm.pc.fapps($file, 4, $lines[], 1, status)
  IF (status < 0) THEN
    TYPE "Failure to append line on file: ", status
  END

  ; Append multiple lines in 1 call specifying the start index
  CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
  IF (status < 0) THEN
    TYPE "Failure to append line on file: ", status
  END

  ; Copy the file from the PC to the controller
  $to.file = "DISK>D:\log.txt"
  CALL rm.pc.fcopy($file, $to.file, 3, status)
  IF (status < 0) THEN
    TYPE "Unable to copy the file to the controller: ", status
  END

  ; List all files on the PC
  $file.spec = "C:\*.txt"
  CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
  IF (status < 0) THEN
    TYPE "Unable to delete the file from the PC: ", status
  END

  FOR i = 1 TO num.items
    TYPE $items[i-1]
    IF (types[i-1] == 0) THEN
      TYPE " Is a file"
    ELSE
      TYPE " Is a directory"
    END
  END

  ; Delete the file on the PC
  CALL rm.pc.fdel($file, 3, status)
  IF (status < 0) THEN
    TYPE "Unable to delete the file from the PC: ", status
  END
.END
```

rm.pc.fcopy($from.file, $to.file, wait.time, status)

Abstract

Copy a file between the controller and the PC.

When specifying the file, one of the files must start with the prefix of 'DISK>'. This is used to determine the direction of the file copy. If neither of the files starts with 'DISK', then it is considered a file copy from 1 PC file to another entirely on the PC.

Input Parameters

Parameter	Description
$from.file	The file to copy from.
$to.file	The file to copy to.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

    AUTO REAL i, num.items, status
    AUTO $file, $to.file, $lines[3]
    AUTO $file.spec
    LOCAL REAL types[]
    LOCAL $items[]

    $file = "C:\log.txt"

    ; Write 3 lines in a file on the PC
    FOR i = 1 TO 3
        CALL rm.pc.fapp($file, $TIME(), 1, status)
        IF (status < 0) THEN
            TYPE "Failure to append line on file: ", status
        END
    END

    ; Append multiple lines in 1 call to the same file
    $lines[0] = "Line 1"
    $lines[1] = "Line 2"
    $lines[2] = "Line 3"
    $lines[3] = "Line 4"
    CALL rm.pc.fapps($file, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Append multiple lines in 1 call specifying the start index
    CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Copy the file from the PC to the controller
    $to.file = "DISK>D:\log.txt"
    CALL rm.pc.fcopy($file, $to.file, 3, status)
    IF (status < 0) THEN
```

```
   TYPE "Unable to copy the file to the controller: ", status
END

; List all files on the PC
$file.spec = "C:\*.txt"
CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
IF (status < 0) THEN
   TYPE "Unable to delete the file from the PC: ", status
END

FOR i = 1 TO num.items
   TYPE $items[i-1]
   IF (types[i-1] == 0) THEN
      TYPE " Is a file"
   ELSE
      TYPE " Is a directory"
   END
END

; Delete the file on the PC
CALL rm.pc.fdel($file, 3, status)
IF (status < 0) THEN
   TYPE "Unable to delete the file from the PC: ", status
END

.END
```

rm.pc.fdel($file, wait.time, status)

Abstract
Delete a file on the PC.

Input Parameters

Parameter	Description
$file	name of the file to delete on the PC
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

   AUTO REAL i, num.items, status
   AUTO $file, $to.file, $lines[3]
   AUTO $file.spec
   LOCAL REAL types[]
   LOCAL $items[]
```

```
    $file = "C:\log.txt"

    ; Write 3 lines in a file on the PC
    FOR i = 1 TO 3
        CALL rm.pc.fapp($file, $TIME(), 1, status)
        IF (status < 0) THEN
            TYPE "Failure to append line on file: ", status
        END
    END

    ; Append multiple lines in 1 call to the same file
    $lines[0] = "Line 1"
    $lines[1] = "Line 2"
    $lines[2] = "Line 3"
    $lines[3] = "Line 4"
    CALL rm.pc.fapps($file, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Append multiple lines in 1 call specifying the start index
    CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Copy the file from the PC to the controller
    $to.file = "DISK>D:\log.txt"
    CALL rm.pc.fcopy($file, $to.file, 3, status)
    IF (status < 0) THEN
        TYPE "Unable to copy the file to the controller: ", status
    END

    ; List all files on the PC
    $file.spec = "C:\*.txt"
    CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
    IF (status < 0) THEN
        TYPE "Unable to delete the file from the PC: ", status
    END

    FOR i = 1 TO num.items
        TYPE $items[i-1]
        IF (types[i-1] == 0) THEN
            TYPE " Is a file"
        ELSE
            TYPE " Is a directory"
        END
    END

    ; Delete the file on the PC
    CALL rm.pc.fdel($file, 3, status)
    IF (status < 0) THEN
        TYPE "Unable to delete the file from the PC: ", status
    END
.END
```

rm.pc.fdir($file.spec, wait.time, num.items, $items[], types[], status)

Abstract

Perform a file and directory listing on the PC.

Input Parameters

Parameter	Description
$file.spec	The file specification to match.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
num.items	The number of items returned
$items[]	The listing of files and directories found
types[]	The types of the items found: 0 = File, 1 = Directory
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.pc()

    AUTO REAL i, num.items, status
    AUTO $file, $to.file, $lines[3]
    AUTO $file.spec
    LOCAL REAL types[]
    LOCAL $items[]

    $file = "C:\log.txt"

    ; Write 3 lines in a file on the PC
    FOR i = 1 TO 3
        CALL rm.pc.fapp($file, $TIME(), 1, status)
        IF (status < 0) THEN
            TYPE "Failure to append line on file: ", status
        END
    END

    ; Append multiple lines in 1 call to the same file
    $lines[0] = "Line 1"
    $lines[1] = "Line 2"
    $lines[2] = "Line 3"
    $lines[3] = "Line 4"
    CALL rm.pc.fapps($file, 4, $lines[], 1, status)
    IF (status < 0) THEN
        TYPE "Failure to append line on file: ", status
    END

    ; Append multiple lines in 1 call specifying the start index
    CALL rm.pc.fapps2($file, 0, 4, $lines[], 1, status)
    IF (status < 0) THEN
```

```
    TYPE "Failure to append line on file: ", status
  END

  ; Copy the file from the PC to the controller
  $to.file = "DISK>D:\log.txt"
  CALL rm.pc.fcopy($file, $to.file, 3, status)
  IF (status < 0) THEN
    TYPE "Unable to copy the file to the controller: ", status
  END

  ; List all files on the PC
  $file.spec = "C:\*.txt"
  CALL rm.pc.fdir($file.spec, 3, num.items, $items[], types[], status)
  IF (status < 0) THEN
    TYPE "Unable to delete the file from the PC: ", status
  END

  FOR i = 1 TO num.items
    TYPE $items[i-1]
    IF (types[i-1] == 0) THEN
      TYPE " Is a file"
    ELSE
      TYPE " Is a directory"
    END
  END

  ; Delete the file on the PC
  CALL rm.pc.fdel($file, 3, status)
  IF (status < 0) THEN
    TYPE "Unable to delete the file from the PC: ", status
  END

.END
```

rm.pc.log($source, $text, wait.time, status)

Abstract
Append text to the AceServer log on the PC.

Input Parameters

Parameter	Description
$source	The source generating the log message.
$text	The text to append to the log.
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.log()

   AUTO REAL status

   CALL rm.pc.log("Robot 1", "Message To Log", 1, status)
   IF (status < 0) THEN
      TYPE "Failure to append to AceServer log: ", status
   END
.END
```

rm.read.anums($objects[], $variables[], wait.time, values[], status)

Abstract

Read a list of numeric property variables associated with multiple objects on the PC. The variable referenced must all be numeric types.

Input Parameters

Parameter	Description
$objects[]	The name of the remote objects to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The name of the variables to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
values[]	The values of the variables
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.ns()

   AUTO $object, $objects[1], $variables[1]
   AUTO REAL values[1], status

   ; Read multiple values from multiple properties of the same object
   $object = "/Numeric Variable"
   $variables[0] = "Current Value"
   $variables[1] = "IsBoolean"
   CALL rm.read.nums($object, $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Read Value: ", status
      PAUSE
   END

   TYPE $variables[0], " = ", values[0]
   TYPE $variables[1], " = ", values[1]
```

```
    ; Read multiple values from multiple objects
    $objects[0] = "/Numeric Variable"
    $variables[0] = "Current Value"
    $objects[1] = "/Numeric Variable 2"
    $variables[1] = "Current Value"
    CALL rm.read.anums($objects[], $variables[], 1, values[], status)
    IF (status < 0) THEN
        TYPE "Unable To Read Value: ", status
        PAUSE
    END

    TYPE $objects[0], ".", $variables[0], " = ", values[0]
    TYPE $objects[0], ".", $variables[1], " = ", values[1]
.END
```

rm.read.num($object, $variable, wait.time, value, status)

Abstract

Read a numeric property variable associated with an object on the PC. The variable referenced must be a numeric type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variable	The name of the variable to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
value	The value of the variables
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.n()

    AUTO $object, $variable
    AUTO REAL value, status

    ; Write a value of 10 to the numeric variable in the workspace
    $object = "/Numeric Variable"
    $variable = "CurrentValue"
    value = 10
    CALL rm.write.num($object, $variable, 1, value, status)
    IF (status < 0) THEN
        TYPE "Unable To Write Value: ", status
        PAUSE
    END
```

```
   ; Read the variable and ensure it is the same value
   CALL rm.read.num($object, $variable, 1, value, status)
   IF (status < 0) THEN
      TYPE "Unable To Read the Value: ", status
      PAUSE
   END

   IF (value <> 10) THEN
      TYPE "The update did not succeed"
      PAUSE
   END
.END
```

rm.read.nums($object, $variables[], wait.time, values[], status)

Abstract

Read a list of numeric property variables associated with an object on the PC. The variable referenced must all be numeric types.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
values[]	The values of the variables
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.ns()

   AUTO $object, $objects[1], $variables[1]
   AUTO REAL values[1], status

   ; Read multiple values from multiple properties of the same object
   $object = "/Numeric Variable"
   $variables[0] = "Current Value"
   $variables[1] = "IsBoolean"
   CALL rm.read.nums($object, $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Read Value: ", status
      PAUSE
   END

   TYPE $variables[0], " = ", values[0]
   TYPE $variables[1], " = ", values[1]
```

```
    ; Read multiple values from multiple objects
    $objects[0] = "/Numeric Variable"
    $variables[0] = "Current Value"
    $objects[1] = "/Numeric Variable 2"
    $variables[1] = "Current Value"
    CALL rm.read.anums($objects[], $variables[], 1, values[], status)
    IF (status < 0) THEN
        TYPE "Unable To Read Value: ", status
        PAUSE
    END

    TYPE $objects[0], ".", $variables[0], " = ", values[0]
    TYPE $objects[0], ".", $variables[1], " = ", values[1]
.END
```

rm.read.str($object, $variable, wait.time, $value, status)

Abstract

Read a string property variable associated with an object on the PC. The variable referenced must be a string type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variable	The name of the variable to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
$value	The value of the variable
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.s()

    AUTO $object, $variable, $value
    AUTO REAL status

    ; Write a value of "Text" to the string variable in the workspace
    $object = "/String Variable"
    $variable = "CurrentValue"
    $value = "Text"
    CALL rm.write.str($object, $variable, 1, $value, status)
    IF (status < 0) THEN
        TYPE "Unable To Write Value: ", status
        PAUSE
    END
```

```
    ; Read the variable and ensure it is the same value
    CALL rm.read.str($object, $variable, 1, $value, status)
    IF (status < 0) THEN
        TYPE "Unable To Read the Value: ", status
        PAUSE
    END

    IF $value <> "Text" THEN
        TYPE "The update did not succeed"
        PAUSE
    END
.END
```

rm.read.strs($object, $variables[], wait.time, $values[], status)

Abstract

Read a list of string property variables associated with an object on the PC. The variable referenced must all be string types.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
$values[]	The values of the variables
status	Status of the operation: 0 = success, < 0 = error

Example

rm.read.trns($object, $variable, wait.time, value, status)

Abstract

Read a transform property variable associated with an object on the PC. The variable referenced must be a Transform3D type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"

$variable	The name of the variable to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
value	The value of the variable
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.t()

   AUTO $object, $variable
   AUTO REAL status
   AUTO LOC t.value

   ; Write a value of 10 to the transform variable in the workspace
   $object = "/Box"
   $variable = "OffsetFromParent"
   SET t.value = TRANS(10,10,10,0,0,180)
   CALL rm.write.trns($object, $variable, 1, t.value, status)
   IF (status < 0) THEN
      TYPE "Unable To Write Value: ", status
      PAUSE
   END

   ; Read the variable and ensure it is the same value
   CALL rm.read.trns($object, $variable, 1, t.value, status)
   IF (status < 0) THEN
      TYPE "Unable To Read the Value: ", status
      PAUSE
   END
.END
```

rm.read.trnss($object, $variables[], wait.time, values[], status)

Abstract

Read a list of transform property variables associated with an object on the PC. The variable referenced must all be Transform3D types.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access
wait.time	The amount of time to wait for the execution to complete

Output Parameters

Parameter	Description
values[]	The values of the variables
status	Status of the operation: 0 = success, < 0 = error

rm.save($file, wait.time, status)

Abstract
Saves the workspace file on the PC.

Input Parameters

Parameter	Description
$file	Name of the file on the PC to save. If empty, save to the current file
wait.time	The amount of time to wait for the operation to complete.

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example
```
.PROGRAM ex1.rm.save()
    AUTO REAL status
    AUTO $file

    $file = "C:\workspace.awp"
    CALL rm.save($file, 10, status)
    IF (status < 0) THEN
       TYPE "Unable To Save Workspace: ", status
       PAUSE
    END
.END
```

rm.write.anums($objects[], $variables[], wait.time, values[], status)

Abstract
Set the values of an array of numeric properties associated with a collection of objects on the PC. The property variables referenced must all be of a numeric type.

Input Parameters

Parameter	Description

$objects[]	The names of the remote objects to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access
wait.time	The amount of time to wait for the execution to complete
values[]	The values of the variables

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.nws()

   AUTO $objects[0], $variables[0]
   AUTO REAL values[0], status

   ; Write multiple values from multiple properties of the same object
   $objects[0] = "/Numeric Variable"
   $variables[0] = "CurrentValue"
   values[0] = 110
   CALL rm.write.nums($objects[0], $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Write Values: ", status
      PAUSE
   END

   ; Perform a multiple object write
   CALL rm.write.anums($objects[], $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Write Values: ", status
      PAUSE
   END
.END
```

rm.write.num($object, $variable, wait.time, value, status)

Abstract

Set the value of a numeric property associated with an object on the PC. The property variable referenced must be of a numeric type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variable	The name of the variable to access

wait.time	The amount of time to wait for the execution to complete
value	The value of the variable

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.n()

   AUTO $object, $variable
   AUTO REAL value, status

   ; Write a value of 10 to the numeric variable in the workspace
   $object = "/Numeric Variable"
   $variable = "CurrentValue"
   value = 10
   CALL rm.write.num($object, $variable, 1, value, status)
   IF (status < 0) THEN
      TYPE "Unable To Write Value: ", status
      PAUSE
   END

   ; Read the variable and ensure it is the same value
   CALL rm.read.num($object, $variable, 1, value, status)
   IF (status < 0) THEN
      TYPE "Unable To Read the Value: ", status
      PAUSE
   END

   IF (value <> 10) THEN
      TYPE "The update did not succeed"
      PAUSE
   END
.END
```

rm.write.nums($object, $variables[], wait.time, values[], status)

Abstract

Set the values of an array of numeric properties associated with an object on the PC. The property variables referenced must all be of a numeric type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access

wait.time	The amount of time to wait for the execution to complete
values[]	The values of the variables

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.nws()

   AUTO $objects[0], $variables[0]
   AUTO REAL values[0], status

   ; Write multiple values from multiple properties of the same object
   $objects[0] = "/Numeric Variable"
   $variables[0] = "CurrentValue"
   values[0] = 110
   CALL rm.write.nums($objects[0], $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Write Values: ", status
      PAUSE
   END

   ; Perform a multiple object write
   CALL rm.write.anums($objects[], $variables[], 1, values[], status)
   IF (status < 0) THEN
      TYPE "Unable To Write Values: ", status
      PAUSE
   END
.END
```

rm.write.str($object, $variable, wait.time, $value, status)

Abstract

Set the value of a string property associated with an object on the PC. The property variable referenced must be of a string type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variable	The name of the variable to access
wait.time	The amount of time to wait for the execution to complete
$value	The value of the variable

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.s()

   AUTO $object, $variable, $value
   AUTO REAL status

   ; Write a value of "Text" to the string variable in the workspace
   $object = "/String Variable"
   $variable = "CurrentValue"
   $value = "Text"
   CALL rm.write.str($object, $variable, 1, $value, status)
   IF (status < 0) THEN
      TYPE "Unable To Write Value: ", status
      PAUSE
   END

   ; Read the variable and ensure it is the same value
   CALL rm.read.str($object, $variable, 1, $value, status)
   IF (status < 0) THEN
      TYPE "Unable To Read the Value: ", status
      PAUSE
   END

   IF $value <> "Text" THEN
      TYPE "The update did not succeed"
      PAUSE
   END
.END
```

rm.write.strs($object, $variables[], wait.time, $values[], status)

Abstract

Set the values of an array of string properties associated with an object on the PC. The property variables referenced must all be of a string type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The names of the variables to access
wait.time	The amount of time to wait for the execution to complete
$values[]	The values of the variables

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

rm.write.trns($object, $variable, wait.time, value, status)

Abstract

Set the value of a transformation property associated with an object on the PC. The property variable referenced must be of a Transform3D type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access. This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variable	The name of the variable to access
wait.time	The amount of time to wait for the execution to complete
value	The value of the variable

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

Example

```
.PROGRAM ex1.rm.vars.t()

   AUTO $object, $variable
   AUTO REAL status
   AUTO LOC t.value

   ; Write a value of 10 to the transform variable in the workspace
   $object = "/Box"
   $variable = "OffsetFromParent"
   SET t.value = TRANS(10,10,10,0,0,180)
   CALL rm.write.trns($object, $variable, 1, t.value, status)
   IF (status < 0) THEN
      TYPE "Unable To Write Value: ", status
      PAUSE
   END

   ; Read the variable and ensure it is the same value
   CALL rm.read.trns($object, $variable, 1, t.value, status)
   IF (status < 0) THEN
      TYPE "Unable To Read the Value: ", status
      PAUSE
```

```
    END
.END
```

rm.write.trnss($object, $variables[], wait.time, values[], status)

Abstract

Set the values of an array of transformation properties associated with an object on the PC. The property variables referenced must all be of a Transform3D type.

Input Parameters

Parameter	Description
$object	The name of the remote object to access This must include the full path name of the object to access. e.g. "/directory/Robot 1"
$variables[]	The name of the variables to access
wait.time	The amount of time to wait for the operation to complete.
values[]	The values of the variables

Output Parameters

Parameter	Description
status	Status of the operation: 0 = success, < 0 = error

OMRON Corporation **Industrial Automation Company**
Kyoto, JAPAN
Contact: industrial.omron.eu

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200 Hoffman Estates,
IL 60169 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ADEPT TECHNOLOGIES, INC.
4550 Norris Canyon Road, Suite 150, San Ramon, CA 94583 U.S.A.
Tel: (1) 925-245-3400/Fax: (1) 925-960-0590

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower, 200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Printed in USA
Cat. No. I609-E-01 0316