

# EPSON

*EPSON RC+ 7.0 Option*

*Vision Guide 7.0*

Ver. 7.4

*Properties and Results Reference*

Rev.1

EM186S3688F

EPSON RC+ 7.0 Option Vision Guide 7.0 (Ver.7.4) Properties and Results Reference Rev.1

EPSON RC+ 7.0 Option

*Vision Guide 7.0 (Ver.7.4) Properties and Results Reference*

Rev.1

## FOREWORD

Thank you for purchasing our robot products. This manual contains the information necessary for the correct use of the EPSON RC+ software. Please carefully read this manual and other related manuals when using this software. Keep this manual in a handy location for easy access at all times.

## WARRANTY

The robot and its optional parts are shipped to our customers only after being subjected to the strictest quality controls, tests and inspections to certify its compliance with our high performance standards.

Product malfunctions resulting from normal handling or operation will be repaired free of charge during the normal warranty period. (Please ask your Regional Sales Office for warranty period information.)

However, customers will be charged for repairs in the following cases (even if they occur during the warranty period):

1. Damage or malfunction caused by improper use which is not described in the manual, or careless use.
2. Malfunctions caused by customers' unauthorized disassembly.
3. Damage due to improper adjustments or unauthorized repair attempts.
4. Damage caused by natural disasters such as earthquake, flood, etc.

Warnings, Cautions, Usage:

1. If the robot or associated equipment is used outside of the usage conditions and product specifications described in the manuals, this warranty is void.
2. If you do not follow the WARNINGS and CAUTIONS in this manual, we cannot be responsible for any malfunction or accident, even if the result is injury or death.
3. We cannot foresee all possible dangers and consequences. Therefore, this manual cannot warn the user of all possible hazards.

## TRADEMARKS

Microsoft, Windows, Windows logo, Visual Basic, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other brand and product names are trademarks or registered trademarks of the respective holders.

## TRADEMARK NOTIFICATION IN THIS MANUAL

Microsoft® Windows® XP Operating system

Microsoft® Windows® Vista Operating system

Microsoft® Windows® 7 Operating system

Microsoft® Windows® 8 Operating system

Microsoft® Windows® 10 Operating system

Throughout this manual, Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10 refer to above respective operating systems. In some cases, Windows refers generically to Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10.

## NOTICE

No part of this manual may be copied or reproduced without authorization.

The contents of this manual are subject to change without notice.

Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

## MANUFACTURER

**SEIKO EPSON CORPORATION**

## SAFETY PRECAUTIONS

Installation of robots and robotic equipment should only be performed by qualified personnel in accordance with national and local codes. Please carefully read this manual and other related manuals when using this software.

Keep this manual in a handy location for easy access at all times.

<b>Vision Properties and Results Reference</b>	<b>1</b>
Overview .....	1
Application .....	1
Vision Properties and Results Format Description .....	1
Vision Constants.....	2
AbortSeqOnFail Property .....	5
Accept Property .....	6
AcquireState Result.....	7
AllFound Result .....	8
AllPassed Result.....	9
AllRobotXYU Result .....	10
Angle Result .....	11
Angle1 Result .....	12
Angle2 Result .....	13
AngleAccuracy Property .....	14
AngleBase Property.....	15
AngleEnable Property.....	16
AngleEnd Property .....	17
AngleMaxIncrement Property .....	18
AngleMode Property.....	19
AngleObject Property .....	21
AngleObjectResult Property .....	22
AngleOffset Property .....	23
AngleRange Property .....	24
AngleStart Property .....	26
ApproachPoint Property .....	27
ArcObject Property .....	28
ArcObjResult Property.....	29
ArcSearchType Property .....	30
Area Result.....	31
AsyncMode Property .....	32
AutoCamPoints Property.....	33
AutoReference Property .....	34
AutoRefFinalRotation Property.....	35
AutoRefInitRotation Property.....	36
AutoRefMode Property .....	37
AutoRefTolerance Property .....	38
CalComplete Result.....	39

Calibration Property .....	40
CallImageSize Result .....	41
CalRobotPlacePos Property .....	42
Camera Property.....	43
CameraBrightness Property .....	45
CameraContrast Property.....	46
CameraOrientation Property.....	47
CameraX Result.....	48
CameraX1 Result.....	49
CameraX2 Result.....	51
CameraX3 Result.....	53
CameraX4 Result.....	54
CameraXYU Result .....	55
CameraY Result.....	56
CameraY1 Result.....	57
CameraY2 Result.....	59
CameraY3 Result.....	61
CameraY4 Result.....	62
Caption Property .....	63
CenterPntObjResult Property .....	64
CenterPntRotOffset Property.....	65
CenterPntOffsetX Property .....	66
CenterPntOffsetY Property .....	67
CenterPointObject Property.....	68
CenterX Property .....	69
CenterY Property .....	70
CharToTeach Property .....	71
CodabarChecksumEnabled Property .....	72
CoderbarOutputChecksum Property .....	73
CodabarOutputStartStop Property .....	74
Code39ChecksumEnabled Property .....	75
Code39OutputChecksum Property.....	76
Code39OutputStartStop Property.....	77
Code128OutputChecksum Property.....	78
CodeType Property.....	79
ColorIndex Result .....	80
ColorMode Property.....	81
ColorName Result.....	82

ColorValue Result.....	83
Compactness Result .....	84
Confusion Property.....	85
ContourMode Property .....	86
ContourTolerance Property .....	87
Contrast Result.....	88
ContrastTarget Property .....	89
ContrastVariation Property .....	90
Count Property.....	91
CurrentModel Property .....	92
CurrentResult Property.....	93
DataMatrixConnectDots Property.....	95
DataMatrixMinLength Property.....	96
DataMatrixPolarity Property .....	97
DefectAreaExtended Property.....	98
DefectLevel Result .....	99
DefectLevelThreshNeg Property.....	100
DefectLevelThreshPos Property .....	101
DetailLevel Property .....	102
DictionaryMode Property.....	103
Directed Property.....	104
Direction Property.....	105
DistCorrectCal Property.....	106
DistCorrectCalComplete Result .....	107
DistCorrectEnable Property.....	108
DistCorrectTargetSeq Property.....	109
DistCorrectType Property .....	110
EdgeCameraXYU Result.....	111
EdgePixelXYU Result.....	112
EdgeRobotXYU Result.....	113
EdgeThreshold Property.....	114
EdgeSort Property .....	115
EdgeType Property.....	116
EllipseAngle Property .....	117
EllipseMajorDiam Property .....	118
Enabled Property .....	119
EllipseMinorDiam Property .....	120
EndPntObjResult Property .....	121

EndPointObject Property .....	122
EndPointType Property.....	123
ExportFont Property.....	125
ExposureDelay Property.....	126
ExposureTime Property .....	127
Extrema Result .....	128
FailColor Property.....	129
FillHoles Property.....	130
FindChar Property.....	131
FitError Result.....	132
FittingThreshold Property .....	133
FocusValue Result.....	134
Font Property .....	135
FontBold Property.....	136
FontItalic Property.....	137
FontName Property .....	138
FontSize Property .....	139
Found Result.....	140
FoundCodeType Result.....	141
FoundMajorDiam Result.....	143
FoundMinorDiam Result.....	144
FoundOnEdge Result .....	145
FoundRadius Result .....	146
FOVHeight Result.....	147
FOVWidth Result .....	148
Frame Property.....	149
FrameResult Property.....	150
Graphics Property.....	151
GridColor Property.....	153
GridPitchX Property .....	154
GridPitchY Property .....	155
GridShow Property.....	156
GridType Property.....	157
GridUnits Property .....	158
HDRMode Property .....	159
Holes Result.....	160
ImageBuffer Property.....	161
ImageBuffer1 Property.....	162

---

ImageBuffer2 Property .....	163
ImageColor Property .....	164
ImageFile Property .....	165
ImageFileScale Property .....	166
ImageSize Property .....	167
ImageSource Property.....	168
ImportFont Property.....	169
Index Property .....	170
InspectEndOffset Property .....	171
InspectStartOffset Property .....	172
InvalidChar Property.....	173
Iterations Property .....	174
ITFChecksumEnabled Property .....	175
ITFOutputChecksum Property.....	176
KeepRGBRatio Property .....	177
KernelHeight Property .....	178
KernelWidth Property .....	179
LabelBackColor Property.....	180
Lamp Property .....	181
LampDelay Property.....	182
Length Result.....	183
LineDirection Property .....	184
LineObject Property .....	185
LineObjResult Property .....	186
LineObj1Result Property .....	187
LineObj2Result Property .....	188
LineObject1 Property.....	189
LineObject2 Property.....	190
LJMMode Property .....	191
LuminanceCorrection Property.....	192
MajorDiameter Result.....	193
MaxArea Property.....	194
MaxError Result.....	195
MaxFeretDiameter Result.....	196
MaxLength Property .....	197
MaxMoveDist Property .....	198
MaxPixelLength Property .....	199
MaxRGB Property .....	200

MaxX Result.....	201
MaxY Result.....	202
MinArea Property .....	203
MinorDiameter Result.....	204
MinLength Property .....	205
MinMaxArea Property .....	206
MinPixelLength Property.....	207
MinRGB Property.....	208
MinX Result.....	209
MinY Result.....	210
MissingEdgeType Property.....	211
ModelColor Property.....	212
ModelColorTol Property.....	213
ModelName Property.....	214
ModelObject Property.....	215
ModelOK Property .....	216
ModelOrgAutoCenter Property .....	217
ModelOrgFindCenter Property .....	218
ModelOrgX Property .....	219
ModelOrgY Property .....	220
ModelWin Property .....	221
ModelWinAngle Property .....	222
ModelWinCenterX Property .....	223
ModelWinCenterY Property .....	224
ModelWinHeight Property.....	225
ModelWinLeft Property .....	226
ModelWinTop Property .....	227
ModelWinType Property .....	228
ModelWinWidth Property .....	229
MotionDelay Property .....	230
Name Property.....	231
NumberFound Result.....	232
NumberOfEdges Property .....	234
NumberOfModels Property .....	235
NumberOfResults Property.....	236
NumberToFind Property .....	237
Objects Property .....	239
Operation Property.....	240

---

Orientation Property .....	243
OriginAngleEnabled Property .....	244
OriginPntObjResult Property .....	245
OriginPoint Property .....	246
Overlapped Result.....	247
PassColor Property .....	248
Passed Result.....	249
PassType Property .....	250
PDFScanInterval Property.....	251
Perimeter Result.....	252
PixelLength Result.....	253
PixelLine Result.....	254
PixelMajorDiam Result .....	255
PixelMinorDiam Result .....	256
PixelRadius Result .....	257
PixelX Result .....	258
PixelX1 Result .....	259
PixelX2 Result .....	261
PixelX3 Result .....	263
PixelX4 Result .....	264
PixelXYU Result .....	265
PixelY Result .....	266
PixelY1 Result .....	267
PixelY2 Result .....	269
PixelY3 Result .....	271
PixelY4 Result .....	272
PointsTaught Property.....	273
PointType Property.....	274
Polarity Property .....	275
QRLargeSize Property .....	278
QRMinContrast Property .....	279
QRMinLength Property.....	280
QRNarrowQuietZone Property .....	281
QROutputID Property .....	282
Radius Property .....	283
RadiusInner Property.....	284
RadiusOuter Property.....	285
RejectOnEdge Property.....	286

ReferenceType Property.....	287
ResultObject Property.....	288
ResultText1 Property.....	289
ResultText2 Property.....	292
ResultText3 Property.....	295
Reversed Result.....	298
RobotAccel Property.....	299
RobotArm Property.....	300
RobotLimZ Property.....	301
RobotLocal Property.....	302
RobotNumber Property.....	303
RobotPlacePos Result.....	304
RobotPlaceTargetPos Property.....	305
RobotPos Property.....	306
RobotSpeed Property.....	307
RobotTool Property.....	308
RobotToolXYU Result.....	309
RobotU Result.....	310
RobotUOffset Property.....	311
RobotX Result.....	312
RobotX1 Result.....	313
RobotX2 Result.....	315
RobotX3 Result.....	317
RobotX4 Result.....	318
RobotXYU Result.....	319
RobotY Result.....	320
RobotY1 Result.....	321
RobotY2 Result.....	323
RobotY3 Result.....	325
RobotY4 Result.....	326
RotationAngle Property.....	327
RotationDirection Property.....	328
Roughness Result.....	329
RuntimeAcquire Property.....	330
RuntimeContour Property.....	332
RuntimeFreeze Property.....	333
SamplingPitchProperty.....	334
SaveImage Property.....	335

Scale Result.....	336
ScaleEnable Property.....	337
ScaleFactorMax Property.....	338
ScaleFactorMin Property.....	339
ScaleTarget Property.....	340
ScaleTargetPriority Property.....	341
Score Result.....	342
ScoreWeightContrast Property.....	344
ScoreWeightStrength Property.....	345
SearchLength Property.....	346
SearchLength1 Property.....	347
SearchLength2 Property.....	348
SearchLength3 Property.....	349
SearchLength4 Property.....	350
SearchPolarity Property.....	351
SearchReducedImage Property.....	352
SearchType Property.....	353
SearchWidth Property.....	354
SearchWin Property.....	355
SearchWinAngle Property.....	356
SearchWinCenterX Property.....	357
SearchWinCenterY Property.....	358
SearchWinHeight Property.....	359
SearchWinLeft Property.....	360
SearchWinTop Property.....	361
SearchWinType Property.....	362
SearchWinWidth Property.....	363
SeparationAngle Property.....	364
SeparationMinX Property.....	365
SeparationMinY Property.....	366
SeparationScale Property.....	367
Sequences Property.....	368
SharedEdges Property.....	369
ShowAllResults Result.....	370
ShowConfirmation Property.....	371
ShowExtensions Property.....	372
ShowLabel Property.....	373
ShowModel Property.....	374

ShowProcessing Property .....375

SizeToFind Property .....376

SkewDirection Result.....377

SkewFitEnable Property .....378

SkewRatio Result.....379

Smoothness Property .....380

Sort Property .....381

StartPntObjResult Property .....383

StartPointObject Property .....384

StartPointType Property .....385

Strength Result .....387

StrengthTarget Property .....388

StrengthVariation Property .....389

StrobeBlackVideo Property.....390

StrobeDelay Property .....391

StrobeTime Property.....392

TargetSequence Property.....393

Text Result.....394

TextBackColor Property.....395

Thickness Property .....396

ThresholdColor Property.....397

ThresholdAuto Property.....398

ThresholdHigh Property.....399

ThresholdLow Property.....400

Time Result.....401

TimedOut Result.....402

Timeout Property .....403

TotalArea Result .....404

TriggerDebounce Property .....405

TriggerMode Property.....406

TwoRefPoints Property.....407

Type Property .....408

UPCExpansionEnabled Property .....410

UPCOutputChecksum Property.....411

UpwardLamp Property.....412

UpwardSequence Property.....413

UserText Property.....414

VCal Statement.....415

---

VCalPoints Statement .....	417
VClIs Statement.....	418
VCreateCalibration Statement.....	419
VCreateObject Statement.....	420
VCreateSequence Statement.....	421
VDefArm Statement.....	422
VDefGetMotionRange Statement.....	424
VDefLocal Statement.....	425
VDefSetMotionRange Statement .....	427
VDefTool Statement .....	428
VDeleteCalibration Statement .....	430
VDeleteObject Statement .....	431
VDeleteSequence Statement .....	432
VGet Statement .....	433
VGoCenter Statement .....	435
VLoad Statement .....	436
VLoadModel Statement .....	437
VRun Statement .....	438
VSave Statement.....	439
VSavelImage Statement.....	440
VSaveModel Statement.....	441
VSet Statement.....	442
VShowModel Statement .....	444
VStatsReset Statement .....	445
VStatsResetAll Statement .....	446
VStatsSave Statement .....	447
VStatsShow Statement.....	448
VTeach Statement.....	449
VTrain Statement.....	450
X Property .....	451
X1 Property.....	452
X2 Property.....	453
XAvgError Result.....	454
XMaxError Result .....	455
XmmPerPixel Result.....	456
XTilt Result .....	457
Y Property .....	458
Y1 Property.....	459

Y2 Property .....	460
YAxisPntObjResult Property.....	461
YAxisPoint Property .....	462
YAvgError Result .....	463
YMaxError Result.....	464
YmmPerPixel Result.....	465
YTilt Result.....	466
ZoomFactor Property .....	467

# Vision Properties and Results Reference

## Overview

This reference manual explains all Vision Guide sequence, object, and calibration properties and results, and all Vision Guide SPEL+ commands. For more information on how to use Vision Guide, refer to the *Vision Guide* manual.

## Application

This manual applies to the following versions of the products.

EPSON RC+ 7.0 Ver. 7.1.4 or later

CV1 / CV2 firmware Ver. 2.3.1.0 or later

## Vision Properties and Results Format Description

All Vision Guide properties and results are listed in the pages that follow. An explanation of the headings for the property and result reference pages is given below:

Applies To	If the property or result is used with vision objects, then this section simply lists the vision objects for which this property applies. (Ex. Blob, Correlation, Polar...) If the property or result is used with vision sequences then the words Vision Sequence will appear in this section. If the property or result is used with vision calibration then the words Vision Calibration will appear in this section.
Description	A simple description is given for each property or result. This section is normally very short for simplicity.
Usage	The Usage Section describes how to access the property or result from the SPEL+ Language.
Values	Describes the range of acceptable values which the property can be set to or which the result will return. A default value is also shown for those properties that have a default.
Remarks	Explains more details than the Description Section. This section is normally used to describe any caveats or special information that may apply to the specific property or result. (It is highly recommended to read the Remarks Section for each property prior to its usage.)
See Also	Gives a list of related properties, results, vision objects and other topics that may prove useful to review.
Runtime only	This is displayed under the property or result name when it applies. Runtime only properties and results cannot be accessed from the Vision Guide GUI. They can only be accessed from the SPEL+ language or from RC+ API .

## Vision Constants

Following constants are provided for Vision Guide 7.0.  
Constants are used at program creation as needed.

### Tip

Although values can be specified directly instead of using constant names, it is recommended to use the constant names for programs.

Constant Name	Value	Application
VISION_ACQUIRE_NONE	0	RuntimeAcquire property
VISION_ACQUIRE_STATIONARY	1	
VISION_ACQUIRE_STROBED	2	
VISION_ANGLEMODE_DEFAULT	1	AngleMode property
VISION_ANGLEMODE_USEANGLEBASE	2	
VISION_ARCSEARCHTYPE_CIRCLE	0	ArcSearchType property
VISION_ARCSEARCHTYPE_ELLIPSE	1	
VISION_BACKCOLOR_BLACK	1	BackColor property
VISION_BACKCOLOR_NONE	0	
VISION_BACKCOLOR_WHITE	2	
VISION_CALIBPLATE_L	1	VDefLocal statement
VISION_CALIBPLATE_M	2	
VISION_CALIBPLATE_S	3	
VISION_CALIBPLATE_XS	4	
VISION_CAMORIENT_FIXEDDOWN	2	CameraOrientation property
VISION_CAMORIENT_FIXEDUP	3	
VISION_CAMORIENT_MOBILEJ2	4	
VISION_CAMORIENT_MOBILEJ4	5	
VISION_CAMORIENT_MOBILEJ5	6	
VISION_CAMORIENT_MOBILEJ6	7	
VISION_CAMORIENT_STANDALONE	1	
VISION_CODETYPE_AUTO	0	CodeType property
VISION_CODETYPE_CODABAR	6	
VISION_CODETYPE_CODE39	3	
VISION_CODETYPE_CODE128	5	
VISION_CODETYPE_DATAMATRIX	1	
VISION_CODETYPE_EAN8	13	
VISION_CODETYPE_EAN13	2	
VISION_CODETYPE_INTERLEAVED25	4	
VISION_CODETYPE_PDF417	8	
VISION_CODETYPE_QR	10	
VISION_CODETYPE_UPC	20	
VISION_CODETYPE_UPCA	18	
VISION_CODETYPE_UPCE	19	
VISION_CONTOURMODE_BLOB	1	ContourMode property
VISION_CONTOURMODE_LINE	2	
VISION_CONTOURMODE_ARC	3	
VISION_DEFARM_J2CAM	1	VDefArm statement
VISION_DEFARM_MODE_ROUGH	1	VDefArm statement
VISION_DEFARM_MODE_FINE	2	
VISION_DEFLOCAL_J5CAM	1	VDefLocal statement
VISION_DEFLOCAL_J6CAM	2	
VISION_DEFLOCAL_UPCAM	3	
VISION_DEFLOCAL_DOWNCAM	4	
VISION_DEFTOOL_J4CAM	1	VDefTool statement
VISION_DEFTOOL_J6CAM	2	
VISION_DEFTOOL_FIXEDNOCAL	3	
VISION_DEFTOOL_FIXEDWITHCAL	4	

Constant Name	Value	Application
VISION_DETAILLEVEL_HIGH	2	DetailLevel property
VISION_DETAILLEVEL_MEDIUM	1	
VISION_DETAILLEVEL_VERYHIGH	3	
VISION_EDGETYPE_SINGLE	1	EdgeType property
VISION_EDGETYPE_PAIR	2	
VISION_GRAPHICS_ALL	1	Graphics property
VISION_GRAPHICS_NONE	3	
VISION_GRAPHICS_POSONLY	2	
VISION_GRIDTYPE_CROSSHAIR	1	GridType property
VISION_GRIDTYPE_RECTANGLE	2	
VISION_GRIDUNITS_PIXEL	1	GridUnits property
VISION_GRIDUNITS_MM	2	
VISION_IMAGECOLOR_ALL	1	ImageColor property
VISION_IMAGECOLOR_BLUE	4	
VISION_IMAGECOLOR_GRAYSCALE	5	
VISION_IMAGECOLOR_GREEN	3	
VISION_IMAGECOLOR_RED	2	
VISION_IMAGESIZE_320X240	1	ImageSize property
VISION_IMAGESIZE_640X480	2	
VISION_IMAGESIZE_800X600	3	
VISION_IMAGESIZE_1024X768	4	
VISION_IMAGESIZE_1280X1024	5	
VISION_IMAGESIZE_1600X1200	6	
VISION_IMAGESIZE_2048X1536	7	
VISION_IMAGESIZE_2560X1920	8	
VISION_IMAGESIZE_3664X2748	9	
VISION_IMAGESOURCE_CAMERA	1	ImageSource property
VISION_IMAGESOURCE_FILE	2	
VISION_LINEDIRECTION_LEFTTORIGHT	1	LineDirection property
VISION_LINEDIRECTION_RIGHTTOLEFT	2	
VISION_LUMINANCECORRECTION_NONE	1	LuminanceCorrection property
VISION_LUMINANCECORRECTION_HISTGRAM	2	
VISION_OPERATION_BINARIZE	16	Operation property
VISION_OPERATION_CLOSE	2	
VISION_OPERATION_COLORFILTER	21	
VISION_OPERATION_COLORSTRETCH	24	
VISION_OPERATION_OPEN	1	
VISION_OPERATION_DETECTFOCUS	26	
VISION_OPERATION_DILATE	4	
VISION_OPERATION_EDGEDETECT1	10	
VISION_OPERATION_EDGEDETECT2	11	
VISION_OPERATION_ERODE	3	
VISION_OPERATION_FLIPBOTH	20	
VISION_OPERATION_FLIPHORIZ	18	
VISION_OPERATION_FLIPVERT	19	
VISION_OPERATION_HORIZEDGE	8	
VISION_OPERATION_LAPLACE1	12	
VISION_OPERATION_LAPLACE2	13	
VISION_OPERATION_ROTATE	17	
VISION_OPERATION_SHARPEN1	6	
VISION_OPERATION_SHARPEN2	7	
VISION_OPERATION_SHIFT	25	
VISION_OPERATION_SMOOTH	5	
VISION_OPERATION_SUBTRACTABS	22	
VISION_OPERATION_THICKEN	15	
VISION_OPERATION_THIN	14	
VISION_OPERATION_VERTEDGE	9	
VISION_OPERATION_ZOOM	23	

## Vision Constants

Constant Name	Value	Application
VISION_ORIENT_BOTH	1	Orientation property
VISION_ORIENT_HORIZ	2	
VISION_ORIENT_VERT	3	
VISION_PASSTYPE_SOMEFOUND	1	PassType property
VISION_PASSTYPE_ALLFOUND	2	
VISION_PASSTYPE_SOMENOTFOUND	3	
VISION_PASSTYPE_ALLNOTFOUND	4	
VISION_POINTTYPE_ENDPOINT	1	PointType property
VISION_POINTTYPE_MIDPOINT	2	
VISION_POINTTYPE_PERPTOENDPOINT	7	
VISION_POINTTYPE_PERPTOLINE	3	
VISION_POINTTYPE_PERPTOMIDPOINT	6	
VISION_POINTTYPE_PERPTOSTARTPOINT	5	
VISION_POINTTYPE_POINT	0	
VISION_POINTTYPE_STARTPOINT	4	
VISION_POLARITY_DARK	1	Polarity property
VISION_POLARITY_LIGHT	2	
VISION_POLARITY_BOTH	3	
VISION_REFTYPE_TAUGHTPOINTS	1	ReferenceType property
VISION_REFTYPE_UPWARDCAMERA	2	
VISION_ROTATIONDIR_CCW	1	RotationDirection property
VISION_ROTATIONDIR_CW	2	
VISION_SIZETOFIND_ANY	0	SizeToFind property
VISION_SIZETOFIND_SMALLEST	2	
VISION_SIZETOFIND_LARGEST	1	
VISION_SORT_CAMERAX	4	Sort property
VISION_SORT_CAMERAY	5	
VISION_SORT_CAMERAXY	6	
VISION_SORT_NONE	0	
VISION_SORT_PIXELX	1	
VISION_SORT_PIXELY	2	
VISION_SORT_PIXELXY	3	
VISION_SORT_ROBOTX	7	
VISION_SORT_ROBOTY	8	
VISION_SORT_ROBOTXY	9	
VISION_TRIGGERMODE_LEADINGEDGE	1	TriggerMode property
VISION_TRIGGERMODE_TRAILINGEDGE	2	
VISION_THRESHCOLOR_BLACK	1	ThresholdColor property
VISION_THRESHCOLOR_WHITE	2	
VISION_WINTYPE_RECTANGLE	1	ModelWinType property SearchWinType property
VISION_WINTYPE_ROTATEDRECT	2	
VISION_WINTYPE_CIRCLE	3	

# AbortSeqOnFail Property

## Applies To

Vision Objects: All

## Description

Allows the user to specify that if an object fails (i.e. is not passed), then the entire sequence is aborted at that point and no further objects are processed.

## Usage

**VGet** *Sequence.Object.AbortSeqOnFail*, *var*

**VSet** *Sequence.Object.AbortSeqOnFail*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

False Does not cause the sequence to abort when the object is not found.

True Causes the sequence to abort when the object is not found.

Default: 0 – False

## Remarks

Use AbortSeqOnFail when you no longer want a sequence to continue if an object is not passed.

## See Also

Blob Object, ColorMatch Object, Correlation Object, Edge Object, Frame Object, Geometric Object, Line Object, Point Object, CodeReader Object, OCR Object, Polar Object, LineFinder Object, ArcFinder Object, DefectFinder Object, ArcInspector Object, LineInspector Object, Object Tab

## Accept Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, ColorMatch, Contour, CornerFinder, Correlation, Edge, Geometric, LineFinder, LineInspector, Polar

### Description

The Accept property specifies the score that a feature must equal or exceed to be considered found.

### Usage

**VGet** *Sequence.Object.Accept*, *var*

**VSet** *Sequence.Object.Accept*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number from 1 - 999

Default: 700 – ColorMatch, Correlation, Geometric, Polar  
100 – Edge, ArcFinder, LineFinder, ArcInspector, LineInspector, BoxFinder, Contour, CornerFinder

### Remarks

(Only Correlation, Geometric, Polar)The Accept property also affects the searching speed in a given area of the Region of Interest. When the Accept property is high, a target feature must be very similar to the registered model. Therefore, many regions can be ruled out by a cursory search and not pursued further. However, if the Accept property is low, target features that are only slightly similar to the registered model may exceed the Accept property, so that a detailed search of more regions in the scene is needed. Thus, raising the Accept property tends to reduce the time required for searches.

If the specified value is small, it may result in false detection.

### See Also

ColorMatch Object, Confusion Property, Correlation Object, Edge Object, Geometric Object, Polar Object, Score Result, ArcFinder Object, LineFinder Object, ArcInspector Object, LineInspector Object, Object Tab, BoxFinder Object, CornerFinder Object, Contour Object

# AcquireState Result

Runtime only

## Applies To

Vision Sequence

## Description

The AcquireState result is used to determine if a picture has been taken for a sequence after the external trigger becomes active.

To use the external trigger (strobe), set RuntimeAcquire property of the sequence to Strobed.

## Usage

**VGet** *Sequence.AcquireState*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the result.

## Values

0 Picture has not been taken.

3 Image has been acquired.

## Remarks

After calling VRun for a sequence to use the external trigger (strobe), the SPEL<sup>+</sup> program can wait for the external trigger input becomes active (strobe flashes) and the AcquireState to become 3 before further vision processing can continue. If the program does not wait for AcquireState to become 3, then executing vision commands for the same sequence will wait automatically for AcquireState to become 3 before executing.

## See Also

Object Tab, RunTimeAcquire Property

## AllFound Result

### Applies To

Vision Sequence

### Description

The AllFound result returns whether or not all objects within the specified sequence were found.

### Usage

**VGet** *Sequence.AllFound*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the result.

### Values

0 – False One of the objects within the sequence was not found.

1 – True All objects within the sequence were found.

### Remarks

The AllFound result is used to determine that all objects within a specified sequence are found. This result only applies to sequences.

### See Also

AllPassed Result, Found Result, Passed Result, Sequence Tab, Time Result, Vision Sequences

## AllPassed Result

### Applies To

Vision Sequence

### Description

Returns whether all objects of the specified sequence are passed or not.

### Usage

**VGet** *Sequence.AllPassed*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the result.

### Values

0 – False One or more of the objects within the sequence did not pass.

1 – True All objects within the sequence were passed.

### Remarks

The AllPassed result is used to determine that all objects within a specified sequence are passed. This result only applies to sequences.

### See Also

AllFound Result, Passed Result, Found Result, Time Result, Sequence Tab, Vision Sequence

## AllRobotXYU Result

Runtime only

### Applies To

Vision Objects: ArcInspector, Blob, CodeReader, Correlation, Edge, Geometric, Point, Polar, DefectFinder, LineInspector

### Description

Stores all of the found RobotX, RobotY and RobotU position coordinates of the found part's position with respect to the robot coordinate system into a WorkQue.

### Usage

**VGet** *Sequence.Object.AllRobotXYU, workQueID*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*workQueID* Integer expression representing the ID for the workQue that will receive the data.

### Remarks

The AllRobotXYU result stores all of the found positions in the robot coordinate system into the specified WorkQue.

It should be noted that the AllRobotXYU result can be only used for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then the RobotXYU result cause an error to occur.

### See Also

Blob Object, CameraX Result, CameraY Result, CameraXYU Result, Correlation Object, DefectFinder Object, Edge, Found Result, Geometric Object, PixelXYU Result, Point Object, Polar Object, RobotUOffset Property, RobotX Result, RobotY Result, RobotU Result

# Angle Result

## Applies To

Vision Objects: Blob, BoxFinder, CodeReader, CornerFinder, Correlation, DefectFinder, Frame, Geometric, Line, LineFinder, Polar

## Description

Returns the angle of the found object.

## Usage

**VGet** *Sequence.Object.Angle*[(result)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in degrees

Blob	: -90 to 90 degrees
BoxFinder, CornerFinder	: -180 to 180 degrees
Others	: 0 to 360 degrees

## Remarks

The Angle result returns the found part's angle in the image coordinate system. In some cases, you may want to use a Polar object to determine angle because it can be faster and more accurate.

## Statistics

For the Angle result, the following results can be acquired by statistics:

AngleMax, AngleMean, AngleMin, AngleRange, AngleStdDev.

Refer to *Statistics* in the *Vision Guide* manual for details on how to use statistics.

## See Also

AngleEnable Property, AngleMaxIncrement Property, AngleOffset Property, AngleTolerance Property, Blob Object, Correlation Object, Frame Object, Geometric Object, Line Object, NumberFound Result, NumberToFind Property, Polar Object, RobotU Result, LineFinder Object, DefectFinder Object, CodeReader Object, BoxFinder Object, CornerFinder Object, Object Tab

## Angle1 Result

### Applies To

Vision Objects: ArcFinder

### Description

Returns a starting point angle of the circular object found by ArcFinder Object.

### Usage

**VGet** *Sequence.Object.Angle1*[(result)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number representing the starting point angle of the found circular object.

### Remarks

Angle1 Result returns the starting point angle of the found circular object in the image coordinate system.

### See Also

ArcFinder Object, Angle2 Result

## Angle2 Result

### Applies To

Vision Objects: ArcFinder

### Description

Returns the end angle of the circular object found by an ArcFinder Object.

### Usage

**VGet** *Sequence.Object.Angle2*[(result)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number representing the ending point angle of the found circular object.

### Remarks

Angle2 Result returns the end angle of the circular object found by an ArcFinder Object in image coordinate system.

### See Also

ArcFinder Object, Angle1Result

## AngleAccuracy Property

### Applies To

Vision Objects: Correlation

### Description

Specifies the angle detection accuracy of a correlation search. (Specify the angle detection accuracy in angle.)

### Usage

**VGet** *Sequence.Object.AngleAccuracy, var*

**VSet** *Sequence.Object.AngleAccuracy, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real number in degrees from 0.1 – 10

Default: 1

### Remarks

The AngleAccuracy property is used at model training time and not at run-time. This value specifies the desired accuracy for angle search.

The Correlation model must be taught after a new value for the AngleAccuracy property is set in order for the new setting to become effective. If you teach a Correlation Model, then later set the AngleAccuracy property to a new value, and then try to run that Correlation object, it will not search with the new angle accuracy. You must re-teach the Correlation Model with the AngleEnable property set to 1–True, and with the new value for the AngleAccuracy property in order for Correlation search with angle to use the new AngleAccuracy property value.

### See Also

AngleMaxIncrement Property, AngleRange Property, Angle Result, Correlation Object, Object Tab

# AngleBase Property

## Applies To

Vision Objects: Line, LineFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Sets the reference angle for outputting angles.

## Usage

**VGet** *Sequence.Object.AngleBase*, *var*

**VSet** *Sequence.Object.AngleBase*, *value*

**VGet** *Sequence.Object.AngleBase*, *var*

**VSet** *Sequence.Object.AngleBase*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Real number in degrees from 0 to 360 when the Directed property is True

Real number in degrees from -90 to 90 when the Directed property is False

Default: 0

## Remarks

AngleBase sets the reference angle for outputting angles. AngleOutType must be set to “2: UseAngleBase” to enable this property. Angles are output with this reference angle at the center based on the conditions set for the Directed property. For details, refer to Remarks in the AngleOutType property.

## See Also

Angle Result, AngleMode Property, Directed Property

# AngleEnable Property

## Applies To

Vision Objects: Correlation, Geometric

## Description

Specifies whether or not a correlation or geometric object will search for rotation of a feature.

## Usage

**VGet** *Sequence.Object.AngleEnable, var*

**VSet** *Sequence.Object.AngleEnable, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Do not search for rotation.

1 – True Search for rotation.

Default: Correlation: 0 – False, Geometric: 1-True

## Remarks

The AngleRange and AngleMaxIncrement properties will not be used with the correlation or geometric search object if the AngleEnable property is set to 0–False.

It should be noted that while correlation with angle is normally able to find rotated parts, the correlation search time usually increase significantly. This is why correlation with angle is most useful for finding parts which rotate slightly. On the other hand, Geometric search is suitable for a pattern search which includes large rotation detection. If it is predictable that the parts have a large rotation angle, consider to use Geometric search.

Also, the Polar object is normally very fast and can be used in conjunction with the Correlation object for a powerful and fast combination. (See the sections on Correlation or Polar Searching for more information.)

The Correlation Model must be taught after the AngleEnable property is set to 1–True. If you teach a Correlation Model, then set the AngleEnable property to 1–True, and then try to run that Correlation object, it will not search with angle. You must re-teach the Correlation Model with the AngleEnable property set to 1–True in order for Correlation search with angle to work properly. You must also have the proper settings for the AngleMaxIncrement and AngleRange Properties prior to teaching the new Model as well.

## See Also

AngleMaxIncrement Property, AngleRange Property, Angle Result, Correlation Object, Geometric Object, Object Tab

# AngleEnd Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Contour

## Description

Sets the end angle of the range in which ArcFinder finds circular objects.

## Usage

**VGet** *Sequence.Object.AngleEnd* , *var*

**VSet** *Sequence.Object.AngleEnd* , *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

AngleStart + 10 to 360 (degree)

Default: 10

## Remarks

Specifies a range in which ArcFinder finds circular objects or ArcInspector inspects circular objects. The range should be between AngleStart and AngleEnd.

## See Also

ArcFinder Object, ArcInspector Object, AngleStart Property, Contour Object ,Object Tab

# AngleMaxIncrement Property

## Applies To

Vision Objects: Correlation

## Description

The AngleMaxIncrement property specifies the maximum angle increment amount for teaching a correlation model for searching with angle. When teaching a model, the system selects the angle increment amount automatically. By setting the AngleMaxIncrement property, the system compares the automatically-selected angle increment amount and the setting value. Then, the smaller amount will be used.

## Usage

**VGet** *Sequence.Object.AngleMaxIncrement, var*

**VSet** *Sequence.Object.AngleMaxIncrement, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real number in degrees from 1 - 45

Default: 5

## Remarks

Before reading anything else about the AngleMaxIncrement property it should be noted that the Correlation Model must be taught after a new value for the AngleMaxIncrement property is set in order for those settings to become effective. If you try to find the Correlation objects by changing the AngleMaxIncrement property to a new value after teaching a Correlation Model, it will not search with the new angle increment. You must re-teach the Correlation Model when you change the AngleMaxIncrement property.

If you wish to measure angle precisely, you should provide a maximum angle increment corresponding to the degree of precision you desire. You should keep in mind, however, that the smaller the angle increment, the more storage will be required for the model and the slower the search time will be.

It should be noted that while correlation with angle is normally able to find rotated parts, the correlation search time usually increase significantly. This is why correlation with angle is most useful for finding parts which rotate slightly. On the other hand, Geometric search is suitable for a pattern search which includes large rotation detection. If it is predictable that the parts have a large rotation angle, consider to use Geometric search.

Also, the Polar object is normally very fast and can be used in conjunction with a Correlation object for a powerful and fast combination. (See the sections on Correlation or Polar Searching for more information.)

## See Also

Angle Result, AngleEnable Property, AngleRange Property, Correlation Object, Geometric Object, Object Tab

# AngleMode Property

## Applies To

Vision Objects: Line, LineFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Sets the output format for the angle detected.

## Usage

**VGet** *Sequence.Object.AngleMode*, *var*

**VSet** *Sequence.Object. AngleMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 - Default Outputs the angle as a real number in degrees from 0 to 360.

2 - UseAngleBase Outputs the angle based on AngleBase and Directed property settings.

Default 1 - Default

## Remarks

1 - Default

Outputs the angle in the conventional manner as a real number in degrees from 0 to 360.

Directed and AngleBase setting values are not used.

2 – UseAngleBase

Outputs the angle based on the conditions set for the Directed property with the reference angle set for the AngleBase property at the center.

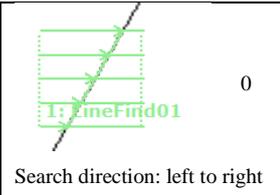
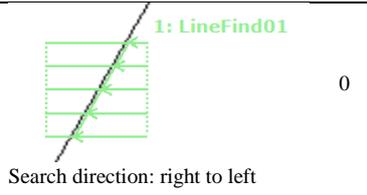
If the Directed property is true

$-180 \text{ degrees} + \text{AngleBase} \leq \text{output angle} \leq 180 \text{ degrees} + \text{AngleBase}$

If the Directed property is false

$-90 \text{ degrees} + \text{AngleBase} \leq \text{output angle} \leq 90 \text{ degrees} + \text{AngleBase}$

Example: The angle output when the AngleBase is set to 0 degrees when detecting a line of 60 degrees.

	 <p>Search direction: left to right</p>	 <p>Search direction: right to left</p>
AngleOutType: <b>Default</b>	60	240
AngleOutType: <b>UseAngleStart</b> Directed: <b>true</b>	60	-120
AngleOutType: <b>UseAngleStart</b> Directed: <b>false</b>	60	60

(Unit: degrees)

The output angles for Line and Line Finder objects will vary as shown in the diagram above even if line placement conditions put them on the same line.

The left side is output as 60 degrees and the right side is output as 240 degrees.

If the Directed property is “true”, the object placement conditions are taken into consideration to output the angle using the AngleBase as a reference. As such, the left side is output as 60 degrees and the right side is output as -120 degrees.

If the Directed property is “false”, the angle is output without regard for the object placement conditions. As such, the diagram above would be output as 60 degrees, regardless of the object placement conditions.

**See Also**

Angle Result, Directed Property

# AngleObject Property

## Applies To

Vision Objects: ImageOp, Point

## Description

Sets what object is used as a reference to determine the object angle.

## Usage

**VGet** *Sequence.Object.AngleObject*, *var*

**VSet** *Sequence.Object.AngleObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

“Screen”, or the name of the object in front of the object returning Angle results

Default: Screen (default setting when creating an object)

## Remarks

For the ImageOp object, use AngleObject together with the Rotate setting of the Operation property to automatically rotate the image according to the Angle result for the object specified in AngleObject.

When specifying an object as the AngleObject, to compensate for the degree of rotation of the object, set the RotationDirection property to “2 – CW”.

The resulting angle of the object set for the AngleObject property will be output as the resulting angle for the Point object. For example, if Blob is set as the AngleObject for a Point object, the resulting angle for the Point object will be the resulting angle for the Blob object. If objects set as the AngleObject contain multiple results, the result number to be used can be specified with the AngleObjectResult property.

The following objects can be specified:

Blob, BoxFinder, Correlation, CodeReader, Contour, DefectFinder, Frame, Geometric, ImageOp, Line, LineFinder, Polar, Point

## See Also

AngleObjectResult Property, Blob Object, BoxFinder Object, CodeReader Object, Contour Object, Correlation Object, DefectFinder Object, Frame Object, Geometric Object, ImageOp Object, Operation Property, Polar Object, Point Object, RotationAngle Property, RotationDirection Property, Line Object, LineFinder Object

## AngleObjectResult Property

### Applies To

Vision Objects: ImageOp, Point  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Specifies the results used by the AngleObject property.

### Usage

**VGet** *Sequence.Object.AngleObjectResult*, *var*

**VSet** *Sequence.Object. AngleObjectResult*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

AngleObjectResult specifies the results number.

Default: 1

### Remarks

AngleObjectResult specifies one result for an object set as the AngleObject. For example, create a Blob object for which the NumberToFind value is “4”. Then, by setting Blob object as the AngleObject for the Point object, and setting the AngleObjectResult for the Point object to “2”, the resulting angle for the second Blob object will be used for the Point object.

### See Also

AngleObject Property, ImageOp Object, Point Object

# AngleOffset Property

## Applies To

Vision Objects: Polar, Correlation, Geometric

## Description

An angle value which is used as an offset to line up the search direction indicator (graphic line on the image display) with the part since it is virtually impossible and normally impractical to teach an object with the proper rotation of a part so that the direction lines up with the part.

## Usage

**VGet** *Sequence.Object.AngleOffset, var*

**VSet** *Sequence.Object.AngleOffset, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real number in degrees from 0 - 360

Default: 0

## Remarks

The AngleOffset is used to adjust the rotation position of objects. Detection angle of the object is normally set at the default position which is 0 degrees (at 3 O'clock). For the AngleOffset property, the offset rotation angle in a counter clockwise direction of the detection angle should be set.

## See Also

Object Tab, Polar Object, Correlation Object, Geometric Object

## AngleRange Property

### Applies To

Vision Objects: Correlation, Geometric

### Description

Specifies the rotation detection range.

### Usage

**VGet** *Sequence.Object.AngleRange*, *var*

**VSet** *Sequence.Object.AngleRange*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Correlation: Integer number in degrees from 0 – 180 for PC Vision cameras.  
Integer number in degrees from 0 – 45 for Compact Vision cameras

Default: 10

Geometric: Integer number in degrees from 0 – 180 for all cameras.

Default: 180

### Remarks

Before reading anything else about the AngleRange property it should be noted that the Correlation Model must be taught after a new value for the AngleRange property is set in order for those settings to take affect. If you teach a Correlation Model, then later set the AngleRange property to a new value, and then try to run that Correlation object, it will not search with the new angle range. You must re-teach the Correlation Model with the AngleEnable property set to 1–True, and with the new value for the AngleRange property in order for Correlation search with angle to use the new AngleRange property value. You must also have the proper settings for the AngleMaxIncrement property prior to teaching the new Model as well.

The AngleRange property must be set before teaching the model. This value specifies the range within which to train a series of rotated models. For example, if the AngleRange property is set to 5, then when the model is trained, a set of models is actually trained within +/- 5 degrees of the current model position. These models are then used when a correlation search with angle is specified.

It should be noted that using correlation with angle will generally cause the correlation time to increase significantly. This is why correlation with angle is normally used for small angle increments slightly. On the other hand, Geometric search is suitable for a pattern search which includes large rotation detection. If it is predictable that the parts have a large rotation angle, consider to use Geometric search.

Also, the Polar object is normally very fast and can be used in conjunction with the Correlation object for a powerful and fast combination. (See the sections on Correlation or Polar Searching in the *Vision Guide Manual* for more information.)

Specify a small value for the setting. If the value is large, the detection time gets longer and may result in false detection.

See Also

Angle Result, AngleEnable Property, AngleMaxIncrement Property, Correlation Object, Geometric Object, Object Tab

## AngleStart Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, CodeReader, Contour, Correlation, Geometric

### Description

Sets / returns the starting search angle.

### Usage

**VGet** *Sequence.Object.AngleStart*, *var*

**VSet** *Sequence.Object.AngleStart*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number in degrees from 0 - 360

Default: 0

### Remarks

Use AngleStart to set the starting search angle. This property is only valid when AngleEnable is set to "1-True". The system will search for the model within the AngleRange range centering on the angle specified in AngleStart. For example, if AngleStart is 45 degrees and AngleRange is 10 degrees, then the system will search from 35 to 55 degrees.

For ArcFinder and ArcInspector, AngleStart sets the start of the detection range to find or inspect circular objects. The range is between AngleStart and AngleEnd.

### See Also

Angle Result, AngleEnable Property, AngleEnd Property, AngleMaxIncrement Property, AngleRange Property, ArcFinder Object, ArcInspector Object, Contour Object, Correlation Object, Geometric Object, Object Tab

# ApproachPoint Property

## Applies To

Vision Calibration

CV2 firmware Ver. 3.0.0.0 or later

## Description

Sets / returns the approach point which will be the start point of the robot for moving to each camera point in the calibration.

## Usage

**VGet** *Calibration.ApproachPoint*, *var*

**VSet** *Calibration.ApproachPoint*, *value*

*Calibration*    Name of a calibration or string variable containing a calibration name.

*var*             String variable that will contain the value of the property.

*value*           String expression for the new value of the property.

## Values

Point string

Default: ""

## Remarks

The approach point is the start point of the robot for moving to each camera point in the calibration. This string is used as a parameter for Go command. If you do not use the approach point, specify an empty string ("").

## See Also

VCal

## ArcObject Property

### Applies To

Vision Objects: ArcInspector

### Description

Set an ArcFinder object that is used to locate an arc for ArcInspector to inspect.

### Usage

**VGet** *Sequence.Object.ArcObject*, *var*

**VSet** *Sequence.Object.ArcObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

None, or the name of an ArcFinder object whose step number is prior to the step number of the ArcInspector object.

Default: None

### Remarks

By default, the ArcInspector inspects the arc defined by the Radius, CenterX, and CenterY properties. You can search for the arc with an ArcFinder first by setting the ArcObject property to an ArcFinder.

### See Also

ArcFinder Object, ArcObjResult Property

## ArcObjResult Property

### Applies To

Vision Objects: ArcInspector

### Description

Specifies the result which the ArcObject property uses.

### Usage

**VGet** *Sequence.Object.ArcObjResult, var*

**VSet** *Sequence.Object.ArcObjResult, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

ArcObjResult can be set to All, or you can specify which result to use. By using All, inspection will be executed for each ArcObject result.

Default: 1

### Remarks

ArcObjResult enables you to attach one or more objects to the results of one ArcObject.

### See Also

ArcInspector Object, ArcObject Property

## ArcSearchType Property

### Applies To

Vision Objects: ArcFinder, ArcInspector

### Description

Sets / returns a type of arc (circle, ellipse) to be searched for.

### Usage

**VGet** *Sequence.Object*. **ArcSearchType**, *var*

**VSet** *Sequence.Object*. **ArcSearchType**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Integer expression for the new value of the property.

### Values

0 – Circle Vision constant: VISION\_ARCSEARCHTYPE\_CIRCLE  
Circles will be searched.

2 – Ellipse Vision constant: VISION\_ARCSEARCHTYPE\_ELLIPSE  
Ellipses will be searched.

Default: 0 – Circle

### Remarks

ArcSearchType specifies a type of arc to be searched by ArcFinder, or a type of arc of the base line which is used for ArcInspector.

### See Also

ArcFinder Object, ArcInspector Object, Object Tab

# Area Result

## Applies To

Vision Objects: ArcInspector, Blob, DefectFinder, LineInspector

## Description

Returns the area of a blob or defect.

## Usage

**VGet** *Sequence.Object.Area*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult.

## Values

Real value in pixels from 1 to (SearchWinWidth x SearchWinHeight)

## Remarks

The Area result is the total area of the blob or defect in pixels.

## Statistics

For the Area result, the following statistics are available.

AreaMax, AreaMean, AreaMin, AreaStdDev.

Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

ArcInspector Object, Blob Object, DefectFinder Object, LineInspector Object, MaxArea Property, MinArea Property, MinMaxArea Property, Object Tab

## AsyncMode Property

### Applies To

Vision Sequence

### Description

Defines whether or not to return from VRun command after exposure of the image and before sequence processing executes.

### Usage

**VGet** *Sequence.AsyncMode*, *var*

**VSet** *Sequence.AsyncMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Causes VRun to return after the image acquisition and the sequence processing is completed.

1 – True Causes VRun to return after the image exposure is completed in the camera.

Default: 1 – True

### Remarks

The AsyncMode property lets you choose whether VRun should return after acquiring the image and before sequence processing. When AsyncMode is True, VRun acquires the image and returns to SPEL+. The sequence is then processed by the vision system. This may reduce the cycle time since it allows you to move the robot while the vision sequence is processing. If VRun, VGet, VSet, or any other vision commands are called for the same sequence, they will wait for the previous sequence to be processed before executing.

### See Also

VRun

# AutoCamPoints Property

## Applies To

Vision Calibration

## Description

Defines whether or not to generate camera point automatically.

## Usage

**VGet** *Calibration.AutoCamPoints, var*

**VSet** *Calibration.AutoCamPoints, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Camera point has not been generated.

1 – True Camera point has been generated.

Default: 1 – True

## Remarks

The AutoCamPoints Property is used to set whether or not to enable a function that automatically generates nine camera points for calibration which requires nine camera points. When the function is enabled, nine camera points can be automatically generated when calibration is executed only by specifying one camera point near the center of FOV when teaching points.

## See Also

VCal

# AutoReference Property

## Applies To

Vision Calibration

## Description

Defines whether or not to calculate calibration reference point automatically during the calibration of mobile camera.

## Usage

**VGet** *Calibration.AutoReference, var*

**VSet** *Calibration. AutoReference, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Auto calculation will not be performed.

1 – True Auto calculation will be performed.

Default: 1 – True

## Remarks

When calculating the reference point of the calibration automatically, this property uses the mobile camera to automatically set the calibration reference point. The configured tool and arm settings do not affect this auto setting. If this function is enabled, the tool and arm settings are applied to teaching of the calibration reference point

## See Also

VCal

# AutoRefFinalRotation Property

## Applies To

Vision Calibration

## Description

Sets or returns the final angle of rotation for the tool when automatically calculating the calibration reference point.

## Usage

**VGet** *Calibration.AutoRefFinalRotation, var*

**VSet** *Calibration.AutoRefFinalRotation, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real value from -180 to -5, 5 to 180 (Unit: degrees)

Default: 15

## Remarks

If a positive value is set, the tool will rotate in the direction of the +U axis in the tool coordinate system. If a negative value is set, the tool will rotate in the direction of the -U axis within the tool coordinate system. Absolute values must always be greater than the AutoRefInitRotation. Using large values (90 degrees or more) is recommended to ensure greater reference point accuracy.

Further, as this property is used to set the angle of rotation of the tool, it cannot be used for mobile J2 camera calibration.

## See Also

VCal, VDefTool Statement, AutoRefInitRotation Property

## AutoRefInitRotation Property

### Applies To

Vision Calibration

### Description

Sets or returns the initial small angle of rotation for the arm or tool when automatically calculating the calibration reference point.

### Usage

**VGet** *Calibration.AutoRefInitRotation, var*

**VSet** *Calibration.AutoRefInitRotation, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

For mobile J2 camera calibration

Real value from 0.001 to 45 (Unit: degrees), default value: 5

For mobile J4, J6 camera calibration

Real value from -10 to -0.001, 0.001 to 10 (Unit: degrees), default value: 5

### Remarks

This is the angle of rotation for the arm during mobile J2 camera calibration. This is the angle of rotation for the tool during mobile J4, J6 camera calibration.

If a positive value is set, the tool will rotate in the direction of the +U axis in the tool coordinate system. If a negative value is set, the tool will rotate in the direction of the -U axis within the tool coordinate system. Absolute values must always be smaller than AutoRefFinalRotation.

### See Also

VCal, VDefTool Statement, AutoRefFinalRotation Property

# AutoRefMode Property

## Applies To

Vision Calibration

## Description

Sets or returns modes relating to movement or movement angles when automatically calculating the calibration reference point.

## Usage

**VGet** *Calibration.AutoRefMode, var*

**VSet** *Calibration.AutoRefMode, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

- 1 – Rough Vision constant: `VISION_AUTOREFMODE_ROUGH`  
Performs rough positioning.
- 2 – Fine Vision constant: `VISION_AUTOREFMODE_FINE`  
Performs precise positioning.
- 3 – Manual Vision constant: `VISION_AUTOREFMODE_MANUAL`  
Manually enter the angle to move the robot to perform positioning.

Default: 1 – Rough

## Remarks

For mobile J2 camera calibration

Set this property to Rough to move the robot in small increments. Set this to Fine to move the robot in large increments along with changes in its left and right orientation. Set this to Manual to enter in angles of movement for the robot manually. However, note that the robot will not change its left and right orientation when in Manual mode.

For mobile J4, J6 camera calibration

Set this property to Rough to move the robot in small increments. Set this to Fine to have the robot rotate a tool 180 degrees. Set this to Manual to enter in angles of movement for the robot manually.

## See Also

VCal, VDefArm Statement, VDefTool Statement, AutoRefFinalRotation Property, AutoRefInitRotation Property

## AutoRefTolerance Property

### Applies To

Vision Calibration

### Description

Sets or returns the pixel distance between the vision detection position and the target position that is considered a match when automatically calculating the calibration reference point.

### Usage

**VGet** *Calibration.AutoRefTolerance*, *var*

**VSet** *Calibration.AutoRefTolerance*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real value from 0.1 to 3.0 (Unit: pixels)

Default: 1.0

### Remarks

This sets or returns the tolerance for vision detection deviation. The calibration process will be suspended if vision detection remains unstable, and the level of deviation is above this value. To stabilize vision detection, we recommend increasing the MotionDelay property (the stabilization period after robot movement).

### See Also

VCal, VDefTool Statement, VGoCenter Statement

# CalComplete Result

## Applies To

Vision Calibration

## Description

Returns whether a calibration is completed.

## Usage

**VGet** *Calibration.CalComplete*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Boolean variable that will contain the value of the result.

## Values

0 – False Calibration has not been completed.

1 – True Calibration has been completed.

## Remarks

Use CalComplete to check if a calibration has been completed successfully.

## See Also

PointsTaught Property

# Calibration Property

## Applies To

Vision Sequence

## Description

Sets / returns the calibration name used with the vision sequence.

## Usage

**VGet** *Sequence.Calibration*, *var*

**VSet** *Sequence.Calibration*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* String variable that will contain the value of the property.

*value* Name of a calibration or string expression containing a calibration name.

## Values

String value of up to 16 characters containing the name of the calibration

Default: None

## Remarks

Calibration is required for most vision applications to calculate the proper results for the robot coordinate system and camera coordinate system. The Calibration property associates a previously defined calibration with the specified vision sequence. All calibrations which have been registered will appear in the **Sequence Tab** Calibration property which allows the user to select which calibration to use for this sequence.

Each vision sequence may have only 1 calibration defined at a time. However, if you want to use a sequence with some different calibration data, you can set the Calibration property for the sequence at runtime prior to initiating the vision sequence. For example, you can run the sequence *test* with calibration *calib1* and then run the sequence *test* with calibration *calib2* as follows:

```
VSet test.Calibration, calib1
VRun test
VSet test.Calibration, calib2
VRun test
```

## See Also

Calibration Details, Sequence Tab, Vision Sequences

# CallImageSize Result

## Applies To

Vision Calibration  
CV2 firmware Ver. 3.0.0.0 or later

## Description

Returns a image size in a calibration.

## Usage

**VGet** *Sequence*.**ImageSize**, *var*

*Sequence*     String variable representing a sequence name.

*var*             String variable that will contain the value of the property.

## Values

Defines a constant of image size.

## Remarks

Acquire the image size in the calibration. For details of values, refer to *ImageSize* property.

## See Also

ImageSizepropeprty

# CalRobotPlacePos Property

## Applies To

Vision Objects: Arc Finder, Arc Inspector, ColorMatch, Correlation, Defect Finder, Edge, Geometric, Line Inspector, Polar

## Description

The CalRobotPlacePos property is used to calibrate RobotPlacePos at design time or run time.

## Usage

**VGet** *Sequence.Object.CalRobotPlacePos, var*

**VSet** *Sequence.Object.CalRobotPlacePos, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

True – calibrate the robot place position.

False – invalidate the robot place position calibration

Default: False

## Remarks

CalRobotPlacePos is used to calibrate RobotPlacePos. This property is only valid if the sequence calibration is set to an upward camera calibration.

At design time, clicking the CalRobotPlacePos property button will start the CalRobotPlacePos wizard, which will guide you through the calibration steps.

At run time, setting CalRobotPlacePos to True will calibrate the RobotPlacePos. To calibrate RobotPlacePos at run time, follow these steps:

1. Pick up the part with the robot and move the part over the upward camera
2. Execute VRun to find the part.
3. Jog the robot so that the part is in the place position.
4. Set CalRobotPlacePos = True.
5. Execute VSave to save the calibration.

## See Also

ColorMatch Object, Confusion Property, Correlation Object, Edge Object, Geometric Object, Object Tab, Polar Object, RobotPlacePos Result, Score Result

# Camera Property

## Applies To

Vision Sequence  
Vision Calibration

## Description

Specifies which camera to be used for a vision sequence or vision calibration.

## Usage

**VGet** {*Sequence* / *Calibration*}.**Camera**, *var*

**VSet** {*Sequence* / *Calibration*}.**Camera**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Calibration* Name of a calibration or string variable containing a calibration name.

*Sequence* String variable containing a sequence name.

*Calibration* String variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value equals to or greater than 1.

Default: 1

## Remarks

One camera may be associated with a vision sequence calibration.

For sequences, the camera number must be selected before executing VRun.

For calibrations, the camera number must match the camera number of the calibration target sequence.

### Note:

Camera numbers which can be changed by VSet have the following restrictions.

Sequence or calibration with Compact Vision cameras:

Camera numbers which use the different camera channels of the same Compact Vision camera can be set

Sequence or calibration with PC vision cameras:

Camera numbers which use the different cameras of PC vision can be set.

## Example

The following example shows how to use multiple cameras with the same vision sequence. We will set the Camera property prior to executing the vision sequence called FINDMARK.

## Camera Property

---

```
Function test
#define CAMERA1 1
#define CAMERA2 2
VSet findmark.Camera, CAMERA1
VRun findmark
'Get any info req'd from 1st sequence here (i.e. VGet findmark.xxx.xxx)
VSet findmark.Camera, CAMERA2
VRun findmark
'Get any info req'd from 2nd sequence here (i.e. VGet findmark.xxx.xxx)
Fend
```

### See Also

CameraBrightness Property, CameraContrast Property, Sequence Tab, Vision Sequences

# CameraBrightness Property

## Applies To

Vision Sequence

## Description

Specifies the brightness setting for the camera used in the current sequence.

## Usage

**VGet** *Sequence.CameraBrightness, var*

**VSet** *Sequence.CameraBrightness, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0 - 255

Camera	Default	Application			
		CV1	CV2-S/H	CV2-SA/HA	PC vision
NET 1044 BU	128	✓	✓	✓	-
NET 4133 BU / CU	128	✓	✓	✓	-
NET 1500 BU / CU	128	✓	✓	✓	-
acA640-100gm	16	-	✓	✓	✓
acA640-120gm	16	-	✓	✓	✓
acA1300-60gm	128	-	-	✓	✓
acA1600-20gm / gc	16	-	✓	✓	✓
acA1600-60gm / gc	128	-	✓	✓	✓
acA2500-14gm / gc	32	-	✓	✓	✓
acA2500-20gm / gc	4 / 0	-	-	✓	✓
acA3800-10gm	32	-	-	✓	✓

## Remarks

The CameraBrightness property is normally left at the default value. When a new sequence is created, the default CameraBrightness value is set for the current camera.

Before changing CameraBrightness, try adjusting the lighting and aperture of the lens to obtain the desired brightness. If additional adjustment is necessary, then change the CameraBrightness value. The CameraBrightness property can have values in the range of 0 - 255, with higher values giving more brightness.

## See Also

Camera Property, CameraContrast Property, Sequence Tab, Vision Sequences

# CameraContrast Property

## Applies To

Vision Sequence

## Description

Specifies the contrast setting for the camera used in the current sequence.

## Usage

**VGet** *Sequence.CameraContrast*, *var*

**VSet** *Sequence.CameraContrast*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0 to 255

Camera	Default	Application			
		CV1	CV2-S/H	CV2-SA/HA	PC vision
NET 1044 BU	128	✓	✓	✓	-
NET 4133 BU / CU	128	✓	✓	✓	-
NET 1500 BU / CU	128	✓	✓	✓	-
acA640-100gm	55	-	✓	✓	✓
acA640-120gm	55	-	✓	✓	✓
acA1300-60gm	0	-	-	✓	✓
acA1600-20gm / gc	57	-	✓	✓	✓
acA1600-60gm / gc	0	-	✓	✓	✓
acA2500-14gm / gc	0	-	✓	✓	✓
acA2500-20gm / gc	0	-	-	✓	✓
acA3800-10gm	0	-	-	✓	✓

## Remarks

The CameraContrast property is normally left at the default value. When a new sequence is created, the default CameraContrast value is set for the current camera.

Adjust the value of the CameraContrast property when you want to change the contrast of a image acquisition. The CameraContrast property can have values in the range of 0 - 255, with higher values giving more contrast.

## See Also

Camera Property, CameraBrightnessProperty, Sequence Tab, Vision Sequences

# CameraOrientation Property

## Applies To

Vision Calibration

## Description

Sets / returns the CameraOrientation type for the specified calibration.

## Usage

**VGet** *Calibration.CameraOrientation*, *var*

**VSet** *Calibration.CameraOrientation*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 - Standalone	Fixed camera (VISION_CAMORIENT_STANDALONE)
2 - Fixed Downward	Fixed downward camera (VISION_CAMORIENT_FIXEDDOWN)
3 - Fixed Upward	Fixed upward camera (VISION_CAMORIENT_FIXEDUP)
4 - Mobile on J2	Camera mounted on Joint #2 (VISION_CAMORIENT_MOBILEJ2)
5 - Mobile on J4	Camera mounted on Joint #4 (VISION_CAMORIENT_MOBILEJ4)
6 - Mobile on J5	Camera mounted on Joint #5 (VISION_CAMORIENT_MOBILEJ5)
7 - Mobile on J6	Camera mounted on Joint #6 (VISION_CAMORIENT_MOBILEJ6)

Default: 1 – Standalone

## Remarks

The CameraOrientation property must be set before teaching calibration points.

## See Also

Camera Property, CameraBrightness Property, CameraContrast Property, Sequence Tab, Vision Sequences

## CameraX Result

### Applies To

Vision Objects: ArcFinder, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineInspector, OCR, Point, Polar

### Description

Returns the X position coordinate of the found part's position in the camera coordinate frame.

### Usage

**VGet** *Sequence.Object.CameraX* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number in millimeters.

### Remarks

The CameraX result is always in millimeters in the camera coordinate system.

The CameraX result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraX result will be an error.

### Statistics

For the CameraX result, the following statistics are available. CameraXMax, CameraXMean, CameraXMin, CameraXStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

### See Also

Angle Result, ArcFinder Object, Blob Object, CameraY Result, CameraXYU Result, Correlation Object, DefectFinder Object, Edge Object, Found Result, Geometric Object, Point Object, Polar Object, RobotX Result, CodeReader Object, OCR Object, BoxFinder Object, CornerFinder Object, Contour Object, Object Tab

# CameraX1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

Line, LineFinder: Returns the starting point position (X1) of a Line object in Camera coordinates.

BoxFinder: Returns the X (X1) coordinate position for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraX1*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraX1 result depends upon the calibration used for the camera. Values are always returned in millimeters.

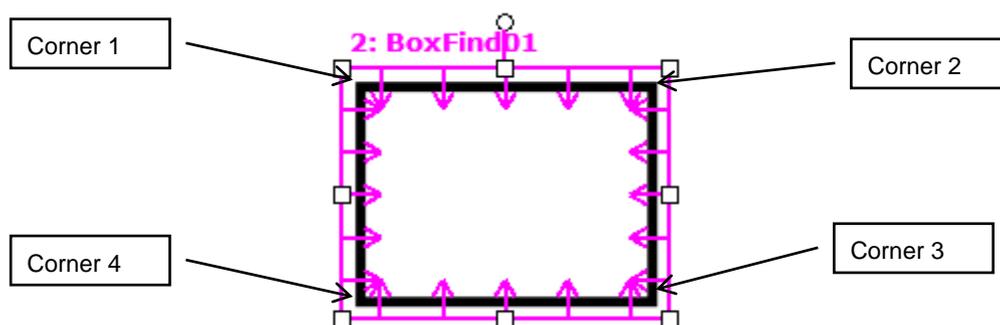
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The CameraX1 and CameraX2 results represent the X coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (CameraX1, CameraY1) and (CameraX2, CameraY2) coordinate pairs can actually be Camera coordinate positions which match the CameraX and CameraY results for other vision objects. (In other words if a Line Object's starting point is defined by a Correlation object, then the (CameraX, CameraY) results from the Correlation object will match the (CameraX1, CameraY1) results for the Line object.)

For BoxFinder

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraX1 is used to retrieve the X coordinate for the Corner1 point shown in the diagram below.



## CameraX1 Result

---

The CameraX1 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraX1 result will be an error.

### See Also

Angle Result, CameraX2 Result, CameraY1 Result, CameraX3 Result, CameraY3 Result, CameraX4 Result, CameraY4 Result, CameraY2 Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, RobotX Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object,

# CameraX2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

Line, LineFinder: Returns the end point position (X2) of a Line object in Camera coordinates.

BoxFinder: Returns the X coordinate position (X2) for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraX2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraX2 result depends upon the calibration used for the camera. Values are always returned in millimeters.

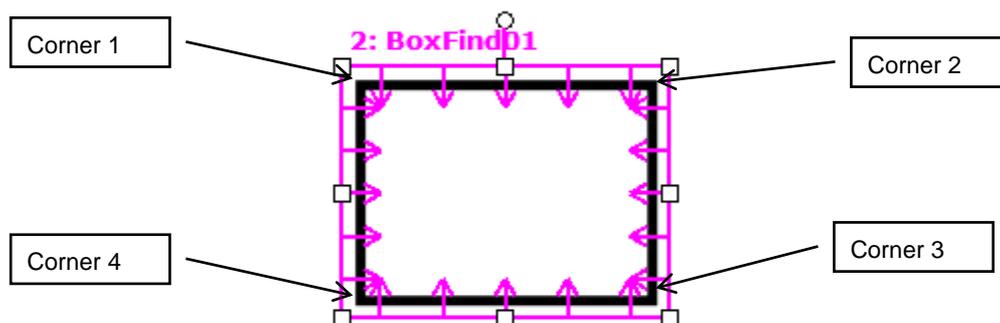
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The CameraX1 and CameraX2 results represent the X coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (CameraX1, CameraY1) and (CameraX2, CameraY2) coordinate pairs can actually be Camera coordinate positions which match the CameraX and CameraY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (CameraX, CameraY) results from the Correlation object will match the (CameraX1, CameraY1) results for the Line object.)

For BoxFinder

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraX2 is used to retrieve the X coordinate for the Corner2 point shown in the diagram below.



## CameraX2 Result

---

The CameraX2 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraX2 result will be an error.

### See Also

Angle Result, CameraX1 Result, CameraY1 Result, CameraY2 Result, CameraX3 Result, CameraY3 Result, CameraX4 Result, CameraY4 Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX2 Result, RobotX Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# CameraX3 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the X coordinate position (X3) for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraX3*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

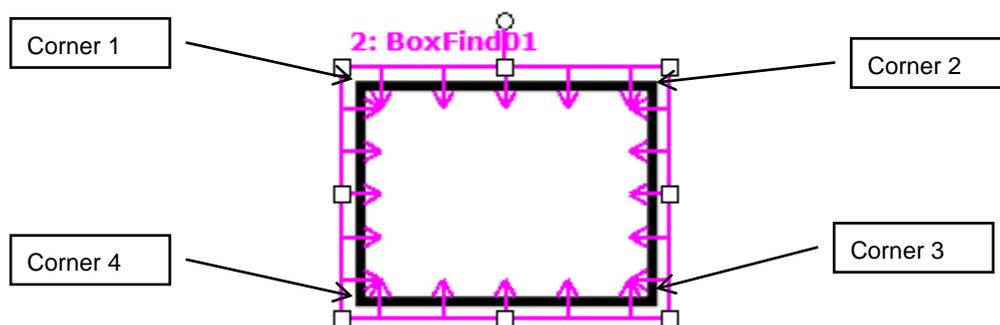
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraX3 result depends upon the calibration used for the camera. Values are always returned in millimeters.

## Remarks

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraX3 is used to retrieve the X coordinate for the Corner3 point shown in the diagram below.



The CameraX3 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraX3 result will be an error.

## See Also

CameraX1 Result, CameraX2 Result, CameraY1 Result, CameraY2 Result, CameraY3 Result, CameraX4 Result, CameraY4 Result, BoxFinder Object, Object Tab

# CameraX4 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the X coordinate position (X4) for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraX4*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

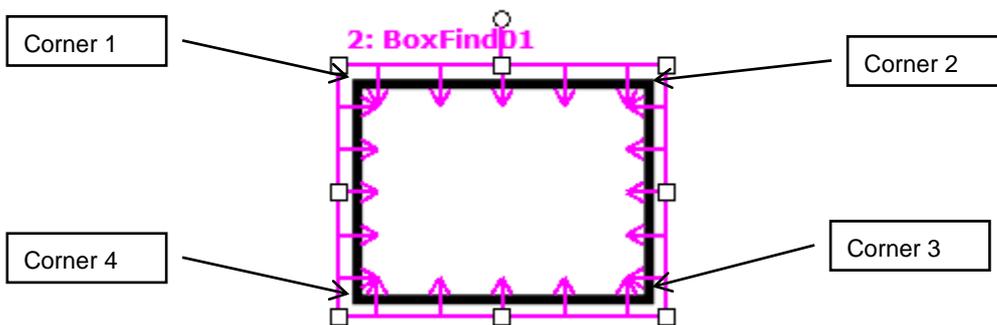
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraX4 result depends upon the calibration used for the camera. Values are always returned in millimeters.

## Remarks

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraX4 is used to retrieve the X coordinate for the Corner4 point shown in the diagram below.



The CameraX4 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraX4 result will be an error.

## See Also

CameraX1 Result, CameraX2 Result, CameraY1 Result, CameraY2 Result, CameraX3 Result, CameraY3 Result, CameraY4 Result, BoxFinder Object, Object Tab

# CameraXYU Result

Runtime Only

## Applies To

Vision Objects: ArcFinder, Blob, BoxFinder, CornerFinder, Correlation, Edge, Geometric, LineInspector, Point, Polar

## Description

Returns the CameraX, CameraY and Angle position coordinates of the found part's position in the camera coordinate frame.

## Usage

**VGet** *Sequence.Object.CameraXYU* [(*result*)], *found*, *xVar*, *yVar*, *uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the part you are looking for was found.

*xVar* Real variable that will contain the X coordinate position of the part.

*yVar* Real variable that will contain the Y coordinate position of the part.

*uVar* Real variable that will contain the angular position (rotation) of the part with respect to the camera coordinate system

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

*found* Boolean value which is either True or False

*xVar* Real number in millimeters

*yVar* Real number in millimeters

*uVar* Real number in degrees

## Remarks

The *xVar* and *yVar* values are returned in millimeters since the Camera Coordinate Frame is calibrated in millimeters. The *uVar* value is returned in degrees.

It should be noted that the CameraXYU result can only be calculated for vision sequences which have been associated with a calibration. If no calibration has been assigned to the vision sequence then if VGet is used to retrieve the CameraXYU result, an error will occur.

The CameraXYU result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraXYU result will be an error.

The CameraXYU result is available at runtime only.

## See Also

Angle Result, ArcFinder Object, Blob Object, CameraX Result, CameraY Result, Correlation Object, Edge Object, Found Result, Geometric Object, Point Object, Polar Object, BoxFinder Object, CornerFinder Object, RobotXYU Result

## CameraY Result

### Applies To

Vision Objects: ArcFinder, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, Edge, Geometric, LineInspector, OCR, Point, Polar

### Description

Returns the Y position coordinate of the found part's position in the camera coordinate frame.

### Usage

**VGet** *Sequence.Object.CameraY* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number in millimeters

### Remarks

The CameraY result is always in millimeters in the camera coordinate system.

The CameraY result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraY result will be an error.

### Statistics

For the CameraY result, the following statistics are available. CameraYMax, CameraYMean, CameraYMin, CameraYStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

### See Also

Angle Result, ArcFinder Object, Blob Object, CameraX Result, CameraXYU Result, Correlation Object, Edge Object, Found Result, Geometric Object, Point Object, Polar Object, RobotXYU Result, RobotY Result, CodeReader Object, OCR Object, BoxFinder Object, CornerFinde Object, Contour Object, Object Tab

# CameraY1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the starting point position Y coordinate (Y1) of a Line object in Camera coordinates.

**BoxFinder:** Returns the Y (Y1) coordinate position for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraY1*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters

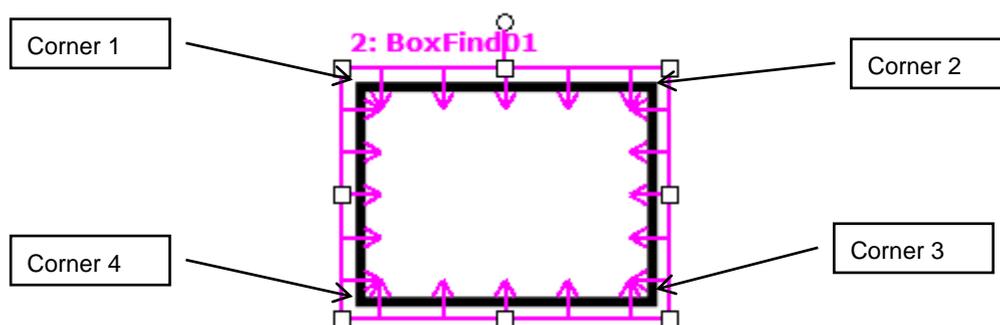
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The CameraY1 and CameraY2 results represent the Y coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (CameraX1, CameraY1) and (CameraX2, CameraY2) coordinate pairs can actually be Camera coordinate positions which match the CameraX and CameraY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (CameraX, CameraY) results from the Correlation object will match the (CameraX1, CameraY1) results for the Line object.)

For BoxFinder

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraY1 is used to retrieve the Y coordinate for the Corner1 point shown in the diagram below.



## CameraY1 Result

---

The CameraY1 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraY1 result will be an error.

### See Also

Angle Result, CameraX1 Result, CameraX2 Result, CameraY2 Result, CameraX3 Result, CameraY3 Result, CameraX4 Result, CameraY4 Result, Line Object, Object Tab, PixelX Result, PixelY2 Result, RobotY Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# CameraY2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the ending point position (Y2) of a Line object in Camera coordinates.

**BoxFinder:** Returns the Y (Y2) coordinate position for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraY2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraY2 result depends upon the calibration used for the camera. Values are always returned in millimeters.

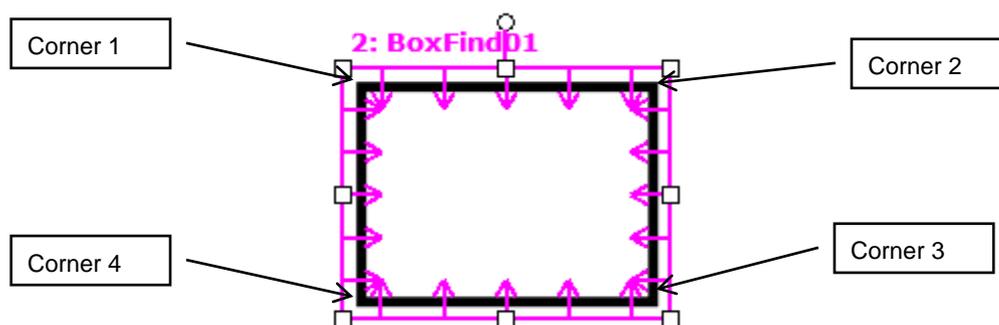
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The CameraY1 and CameraY2 results represent the Y coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (CameraX1, CameraY1) and (CameraX2, CameraY2) coordinate pairs can actually be Camera coordinate positions which match the CameraX and CameraY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (CameraX, CameraY) results from the Correlation object will match the (CameraX1, CameraY1) results for the Line object.)

For BoxFinder

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraY2 is used to retrieve the Y coordinate for the Corner2 point shown in the diagram below.



## CameraY2 Result

---

The CameraY2 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraY2 result will be an error.

### See Also

Angle Result, CameraX1 Result, CameraX2 Result, CameraY1 Result, CameraX3 Result, CameraY3 Result, CameraX4 Result, CameraY4 Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelY2 Result, RobotY Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# CameraY3 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y3) coordinate position for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraY3*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

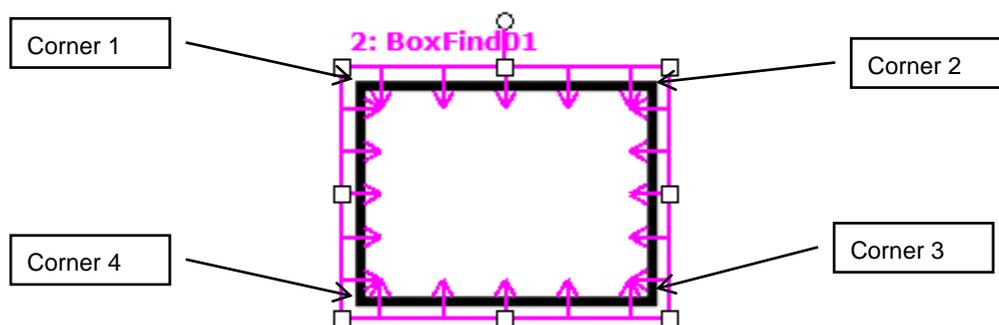
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraY3 result depends upon the calibration used for the camera. Values are always returned in millimeters.

## Remarks

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraY3 is used to retrieve the Y coordinate for the Corner3 point shown in the diagram below.



The CameraY3 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraY3 result will be an error.

## See Also

CameraX1 Result, CameraX2 Result, CameraY1 Result, CameraY2 Result, CameraX3 Result, CameraX4 Result, CameraY4 Result, BoxFinder Object, Object Tab

# CameraY4 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y4) coordinate position for corner points of rectangles detected in a camera coordinate system.

## Usage

**VGet** *Sequence.Object.CameraY4*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

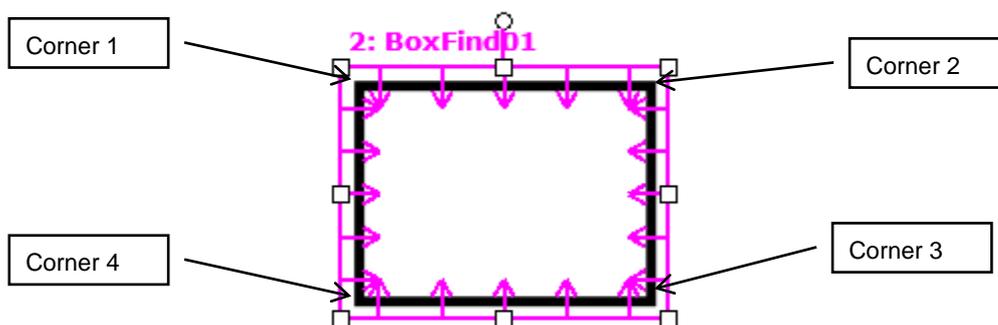
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

The value returned for the CameraY4 result depends upon the calibration used for the camera. Values are always returned in millimeters.

## Remarks

The camera coordinates for the four corners of a rectangle can be retrieved as Camera X1, 2, 3, 4 results and Camera Y1, 2, 3, 4 results. CameraY4 is used to retrieve the Y coordinate for the Corner4 point shown in the diagram below.



The CameraY4 result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the CameraY4 result will be an error.

## See Also

CameraX1 Result, CameraX2 Result, CameraY1 Result, CameraY2 Result, CameraX3 Result, CameraY3 Result, CameraX4 Result, BoxFinder Object, Object Tab

# Caption Property

## Applies To

Vision Objects: All

## Description

Sets or returns the text to be displayed for an object's label on the screen.

## Usage

**VGet** *Sequence.Object.Caption*, *var*

**VSet** *Sequence.Object.Caption*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

String of 16 characters or less. Any alphanumeric or the following punctuation characters are allowed.

' \_ ( ) \* & \$ # @ . : \ / < > "

Default: Empty string

## Remarks

The Caption property allows you to assign an arbitrary label to the vision object. As default, nothing is set to the Caption property. The name of the object will be used as its label on the screen.

## See Also

ArcFinder Object, ArcInspector Object, Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Edge Object, Frame Object, Geometric Object, ImageOp Object, Line Object, LineFinder Object, LineInspector Object, Object Tab, OCR Object, Point Object, Polar Object

## CenterPntObjResult Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, OCR, Polar, Text

### Description

Specifies the result which the CenterPointObject property uses.

### Usage

**VGet** *Sequence.Object.CenterPntObjResult*, *var*

**VSet** *Sequence.Object.CenterPntObjResult*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

CenterPntObjResult can be set to “All”, or you can specify which result to use. By using “All”, results will be created for each CenterPointObject result.

Default: 1

### Remarks

CenterPntObjResult enables you to attach several objects to the results of one CenterPointObject. For example, you could create a blob object with NumberToFind set to 4. Then you could execute a polar object to each one of the results by specifying the blob for the CenterPointObject of each polar object, and specifying “All” to CenterPntObjResult of the blob object.

### See Also

ArcFinder Object, ArcInspector Object, Blob Object, CenterPointObject Property, CenterX Property, CenterY Property, CodeReader Object, Correlation Object, DefectFinder Object, Edge Object, Geometric Object, OCR Object, Object Tab, Polar Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

# CenterPntRotOffset Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, OCR, Polar

## Description

Specifies whether the center point XY offset (CenterPntOffsetX, CenterPntOffsetY) should be rotated according to the Angle result of CenterPointObject.

## Usage

**VGet** *Sequence.Object.CenterPntRotOffset*, *var*

**VSet** *Sequence.Object.CenterPntRotOffset*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

True – rotate the offset

False – do not rotate the offset

Default: False

## Remarks

Set CenterPntRotOffsets to True if you want the XY offset value (CenterPntOffsetX, CenterPntOffsetY) to be rotated by the Angle result of the CenterPointObject.

## See Also

CenterPointObject Property, CenterPntOffsetX Property, CenterPntOffsetY Property, SearchWinType Property

## CenterPntOffsetX Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, OCR, Polar, Text

### Description

Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.

### Usage

**VGet** *Sequence.Object.CenterPntOffsetX*, *var*

**VSet** *Sequence.Object.CenterPntOffsetX*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer in pixels. As a result of this property setting, the search can be done within the valid pixels, even if the search window would not be positioned in the camera view.

Default: 0

### Remarks

The CenterPntOffsetX property can be used to shift a search window that has been positioned by a CenterPointObject in X direction.

If CenterPointObject property is set to None, then CenterPntOffsetX has no effect.

### See Also

Blob Pbject, CenterPointObject, CenterPntOffsetY, CodeReader Object, ColorMatch Object, Correlation Object, Geometric Object, OCR Object, Object Tab, Polar Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

## CenterPntOffsetY Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, OCR, Polar, Text

### Description

Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.

### Usage

**VGet** *Sequence.Object.CenterPntOffsetY*, *var*

**VSet** *Sequence.Object.CenterPntOffsetY*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer in pixels. As a result of this property setting, the search can be done within the valid pixels, even if the search window would not be positioned in the camera view.

Default: 0

### Remarks

The CenterPntOffsetY property can be used to shift a search window that has been positioned by a CenterPointObject in Y direction.

If CenterPointObject property is set to None, then CenterPntOffsetY has no effect.

### See Also

Blob Object, CenterPointObject, CenterPntOffsetX, CodeReader Object, ColorMatch Object, Correlation Object, Geometric Object, OCR Object, Object Tab, Polar Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

## CenterPointObject Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, OCR, Polar, Text

### Description

Specifies the object whose position is used as the center for the specified object.

### Usage

**VGet** *Sequence.Object.CenterPointObject*, *var*

**VSet** *Sequence.Object.CenterPointObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

Name of a vision object or screen.

Default: Screen

### Remarks

The CenterPointObject is based on the resulting coordinate position of an object which is executed prior to the specified object.

When teaching an object which has the CenterPointObject setting other than “Screen”, the vision object defined as CenterPointObject is executed first, and the position results (PixelX and PixelY) are used to determine the current object position. Therefore, to detect the current object, the object defined as the CenterPointObject should be found correctly.

### See Also

Blob Object, CenterX Property, CenterY Property, CodeReader Object, ColorMatch Object, Correlation Object, Geometric Object, Object Tab, OCR Object, Polar Object, ArcFinder Object, LineFinder Object, ArcInspector Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

# CenterX Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Contour, Polar

## Description

Specifies the X Coordinate position to be used as the center point of an object in pixels.

## Usage

**VGet** *Sequence.Object.CenterX*, *var*

**VSet** *Sequence.Object.CenterX*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real number from 0 – (video width in pixels – 1). However, the detection range of the Polar object cannot exceed the valid pixel range of the camera. Therefore, the actual range is:

$(0 + \text{Radius}) - (\text{Current Pixel Coordinate Y Resolution} - \text{Radius} - 1)$

Radius is a distance from the center of the Polar object to the outer side of the reach ring for the object.

For example, if an image resolution of ImageSize property is 640x480 and a radius is 16 pixels, it gives a range of 16 - 623.

Default: X coordinate screen position of the center of the Polar object

## Remarks

This property is filled in automatically when the CenterPointObject property for a Polar object is set to another vision object. However, if the CenterPointObject property for a Polar object is set to Screen, then the user may set the CenterX property to position the Polar object.

The user may also set the CenterPointObject property for a Polar object automatically by physically dragging the Polar object to a new position on the screen. When this drag operation is done, the CenterX property is automatically updated with the new CenterX position of the Polar object.

## See Also

ArcFinder Object, ArcInspector Object, Contour Object, CenterY Property, CenterPoint Property, Object Tab, Polar Object

## CenterY Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, Contour, Polar

### Description

Specifies the Y Coordinate position to be used as the center point for the Polar object.

### Usage

**VGet** *Sequence.Object.CenterY*, *var*

**VSet** *Sequence.Object.CenterY*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real number from 1 – (video width in pixels – 1). However, the detection range of the Polar object cannot exceed the valid pixel range of the camera. Therefore, the actual range is:

$(0 + \text{Radius}) - (\text{Current Pixel Coordinate Y Resolution} - \text{Radius} - 1)$

Radius is a distance from the center of the Polar object to the outer side of the reach ring for the object.

For example, if an image resolution of ImageSize is 640x480 and a radius is 16 pixels, it gives a range of 16 - 463.

Default: Y coordinate Screen Position of the center of the Polar object

### Remarks

This property is filled in automatically when the CenterPointObject property for a Polar object is set to another vision object. However, if the CenterPointObject property for a Polar object is set to Screen, then the user may set the CenterY property to position the Polar object.

The user may also set the CenterPointObject property for a Polar object automatically by physically dragging the Polar object to a new position on the screen. When this drag operation is done, the CenterY property is automatically updated with the new CenterY position of the Polar object.

### See Also

ArcFinder Object, ArcInspector Object, Contour Object, CenterX Property, CenterPoint Property, Object Tab, Polar Object

# CharToTeach Property

Runtime Only

## Applies To

Vision Objects: OCR

## Description

Sets / returns the character used by VTeach.

## Usage

**VGet** *Sequence.Object.CharToTeach*, *var*

**VSet** *Sequence.Object.CharToTeach*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

String for one character.

Default: ""

## Remarks

The CharToTeach property specifies the character that will be taught when VTeach executes. Before executing VTeach, the model window must surround the character being taught.

## See Also

InvalidChar, Object Tab, OCR Object, VTeach

## CodabarChecksumEnabled Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to use the checksum during Codabar detection.

### Remarks

Sets whether to use the checksum during Codabar detection.

Default: False

### See Also

CodeReading Object, Object Tab, CodabarOutputChecksum Property, CodabarOutputStartStop Property

## CoderbarOutputChecksum Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the checksum value in the Codabar Text result.

### Remarks

Sets whether to include the checksum value in the Codabar Text result.

Default: False

### See Also

CodeReading Object, Object Tab, CodabarChecksumEnabled Property, CodabarOutputStartStop Property, Text Result

## CodabarOutputStartStop Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the start and stop characters in the Codabar Text result.

### Remarks

Sets whether to include the start and stop characters in the Codabar Text result.

Default: True

### See Also

CodeReading Object, Object Tab, CodabarOutputChecksum Property, CodabarOutputStartStop Property, Text Result

## Code39ChecksumEnabled Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to use the checksum for CODE 39 detection.

### Remarks

Sets whether to use the checksum for CODE 39 detection.

Default:False

### See Also

CodeReading Object, Object Tab, Code39OutputChecksum Property, Code39OutputStartStop Property

## Code39OutputChecksum Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the checksum value in the CODE 39 Text result.

### Remarks

Sets whether to include the checksum value in the CODE 39 Text result.

Default: True

### See Also

CodeReading Object, Object Tab, Code39OutputChecksum Property, Code39OutputStartStop Property, Text Result

## Code39OutputStartStop Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the start and stop characters in the CODE 39 Text result.

### Remarks

Sets whether to include the start and stop characters in the CODE 39 Text result.

Default: True

### See Also

CodeReading Object, Object Tab, Code39ChecksumEnabled Property, Code39OutputChecksum Property, Text Result

## Code128OutputChecksum Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the checksum value in the CODE 128 Text result.

### Remarks

Sets whether to include the checksum value in the CODE 128 Text result.

Default: True

### See Also

CodeReading Object, Object Tab, Text Result

# CodeType Property

## Applies To

Vision Objects: CodeReader

## Description

Sets / returns which type of bar codes or matrix codes to search for with the CodeReader object.

## Usage

**VGet** *Sequence.Object.CodeType*, *var*

**VSet** *Sequence.Object.CodeType*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Setting value	Vision constant	Remarks
Auto	VISION_CODETYPE_AUTO	Automatic code detection
DataMatrix	VISION_CODETYPE_DATAMATRIX	2D code
Codabar	VISION_CODETYPE_CODABAR	
Code39	VISION_CODETYPE_CODE39	
Code128	VISION_CODETYPE_CODE128	
EAN 8	VISION_CODETYPE_EAN8	
EAN 13	VISION_CODETYPE_EAN13	
Interleaved 2 of 5	VISION_CODETYPE_INTERLEAVED25	Also called ITF
PDF417	VISION_CODETYPE_PDF417	2D code
QR	VISION_CODETYPE_QR	2D code
UPC	VISION_CODETYPE_UPC	
UPC A	VISION_CODETYPE_UPCA	
UPC E	VISION_CODETYPE_UPCE	

Default: Auto

## Remarks

CodeType specifies the type of bar codes (one- or two-dimensional) to be searched for by the CodeReader object.

CodeType is set to “Auto” as default, and the code types are detected automatically.

## See Also

CodeReader Object, Found Result, FoundCodeType Result, Object Tab

## ColorIndex Result

### Applies To

Vision Objects: ColorMatch

### Description

Returns the index of the color model found to be the best match.

### Usage

**VGet** *Sequence.Object.ColorIndex* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult

### Values

Index of the matched color model.

### Remarks

The ColorIndex result is the index of the matched color model. The name given to the color model of the best matched model can be acquired from the ColorName result.

### See Also

ColorMatch Object, ColorName Result, ColorValue Result, Object Tab

# ColorMode Property

## Applies To

Vision Objects: ColorMatch, ImageOp

## Description

Sets the desired color mode (RBG or HSV).

## Usage

**VGet** *Sequence.Object.ColorMode, var*

**VSet** *Sequence.Object.ColorMode, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 – RGB

2 - HSV

Default: 1 (RGB)

## Remarks

ColorMode is used to set the color space used for the search in ColorMatch and for the ColorFilter ImageOp operation.

## See Also

ColorMatch Object, ImageOp Object, ModelColorTol Property

## ColorName Result

### Applies To

Vision Objects: ColorMatch

### Description

Returns the name of the color model found to be the best match.

### Usage

**VGet** *Sequence.Object.ColorName* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the CurrentResult

### Values

A string containing the name of the color model.

Default: Empty string

### Remarks

ColorName is the name given to the color model of the best matched model. The ColorIndex result is the index of the matched model. The name of a color model can be changed at design time from the [Vision Guide] window by clicking the <Teach> button for the object and changing the color model names. The model name can also be changed by setting the ModelName property from SPEL+ at runtime.

### See Also

ColorMatch Object, ColorIndex Result, ColorValue Result, ModelName Property, Object Tab

# ColorValue Result

## Applies To

Vision Objects: ColorMatch

## Description

Returns the RGB or HSV value of the found color and returns the gray value for a monochrome image.

## Usage

**VGet** *Sequence.Object.ColorValue* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the value of CurrentResult.

## Values

For RGB, the format is a 6-digit hexadecimal number (&Hrrggbb for red, green, and blue).

For HSV, the format is a 7-digit hexadecimal number (&Hhhssvv for hue, saturation, and value).

For monochrome images, ColorValue returns the gray value within a range of 0 - 255

## Remarks

ColorValue returns the actual RGB or HSV value of the found color. Normally, ColorIndex is used to determine which color model had the best match. ColorValue returns the actual color that was found.

## See Also

ColorMatch Object, ColorMode Property, Point Object, ColorIndex Result, ColorName Result, Object Tab

## Compactness Result

### Applies To

Vision Objects: Blob, DefectFinder

### Description

Returns the compactness of a blob or defect.

### Usage

**VGet** *Sequence.Object.Compactness* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult

### Values

Minimum value is 1.0.

### Remarks

Compactness is a measure of how close all particles in a blob are from one another. It is derived from the perimeter and area. A circular blob is most compact and is defined to have a compactness measure of 1.0 (the minimum). More convoluted shapes have larger Compactness values.

### See Also

Blob Object, DefectFinder Object, Holes Result, Object Tab, Perimeter Result, Roughness Result

# Confusion Property

## Applies To

Vision Objects: Correlation, Geometric, Polar

## Description

Indicates the amount of confusion expected in the image to be searched. This is the highest shape score a feature can get that is not an instance of the feature for which you are searching. (i.e. Will there be patterns in the image which will "confuse" the searching algorithms? To what level?)

## Usage

**VGet** *Sequence.Object.Confusion*, *var*

**VSet** *Sequence.Object.Confusion*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer number from 1-999 with the higher numbers representing a higher confusion in an image.

Default: 800

## Remarks

Both the Confusion property and the Accept property affect the pattern finding speed.

The search continues until the number of the score patterns specified in NumberToFind which are larger than the Confusion value are found. If the score pattern larger than Accept value and smaller than Confusion is found, the search continues for the higher score pattern, and then confirms the pattern search result if there is no higher score pattern.

The Confusion property allows the system to abort the search before exploring all possible regions of the image, thus speeding up the process.

Set the Confusion property based on the highest value you expect the "wrong part" to get (plus a margin for error). It is recommended to set The Confusion property greater than or equal to the Accept property. Setting the Confusion property to a high value may increase the time of the search, while setting the property to a low value may decrease the time of the search. However, note that the low Confusion property value may increase the possibility of false detection.

The Confusion property becomes very important when there are multiple features within an image which are very similar. In these cases the proper setting of the Confusion property (i.e. at a high enough level) helps eliminate the wrong features. However, when an image has few features which look the same then the Confusion property can be set a little lower. This can help reduce processing time.

## See Also

Accept Property, Correlation Object, Geometric Object, Object Tab, Polar Object, Score Result

# ContourMode Property

## Applies To

Vision Object: Contour  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Defines the edge detection method for Contour objects.

## Usage

**VGet** *Sequence.Object.ContourMode*, *var*

**VSet** *Sequence.Object.ContourMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 – Blob Vision constant: `VISIOIN_CONTOURMODE_BLOB`  
Use Blob mode.

2 – Line Vision constant: `VISIOIN_CONTOURMODE_LINE`  
Use Line mode.

3 – Arc Vision constant: `VISION_CONTOURMODE_ARC`  
Use Arc mode.

Default: 1

## Remarks

Defines the edge detection method for Contour objects.

### Blob mode:

Detects work pieces in a search window as a blob and outputs its contour. This can be used to retrieve the contour from complex-shaped work pieces.

### Line mode:

Detects edges with multiple edge search lines positioned side by side, and sets these edges as contour points. This can be used to retrieve contours with minimal unevenness along parts of a work piece.

### Arc mode:

Detects edges with multiple edge search lines positioned radially, and sets these edges as contour points. This can be used to retrieve contours on circular arcs on a work piece with minimal unevenness.

## See Also

Contour Object

# ContourTolerance Property

## Applies To

Vision Object: Contour  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Specifies the tolerance for reducing contour points.

## Usage

**VGet** *Sequence.Object.ContourTolerance, var*

**VSet** *Sequence.Object. ContourTolerance, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

## Values

Set as a positive real value from 0 to 100

## Remarks

The number of contour points can be reduced for Contour objects. While lowering the ContourTolerance property value increases the accuracy that a trajectory can be traced for a work piece, this results in a greater number of contour points. Conversely, increasing the ContourTolerance property value reduces the trajectory accuracy, but also reduces the number of contour points. Contour points will not be deleted if the ContourTolerance property is set to 0.

## See Also

Contour Object, SamplingPitch Property

## Contrast Result

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, CornerFinder, Edge, LineFinder, LineInspector

### Description

Returns the contrast of the found edge.

### Usage

**VGet** *Sequence.Object.Contrast*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult.

### Values

Integer number from 0 - 255

### Remarks

Contrast is the difference in grayscale values between an edge and its background. Contrast can help find weaker edges. First, find the edge you want to search for and record the contrast value. Next, set the ContrastTarget property to this value. Then set the ScoreWeightContrast to a higher value than ScoreWeightStrength. This tells the Edge object to look for an edge with the desired contrast and base the score on it. For ArcFinder, ArcInspector, LineFinder, LineInspector objects, the Contrast result is the average contrast of all edges used in the search.

### See Also

ArcFinder Object, ArcInspector Object, ContrastTarget Property, ContrastVariation Property, Edge Object, LineFinder Object, LineInspector Object, BoxFinder Object, CornerFinder Object

# ContrastTarget Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

## Description

Sets the desired contrast for the edge search.

## Usage

**VGet** *Sequence.Object.ContrastTarget*, *var*

**VSet** *Sequence.Object.ContrastTarget*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - 255

Default: 0 (any contrast)

## Remarks

ContrastTarget is the expected difference in grayscale values between an edge and its background. Use ContrastTarget to find weaker edges or edges at an angle. First, find the edge you want to search for and record the Contrast result value. You may have to temporarily change the Edge object position to find it. Next, set the ContrastTarget property to this value. Then set the ScoreWeightContrast to a higher value then ScoreWeightStrength. This tells the Edge object to look for an edge with the desired contrast and base the score on it.

## See Also

Contrast Result, ContrastVariation Property, Edge Object, LineFinder Object, ArcFinder Object, ArcInspector Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object

## ContrastVariation Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

### Description

Sets the amount of contrast variation for the ContrastTarget property.

### Usage

**VGet** *Sequence.Object.ContrastVariation*, *var*

**VSet** *Sequence.Object.ContrastVariation*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number from 0 - 255

Default: 0 (any variation)

### Remarks

Use ContrastVariation to set the amount of contrast variation allowed for the edge search with the value set in ContrastTarget.

### See Also

ArcFinder Object, ArcInspector Object, Contrast Result, ContrastTarget Property, Edge Object, LineFinder Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object

# Count Property

## Applies To

Vision Sequence Objects

## Description

Returns the number of sequences, or the number of sequence objects.

## Usage

**VGet** Sequences.**Count**, *var*

**VGet** *Sequence*.Objects.**Count**, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

## Values

An integer that shows the number of sequences, or the number of objects for the sequence specified

## Remarks

Use the Count property to determine how many objects are in a sequence when you want to iterate through all objects.

## See Also

Objects Property, Sequences Property

## CurrentModel Property

Runtime Only

### Applies To

Vision Objects: ColorMatch, ImageOp

### Description

Sets / returns the current model index to change the teaching and model names.

### Usage

**VGet** *Sequence.Object.CurrentModel*, *var*

**VSet** *Sequence.Object.CurrentModel*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number from 1 to the NumberOfModels result value.

### Remarks

Use the CurrentModel property to set the current model index for teaching and for setting the names of color models.

For example, the following code sets the color and the name of model 2:

```
VSet seq1.ColorMatch01.CurrentModel, 2
VSet seq1.ColorMatch01.ModelColor, &Hff0000
VSet seq1.ColorMatch01.ModelName, "Red"
```

### See Also

NumberOfModels Property, ColorMatch Object, ImageOp Object, ModelColor Property

# CurrentResult Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Edge, Frame, Geometric, Line, LineFinder, LineInspector, OCR, Polar, Point, Text

## Description

Defines which result to display in the Results list on the Object tab or which result to return data for when an object searches for multiple results.

## Usage

**VGet** *Sequence.Object.CurrentResult*, *var*

**VSet** *Sequence.Object.CurrentResult*, *value*

**Sequence** Name of a sequence or string variable containing a sequence name.

**Object** Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer number from 1 – NumberOfResults *value*.

Default: 1

## Remarks

Several objects support finding multiple results by setting NumberToFind equal to a value greater than one. The CurrentResult property defines which one of the found results to work with.

When you are only trying to find one result (as defined by the NumberToFind property), the CurrentResult property is automatically set to one since there is only one possible result to return.

When working with the Vision Guide window, you will also notice that the Results list on the Object tab will display a heading like "Result (1 of 15)". This means that the system tried to find 15 features (as defined by the NumberToFind property) and the Results list displays the results for item 1.

If you want to see the results for one of the other results, just change the CurrentResult property value to indicate which result you want to examine.

Results are ordered according to the Sort property setting.

ArcInspector, Blob, DefectFinder, and LineInspector objects results are displayed in descending order of blob size depending on SizeToFind. For Result 1 (CurrentResult = 1), the largest blob result will be displayed.

Correlation object, Edge, and Geometric objects results are displayed in descending order of scores as default. For Result 1 (CurrentResult = 1), the result of the highest score will be displayed.

## See Also

ArcFinder Object, ArcInspector Object, Blob Object, Correlation Object, DefectFinder Object, Edge Object, Found Result, Geometric Object, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object, NumberFound Result, NumberToFind Property, Object Tab, Sort Property

### Example

The following SPEL<sup>+</sup> language example runs a vision sequence called "mtest" which contains a Blob object called "Blob01". "Blob01" has been defined to find multiple blobs (3) from within a single Search Window. (i.e. mtest.Blob01.NumberToFind = 3)

The following program will run the sequence and make sure that the proper number of features (3) was found for "Blob01" and then print the Area for each result.

```
Function main
  #define NUM_TO_FIND 3
  Integer foundCount, area
  VRun mtest
  VGet mtest.Blob01.NumberFound, foundCount
  If foundCount = NUM_TO_FIND Then
    Print "The correct number of blobs were found"
  Else
    Print "Only (", found, ") blobs were found"
  EndIf
  VSet mtest.Blob01.CurrentResult, 1
  VGet mtest.BLOB01.Area, area
  Print "1st blob area =", area, "pixels"

  VSet mtest.Blob01.CurrentResult, 2
  VGet mtest.Blob01.Area, area
  Print "2nd blob area =", area, "pixels"

  VSet mtest.Blob01.CurrentResult, 3
  VGet mtest.Blob01.Area, area
  Print "3rd blob area =", area, "pixels"
Fend
```

## DataMatrixConnectDots Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Specifies whether or not to pre-process connected dots for a DataMatrix code using round dots.

When using the round cells, specify to pre-process if the adjacent cells are connected each other.

### Values

Setting range: True / False

Default: False

### See Also

CodeReader Object, DataMatrixMinLength Property, DataMatrixPolarity Property

## DataMatrixMinLength Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets the minimum DataMatrix code size.

### Values

Integer from 36 to 999 pixels

Default: 46

### Remarks

Reducing the value may enable detection of small DataMatrix codes. However, detection time will be longer.

### See Also

CodeReader Object, DataMatrixConnectDots Property, DataMatrixPolarity Property

# DataMatrixPolarity Property

Design time only

## Applies To

Vision Object: CodeReader

## Description

Specifies the polarity of the DataMatrix code to search for.

## Values

0 – Black DataMatrix

1 – White DataMatrix

2 – Both black and white DataMatrix

Default: 0 - Black

## See Also

CodeReader Object, DataMatrixConnectDots Property, DataMatrixMinLength Property

## DefectAreaExtended Property

### Applies To

Vision Objects: ArcInspector, LineInspector

### Description

Sets / gets whether to use edge results within the defect thresholds to extend the area of the defect.

### Usage

**VGet** *Sequence.Object.DefectAreaExtended*, *var*

**VSet** *Sequence.Object.DefectAreaExtended*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not calculate extended defect area

1 – True Calculate extended defect area

Default: 0 – False

### Remarks

When DefectAreaExtended is True, the system extends the defect area using edge results that are within the defect level thresholds on either side of the found defect.

### See Also

LineInspector Object, ArcInspector Object

# DefectLevel Result

## Applies To

Vision Object: Arc Inspector, Line Inspector

## Description

Returns the level of a defect in pixels.

## Usage

**VGet** *Sequence.Object.DefectLevel*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult

## Values

Values are in pixels and can be negative or positive.

## Remarks

The DefectLevel is the maximum distance of an edge from the line or arc being inspected in pixels. DefectLevel will be greater than DefectLevelThreshPos or less than DefectLevelThreshNeg.

## See Also

ArcInspector Object, DefectLevelThreshNeg Property, DefectLevelThreshPos Property, LineInspector Object

## DefectLevelThreshNeg Property

### Applies To

Vision Objects: ArcInspector, LineInspector

### Description

Sets / returns the negative defect level threshold.

### Usage

**VGet** *Sequence.Object.DefectLevelThreshNeg*, *var*

**VSet** *Sequence.Object.DefectLevelThreshNeg*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Default: 2

Range: 0 - 100

### Remarks

DefectLevelThreshNeg is the threshold in pixels for a defect under the line or arc being inspected. During inspection, if the defect level of a defect candidate is greater than DefectLevelThreshNeg and less than DefectLevelThreshPos, then the candidate is not considered to be a defect.

### See Also

ArcInspector Object, Defect Level Result, DefectLevelThreshPos Property, LineInspector Object

# DefectLevelThreshPos Property

## Applies To

Vision Objects: ArcInspector, LineInspector

## Description

Sets / returns the positive defect level threshold.

## Usage

**VGet** *Sequence.Object.DefectLevelThreshPos, var*

**VSet** *Sequence.Object.DefectLevelThreshPos, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Default: 2

Range: 0 - 100

## Remarks

DefectLevelThreshPos is the threshold in pixels for a defect above the line or arc being inspected. During inspection, if the defect level of a defect candidate is greater than DefectLevelThreshNeg and less than DefectLevelThreshPos, then the candidate is not considered to be a defect.

## See Also

ArcInspector Object, DefectLevelThreshNeg Property, LineInspector Object

## DetailLevel Property

### Applies To

Geometric

### Description

Selects the level at which an edge is considered found during the geometric search.

### Usage

**VGet** *Sequence.Object.DetailLevel*, *var*

**VSet** *Sequence.Object.DetailLevel*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

### Values

1 – Medium (VISION\_DETAILLEVEL\_MEDIUM)

2 – High (VISION\_DETAILLEVEL\_HIGH)

3 - Very High (VISION\_DETAILLEVEL\_VERYHIGH)

Default: 1 – Medium

### Remarks

The DetailLevel property determines what is considered an edge during the search. Edges are defined by the transition in grayscale value between adjacent pixels. The default level (Medium) offers a robust detection of active edges from images with contrast variation, noise, and non-uniform illumination. Nevertheless, in cases where objects of interest have a very low contrast compared to high contrast areas in the image, some low contrast edges can be missed.

If your images contain low-contrast objects, a detail level setting of High should be used to ensure the detection of all important edges in the image. The Very High setting performs an exhaustive edge extraction, including very low contrast edges. However, it should be noted that this mode is very sensitive to noise.

The Smoothness property also affects how edges are extracted.

### See Also

Geometric Object, Smoothness Property

# DictionaryMode Property

## Applies To

Vision Object: OCR

## Description

Specifies the dictionary mode.

## Usage

**VGet** *Sequence.Object.DictionaryMode*, *var*

**VSet** *Sequence.Object.DictionaryMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

1 – All character type mode

2 – English restricted mode

3 – Non-use for system dictionary mode

## Remarks

OCR recognizes characters by using the system dictionary and user definition dictionary. Common fonts are registered in the system dictionary in advance. Also, the user definition dictionary includes characters used in projects and imported to projects.

The DictionaryMode property switches mode of the system dictionary. The all character type mode is recognized by common fonts used in Europe, the United State, and Japan. The English restricted mode is recognized by common fonts of alphabet (A-Z, a-z), number (0-9), and ASCII character code such as “!,”. The non-use for system dictionary mode uses the user definition dictionary only to recognize.

The user definition dictionary is used all registered characters regardless of the dictionary mode.

## See Also

OCR Object

## Directed Property

### Applies To

Vision Objects: Line, LineFinder  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Specifies whether to consider the placement direction of the object when calculating line output angles.

### Usage

**VGet** *Sequence.Object.Enabled*, *var*

**VSet** *Sequence.Object.Enabled*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 - False The placement direction of the object is taken into account.

1 - True The placement direction of the object is not taken into account.

Default: 1-True

### Remarks

This property is not enabled unless AngleOutType is set to "2: UseAngleBase". Refer to Remarks in the AngleOutType property.

### See Also

AngleBase Property, AngleMode Property

# Direction Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder

## Description

Specifies the direction of the Edge search.

## Usage

**VGet** *Sequence.Object.Direction*, *var*

**VSet** *Sequence.Object.Direction*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

1 - InsideOut

2 - OutsideIn

Default: 1 – InsideOut

## Remarks

The Direction property determines the edge detection direction for the ArcFinder, ArcInspector, BoxFinder, and CornerFinder objects.

When Direction is InsideOut, edges are searched in the direction indicated by the Direction Indicator.

When Direction is OutsideIn, the Direction Indicator will be opposite to InsideOut and edges are searched.

## See Also

ArcFinder Object, ArcInspector Object, BoxFinder Object, Contour Object, CornerFinder Object

## DistCorrectCal Property

### Applies To

Vision Calibration

### Description

Executes lens distortion correction and camera tilt correction used for vision calibration, and returns calibration status.

### Usage

**VGet** *Calibration*. **DistCorrectCal**, *var*

**VSet** *Calibration*. **DistCorrectCal**, *value*

*Calibration*     Name of a calibration or string variable containing a calibration name.

*var*                 Real variable that will contain the value of the result.

*value*              Integer expression that contains the new value of the property.

### Values

0 – False     Correction has not been completed.

1 – True       Calibration has been completed.

### Remarks

DistCorrectCal executes lens distortion correction and camera tilt correction used for vision calibration, and returns calibration status.

### See Also

DistCorrectCalComplete Result, DistCorrectEnable Property, DistCorrectTargetSeq Property

## DistCorrectCalComplete Result

### Applies To

Vision Calibration

### Description

Returns whether lens distortion correction and camera tilt correction used for vision calibration has been completed.

### Usage

**VGet** *Calibration*. **DistCorrectCalComplete**, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

0 – False Calibration has not been completed.

1 – True Calibration has been completed.

### Remarks

DistCorrectCalComplete returns whether lens distortion correction and camera tilt correction used for vision calibration has been completed.

### See Also

DistCorrectCal Property, DistCorrectEnable Property, DistCorrectTargetSeq Property

## DistCorrectEnable Property

### Applies To

Vision Calibration

### Description

Enables / disables lens distortion correction and camera tilt correction used for vision calibration and returns calibration status.

### Usage

**VGet** *Calibration*. **DistCorrectEnable**, *var*

**VSet** *Calibration*. **DistCorrectEnable**, *value*

*Calibration*     Name of a calibration or string variable containing a calibration name.

*var*                 Integer variable that will contain the value of the result.

*value*              Integer expression that contains the new value of the property.

### Values

0 – False     Lens distortion correction and camera position correction are disabled.

1 – True        Lens distortion correction and camera position correction are enabled.

### Remarks

DistCorrectEnable enables / disables lens distortion correction and camera tilt correction used for vision calibration and returns calibration status.

#### Note:

When using the results which are affected by vision calibration result (such as RobotXYU result and Length result), be sure to execute calibration again after changing the values of this property; otherwise, the results affected by calibration result do not return correct values. In such a case, do not use these results.

### See Also

DistCorrectCal Property, DistCorrectCalComplete Result, DistCorrectTargetSeq Property

# DistCorrectTargetSeq Property

## Applies To

Vision Calibration

## Description

Specifies a vision sequence to detect the calibration target in lens distortion correction and camera tilt correction used for the vision calibration.

## Usage

**VGet** *Calibration*. **DistCorrectTargetSeq**, *var*

**VSet** *Calibration*. **DistCorrectTargetSeq**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* String variable that will contain the value of the result.

*value* String expression that contains the new value of the property.

## Values

String containing a vision sequence name

Default: None

## Remarks

Specify DistCorrectTargetSeq for all calibrations where lens distortion correction and camera tilt correction are used. For details, refer to *Vision Guide Software 7. Vision Calibration*.

## See Also

DistCorrectCal Property, DistCorrectCalComplete Result, DistCorrectTargetSeq Property

## DistCorrectType Property

### Applies To

Vision Calibration

### Description

Selects or returns which distortion model is applied to lens distortion correction and camera tilt correction for vision calibration.

### Usage

**VGet** *Calibration*. **DistCorrectType**, *var*

**VSet** *Calibration*. **DistCorrectType**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* String variable that will contain the value of the result.

*value* String expression that contains the new value of the property.

### Values

1 – Lens distortion 1

2 – Lens distortion 2

3 – Camera tilt distortion

4 – Camera tilt distortion + Lens distortion 1

5 – Camera tilt distortion + Lens distortion 2

Default: 5

### Remarks

Normally, specify the camera tilt correction + lens distortion 2. If the distortion will not be modified, limit the distortion type to modify the distortion to modify it correctly.

Lens distortion 1 is a distortion model that distorts to radius direction.

Lens distortion 2 is a distortion model that distorts to circle direction.

Camera tilt distortion occurs when the sensor surface of the camera and calibration plate surface are not parallel.

### See Also

DistCorrectCal Property , DistCorrectCalComplete Result, DistCorrectTargetSeq Property

# EdgeCameraXYU Result

Runtime Only

## Applies To

Vision Objects: ArcFinder, LineFinder

## Description

Returns the CameraX, CameraY and Angle position coordinates of the edges found during the search.

## Usage

**VGet** *Sequence.Object.EdgeCameraXYU(result), found, xVar, yVar, uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the edge was found.

*xVar* Real variable that will contain the X coordinate position of the edge.

*yVar* Real variable that will contain the Y coordinate position of the edge.

*uVar* Real variable that will contain the angle of the edge search line in the camera coordinate system.

*result* Edge result number.

## Values

*found* Boolean value which is either True or False

*xVar* Real number in millimeters

*yVar* Real number in millimeters

*uVar* Real number in degrees

## Remarks

LineFinder and ArcFinder use several edges to find a line or arc. You can get the position results for each of the searched edges in the camera world by using EdgeCameraXYU.

You must set the CurrentResult property to the LineFinder result whose edge results you want to retrieve. The number of edge results is equal to the NumberOfEdges property.

The *xVar* and *yVar* values are returned in millimeters since the camera coordinate system is calibrated in millimeters. The *uVar* value is returned in degrees.

It should be noted that the EdgeCameraXYU result can only be calculated for vision sequences which have been associated with a calibration. If no calibration has been assigned to the vision sequence then if VGet is used to retrieve the EdgeCameraXYU result, an error will occur.

The EdgeCameraXYU result can be calculated only when valid calibration data is set to the Calibration property of the vision sequence. If the calibration has not been completed or the Calibration property has not been set, the EdgeCameraXYU result will be an error.

The EdgeCameraXYU result is available at runtime only.

## See Also

ArcFinder Object, EdgePixelXYU Result, EdgeRobotXYU Result, LineFinder Object

## EdgePixelXYU Result

Runtime only

### Applies To

Vision Objects: ArcFinder, LineFinder

### Description

Returns the PixelX, PixelY and Angle position coordinates of the edges found during the search.

### Usage

**VGet** *Sequence.Object.EdgePixelXYU (result) , found, xVar, yVar, uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the edge was found.

*xVar* Real variable representing the X pixel coordinate position of the edge.

*yVar* Real variable representing the Y pixel coordinate position of the edge.

*uVar* Real variable representing the angle of the edge search line in the image coordinate system

*result* Edge result number.

### Values

*found* Boolean value which is either True or False

*xVar* Real number in pixels

*yVar* Real number in pixels

*uVar* Real number in degrees

### Remarks

LineFinder and ArcFinder use several edges to find a line or arc. You can get the position results for each of the searched edges in the image coordinate system by using EdgePixelXYU.

You must set the CurrentResult property to the LineFinder (or ArcFinder) result whose edge results you want to retrieve. The number of edge results is equal to the NumberOfEdges property.

The EdgePixelXYU result is available at runtime only.

### See Also

ArcFinder Object, EdgeCameraXYU Result, EdgeRobotXYU Result, LineFinder Object

# EdgeRobotXYU Result

Runtime only

## Applies To

Vision Objects: ArcFinder, LineFinder

## Description

Returns the RobotX, RobotY and Angle position coordinates of the edges found during the search.

## Usage

**VGet** *Sequence.Object.EdgeRobotXYU(result), found, xVar, yVar, uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the edge was found.

*xVar* Real variable that will contain the X coordinate position of the edge.

*yVar* Real variable that will contain the Y coordinate position of the edge.

*uVar* Real variable that will contain the angle of the edge search line in the robot world coordinate system.

*result* Edge result number.

## Values

*found* True or False

*xVar* Real number in millimeters

*yVar* Real number in millimeters

*uVar* Real number in degrees

## Remarks

LineFinder and ArcFinder use several edges to find a line or arc. You can get the position results for each of the searched edges in the robot world coordinate system by using EdgeRobotXYU.

You must set the CurrentResult property to the LineFinder (or ArcFinder) result whose edge results you want to retrieve. The number of edge results is equal to the NumberOfEdges property.

The EdgeRobotXYU result returns a position in the robot coordinate system and therefore can be used for robot guidance applications. The EdgeRobotXYU result *xVar* and *yVar* values are always returned in millimeters. The *uVar* value is always returned in degrees.

It should be noted that the EdgeRobotXYU result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then the RobotXYU result cause an error to occur.

The EdgeRobotXYU result is available at runtime only.

## See Also

ArcFinder Object, EdgeCameraXYU Result, EdgePixelXYU Result, LineFinder Object

## EdgeThreshold Property

### Applies To

Vision Objects: ArcFinder, BoxFinder, Contour, CornerFinder, Edge, LineFinder

### Description

Sets the threshold percentage for which edges with grayscale variation below this value are ignored.

### Usage

**VGet** *Sequence.Object.EdgeThreshold*, *var*

**VSet** *Sequence.Object.EdgeThreshold*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number from 1 - 100%

Default: 2

### Remarks

Use EdgeThreshold to reject edges along the search path with smaller grayscale variations. During an edge search, the image along the search line will be converted to pixels projected onto one line. The EdgeThreshold allows edge values below the setting to be ignored.

### See Also

Edge Object, LineFinder Object, ArcFinder Object, BoxFinder Object, Contour Object, CornerFinder Object, Strength Result, StrengthTarget Property, StrengthVariation Property

# EdgeSort Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Contour, CornerFinder, Edge, LineFinder, LineInspector, BoxFinder

## Description

Sets the edge sorting order.

## Usage

**VGet** *Sequence.Object.EdgeSort*, *var*

**VSet** *Sequence.Object.EdgeSort*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 - Score	Sort in descending order by score
2 - PositionPos	Sort in descending order by proximity to the starting point on a search line
3 - PositionNeg	Sort in descending order by proximity to the end point on a search line
4 - Light	Sort in descending order by pixel value on the bright side of an edge
5 - Dark	Sort in ascending order by pixel value on the dark side of an edge
6 - Contrast	Sort in descending order by contrast
7 - Strength	Sort in descending order by edge strength

Default: 1 – Score

## Remarks

Select the sorting order to use when multiple edges are detected on a search line.

## See Also

Edge Object, LineFinder Object, ArcFinder Object, LineInspector Object, ArcInspector Object, BoxFinder Object, Contour Object, CornerFinder Object

## EdgeType Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

### Description

Sets / gets the type of edge to search for.

### Usage

**VGet** *Sequence.Object.EdgeType*, *var*

**VSet** *Sequence.Object.EdgeType*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

1 - Single

2 - Pair

Default: 1 – Single

### Remarks

Use **EdgeType** to choose whether to search for a single edge or an edge pair. An edge pair is two opposing edges. The coordinate of the pair is the midpoint of the line between the two edge coordinates.

### See Also

Edge Object, LineFinder Object, ArcFinder Object, LineInspector Object, ArcInspector Object, BoxFinder Object, Contour Object, CornerFinder Object

## EllipseAngle Property

### Applies To

Vision Object: ArcInspector

### Description

Specifies angle of the ellipse which is the inspection base line of ArcInspector.

### Usage

**VGet** *Sequence.Object. EllipseAngle, var*

**VSet** *Sequence.Object. EllipseAngle, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Specify angle of the ellipse within a range from -90 to 90 degrees.

### Remarks

EllipseAngle is used to specify angle of the ellipse when the ellipse is specified as the base line for ArcInspector.

### See Also

ArcInspector Object, EllipseMajorDiam Property, EllipseMinorDiam Property, Object Tab

## EllipseMajorDiam Property

### Applies To

Vision Object: ArcInspector

### Description

Specifies length of the major axis of the ellipse which is the inspection base line of ArcInspector.

### Usage

**VGet** *Sequence.Object*. **EllipseAngle**, *var*

**VSet** *Sequence.Object*. **EllipseAngle**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Specify length of the ellipse's major axis in pixels.

### Remarks

EllipseMajorDiam is used to specify length of the ellipse's major axis when the ellipse is specified as a base line used by ArcInspector.

### See Also

ArcInspector Object, EllipseMinorDiam Property, Object Tab

# Enabled Property

## Applies To

Vision Objects: All  
CV2 firmware Ver. 3.0.0.0 or later

## Description

Sets whether or not to execute the object.

## Usage

**VGet** *Sequence.Object.Enabled*, *var*

**VSet** *Sequence.Object.Enabled*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False The object will not be executed.

1 – True The object will be executed.

Default: 1-True

## Remarks

When the Enabled property is set to [0-False], the object will not be executed.

## See Also

All Vision objects

## EllipseMinorDiam Property

### Applies To

Vision Object: ArcInspector

### Description

Specifies length of the minor axis of the ellipse which is the inspection base line of ArcInspector.

### Usage

**VGet** *Sequence.Object. EllipseAngle, var*

**VSet** *Sequence.Object. EllipseAngle, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Specify length of the ellipse's minor axis in pixels.

### Remarks

EllipseMinorDiam is used to specify length of the ellipse's minor axis when the ellipse is specified as a base line used by ArcInspector.

### See Also

ArcInspector Object, EllipseMajorDiam Property, Object Tab

## EndPntObjResult Property

### Applies To

Vision Objects: Contour, Edge, Line, LineInspector

### Description

Specifies which result to use from the EndPointObject.

### Usage

**VGet** *Sequence.Object.EndPntObjResult*, *var*

**VSet** *Sequence.Object.EndPntObjResult*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

The value can be All, or range from 1 to the NumberOfResults value for the EndPointObject. If the EndPointObject is 'Screen', then the value is always 1.

### Remarks

EndPntObjResult enables you to attach several objects to the results of one EndPointObject. For example, you could create a Blob object with NumberToFind set to 4. Then you could attach a line object to each one of the results by specifying the blob for the EndPointObject of each line and a different EndPntObjResult for each line. In addition, you can specify All. If both the StartPntObjResult and the EndPntObjResult properties are set to All, then the object will be executed for each result.

### See Also

Edge Object, EndPointObject Property, Contour Object, Line Object, LineInspector Object, Object Tab, StartPntObjResult Property

## EndPointObject Property

### Applies To

Vision Objects: Contour, Edge, Line, LineInspector

### Description

Specifies the vision object to use for the end point of a Line object.

### Usage

**VGet** *Sequence.Object.EndPointObject*, *var*

**VSet** *Sequence.Object.EndPointObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property. Valid vision objects for the EndPointObject property are: Blob, Correlation, Edge, Geometric, Line, and Point objects. The Screen may also be used as the EndPointObject.

### Values

Screen or any object that runs prior to the Line object.

Default: Screen

### Remarks

When a Line object is first created, the EndPointObject property is set to Screen. However, Line objects are normally attached to other vision objects. This is the purpose of the StartPointObject and EndPointObject properties. Through these two properties the user can define a line between any two vision objects (except Frames).

Frame objects cannot be used to define an end point for a Line object. However, this does not cause a limitation because Frames are defined by other vision objects. In those cases where you want to define a line end point with a Frame object, use a Point object in the frame to define the end point of the Line object.

It is important to note that for each specific vision sequence, only those vision objects which are executed in a step prior to the Line object in the vision sequence steps will be available to use as an EndPointObject.

### See Also

Edge Object, EndPointType Property, Contour Object, Line Object, LineInspector Object, Object Tab, StartPointObject Property

# EndPointerType Property

## Applies To

Vision Objects: Contour, Edge, Line, LineInspector

## Description

Specifies the type of end point to use for the line object. In most cases the end point type will be a point (which usually means the PixelX and PixelY position of the EndPointObject). However, when the EndPointObject for the current line is a 2nd Line object, the EndPointerType property is used to define an intersection point on the 2nd line such as the lines midpoint, endpoint, startpoint or perpendicular position.

## Usage

**VGet** *Sequence.Object.EndPointerType, var*

**VSet** *Sequence.Object.EndPointerType, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

EndPointObject = Line, LineFinder	EndPointObject = Screen, Blob, Correlation, Geometric, Edge, or Point object
See remarks. Default: 2 – MidPoint	0 – Point When used with objects other than the Line object, the EndPointerType can only be of type Point. Default: 0 – Point

### Remarks

As shown in the Values table above, many of the EndPointObject properties support only “0 – Point” of the EndPointType property. This is because most EndPointObject properties normally use the PixelX and PixelY position for a reference position for defining a Start or End Point for a line. So when the EndPointObject is set as Screen, Blob, Correlation, Edge, or Point, the EndPointType will always be set to 0 – Point.

The range of valid values for EndPointType depend upon the EndPointObject.

When the EndPointObject is another Line object, the user must decide where on the 2nd line to intersect with the 1st line. The choices are as follows:

- |                    |   |
|--------------------|---|
| 1 - EndPoint       | Use the end point of the other line as the endpoint for this line.  |
| 2 - MidPoint       | Cut the other line in half and use the center (or midpoint of the other line as the endpoint for this line.   |
| 3 - Perpendicular  | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion and use this position as the end point.   |
| 4 - StartPoint     | Use the starting point of the other line as the end point for this line.  |
| 5 - PerpToStartPnt | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the start point of the first line and use this position as the end point. |
| 6 - PerpToMidPnt   | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the midpoint of the first line and use this position as the end point.    |
| 7 - PerpToEndPnt   | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the end point of the first line and use this position as the end point.   |

If the EndPointObject is modified to a Line object then the EndPointType is automatically changed to MidPoint.

If the EndPointObject is modified to Screen or Blob, Correlation, Edge, or Point object then the EndPointType is automatically changed to 0 - Point.

### See Also

Contour Object, Edge Object, EndPointObject Property, Line Object, LineInspector Object, Object Tab, StartPointType Property

# ExportFont Property

Design time only

## Applies To

Vision Object: OCR

## Description

Runs a file dialog from the Vision Guide GUI that allows you to export a font file.

## Remarks

Use the ExportFont property to export a font file.

## See Also

ImportFont Property, Object Tab, OCR Object

## ExposureDelay Property

### Applies To

Vision Sequence

### Description

Sets the delay time between receiving the hardware trigger and starting the exposure.

### Usage

**VGet** *Sequence*.**ExposureDelay**, *var*

**VSet** *Sequence*.**ExposureDelay**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

### Values

Long value in microseconds.

Default: 0 (microsecond)

### Remarks

Use `ExposureDelay` to set the time delay between the hardware trigger and the exposure start.

### See Also

`RuntimeAcquire Property`, `ExposureTime Property`, `StrobeDelay Property`, `StrobeTime Property`

# ExposureTime Property

## Applies To

Vision Sequences

## Description

Sets the electronic shutter time for a camera.

## Usage

**VGet** *Sequence.ExposureTime, var*

**VSet** *Sequence.ExposureTime, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

## Values

Long value in microseconds.

Default: 0 (microsecond)

Camera	Default	Application			
		CV1	CV2-S/H	CV2-SA/HA	PC vision
NET 1044 BU	26000 microseconds	✓	✓	✓	-
NET 4133 BU / CU	26000 microseconds	✓	✓	✓	-
NET 1500 BU / CU	26000 microseconds	✓	✓	✓	-
acA640-100gm	8000 microseconds	-	✓	✓	✓
acA640-120gm	8000 microseconds	-	✓	✓	✓
acA640-300gm	5000 microseconds	-	-	✓	✓
acA1600-20gm / gc	10000 microseconds	-	✓	✓	✓
acA1600-60gm / gc	10000 microseconds	-	✓	✓	✓
acA2500-14gm / gc	35000 microseconds	-	✓	✓	✓
acA2500-20gm / gc	5000 microseconds	-	-	✓	✓
acA3800-10gm	35000 microseconds	-	-	✓	✓

## Remarks

When RuntimeAcquire is 1–Stationary and ExposureTime = 0, the default exposure time is used, as shown below.

When RuntimeAcquire is 1–Stationary the camera acquires images in the specified shutter time. When the sequence runs, the camera sensor is exposed for the ExposureTime, and then the image is acquired.

## See Also

RuntimeAcquire Property, CameraBrightness Property, CameraContrast Property, Sequence Tab

## Extrema Result

Runtime only

### Applies To

Vision Objects: Blob, DefectFinder

### Description

Returns the blob extrema coordinates.

### Usage

**VGet** *Sequence.Object.Extrema* [(*result*)], *varMinX*, *varMaxX*, *varMinY*, *varMaxY*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*varMinX* Integer variable containing minimum X position of the blob's Extrema.

*varMaxX* Integer variable containing maximum X position of the blob's Extrema.

*varMinY* Integer variable containing minimum Y position of the blob's Extrema.

*varMaxY* Integer variable containing maximum Y position of the blob's Extrema.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

The values returned are always in pixels and may be from 1 – ImageSize property height.

### Remarks

The MinX, MaxX, MinY, and MaxY results together define a blob's smallest enclosing rectangle that is aligned with the coordinate axes and completely encloses the blob. This rectangle is known as the extrema. The Extrema result enables you to retrieve all four coordinates in one command.

### See Also

Area Result, Blob Object, DefectFinder Object, MinX Result, MaxX Result, MaxY Result, MinY Result, Object Tab

# FailColor Property

## Applies To

Vision Objects: All vision objects

## Description

Sets the object display color when the object is not passed.

## Usage

**VGet** *Sequence.Object.FailColor, var*

**VSet** *Sequence.Object.FailColor, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

Sets the object display color when the object is not passed using a string containing the name of the color.

Default: "Red"

## Remarks

The FailColor property allows you to set the color used for failed objects.

## See Also

Found Result, Graphics Property, PassColor Property, Passed Result

## FillHoles Property

### Applies To

Vision Objects: Blob, Contour, ImageOp  
CV2 firmware Ver. 3.0.0.0 or later

### Description

Sets whether or not to fill the holes when using the binary image.

### Usage

**VGet** *Sequence.Object.FillHoles, var*

**VSet** *Sequence.Object.FillHoles, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not fill the holes.

1 – True Fill the holes.

Default: 0 – False

### Remarks

The FillHoles property is available for the ImageOp objects only in binary operation. The property is always available for the Blob objects. When the FillHoles property is set to “1-True”, the holes in the binary image are filled. The holes are areas that are completely enclosed by the objects.

### See Also

ImageOp Object, Blob Object, Contour Object

# FindChar Property

## Applies To

Vision Object: OCR  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Sets whether to treat each individual character detected in a string as an individual object.

## Usage

**VGet** *Sequence.Object.FindChar*, *var*

**VSet** *Sequence.Object.FindChar*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 - False Each character is not treated as an individual object.

1 - True Each character is treated as an individual object.

Default: 0 – False

## Remarks

The FindChar property can set how strings detected with OCR are handled.

When set to 0 - False, detected strings are treated as a single object.

When set to 1 - True, each character in a detected string is treated as a single object. Set this to 1 - True to reference individual character coordinates, and text and other results individually.

## See Also

OCR Object

## FitError Result

### Applies To

Vision Objects: ArcFinder, BoxFinder, CornerFinder, LineFinder

### Description

Returns the line or arc fitting error value.

### Usage

**VGet** *Sequence*.FitError[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*var* Real type variable containing a value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number indicating the line or arc fitting error.

### Remarks

FitError returns root mean square of the distances between each edge point and the found line or arc except the outliers.

### See Also

ArcFinder Object, LineFinder Object, BoxFinder Object, CornerFinder Object, MaxError Result

# FittingThreshold Property

## Applies To

Vision Objects: ArcFinder, BoxFinder, Contour, CornerFinder, LineFinder

## Description

Defines the line or arc fitting threshold.

## Usage

**VGet** *Sequence.Object.FittingThreshold*, *var*

**VSet** *Sequence.Object.FittingThreshold*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real number in pixels.

Default: 10

## Remarks

The FittingThreshold specifies which edge results to use for fitting the line or arc. During the iterative edge search to find the line or arc, if an edge is found whose distance from the line is greater than the FittingThreshold value, then the edge result is not used in the final line or arc fitting.

## See Also

LineFinder Object, ArcFinder Object, BoxFinder Object, Contour Object, CornerFinder Object, FitError Result

## FocusValue Result

### Applies To

Vision Object: ImageOp  
CV2 firmware Ver. 3.0.0.0 or later

### Description

Displays a relative focus level.

### Usage

**VGet** *Sequence.Object.FocusValue, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*value* Real variable that will contain the value of the result.

### Values

Real number indicating a relative focus level.

### Remarks

The FocusValue result is displayed when “Operation” of the ImageOp object is set to DetectFocus. The FocusValue result shows a relative focus level. The optimum focus can be obtained when the value becomes the minimum.

### See Also

ImageOp Object

# Font Property

## Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

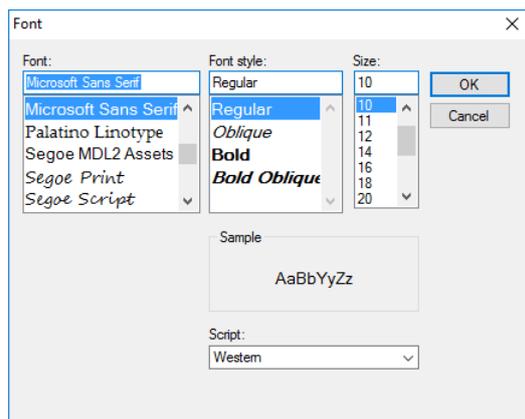
## Description

Sets and confirms the type face of characters rendered as a Text object.

## Remarks

The Font property can be accessed from the Vision Guide window Objects tab. Click the Font property Value Field to display the button. Click this button to display the Fonts window.

You can make changes to the font type face in the Fonts window. Changeable items include the font name, style and size of the font. All fonts installed on your PC are available for selection in “Fonts”.



## See Also

Text Object

## FontBold Property

### Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Displays the character string rendered in bold.

### Usage

**VGet** *Sequence.Object.FontBold, var*

**VSet** *Sequence.Object. FontBold, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

False Characters are not displayed in bold.

True Characters are displayed in bold.

Default: False

### Remarks

You can switch the format of character strings rendered with the TextView object. Set the FontBold property to True to render character strings in bold.

### See Also

Text Object

# FontItalic Property

## Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Displays the character string rendered in italics.

## Usage

**VGet** *Sequence.Object.FontItalic, var*

**VSet** *Sequence.Object.FontItalic, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

False Characters are not displayed in italics.

True Character are displayed in italics.

Default: False

## Remarks

You can switch the format of character strings rendered with the TextView object. Set the Fontitalic property to True to render character strings in italics.

## See Also

Text Object

## FontName Property

### Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Sets and retrieves the name of the font.

### Usage

**VGet** *Sequence.Object.FontName, var*

**VSet** *Sequence.Object.FontName, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

The character string for the name of the font.

Default: Microsoft Sans Serif

### Remarks

Sets the name of the font for the character string rendered with the Text object.

### See Also

Text Object

# FontSize Property

## Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Defines the size of the font rendered.

## Usage

**VGet** *Sequence.Object.FontSize*, *var*

**VSet** *Sequence.Object.FontSize*, *value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Default: 10

## Remarks

Defines the size of the character string rendered with Text.

## See Also

Text Object

## Found Result

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, Defect Finder, Edge, Frame, Geometric, Line, LineFinder, LineInspector, OCR, Point, Polar, Text

### Description

Returns whether or not the object was found.

### Usage

**VGet** *Sequence.Object.Found* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

### Values

0-False The part was not found

1-True The part was found

### Remarks

The Found result simply returns whether or not the part or the current object is looking for was found. For example, and Edge Object returns whether or not an Edge was found and a Correlation object returns whether or not an image was found which matches the taught model.

The Found result also can be acquired by the RobotXYU, PixelXYU, and CameraXYU results.

You can also configure how an object is considered to be Passed by using PassType. For example, set PassType = AllNotFound for DefectFinder object to consider the result as failed if there was a found object (defect).

### See Also

Blob Object, CameraXYU Result, CodeReader Object, Correlation Object, CurrentResult Property, DefectFinder Object, Edge Object, FoundOnEdge Result, Frame Object, Geometric Object, Line Object, LineFinder Object, ArcFinder Object, NumberFound Result, NumberToFind Property, Object Tab, OCR Object, Passed Result, Point Object, Polar Object, RobotXYU Result, Score Result, PassType Property, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

# FoundCodeType Result

## Applies To

Vision Object: CodeReader

## Description

Returns the type of the detected bar code.

## Usage

**VGet** *Sequence.Object.FoundCodeType*[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable containing the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Returns the type of the detected bar code. For details on values to be acquired, refer to *Remarks* below.

## Remarks

Returns the bar code type found in CodeReader object. Return values are constants as shown below.

Code	Constant	Remarks
Codabar	VISION_CODETYPE_CODABAR	
Code39	VISION_CODETYPE_CODE39	
Code128	VISION_CODETYPE_CODE128	
DataMatrix	VISION_CODETYPE_DATAMATRIX	2D code
EAN 8	VISION_CODETYPE_EAN8	
EAN 13	VISION_CODETYPE_EAN13	
Interleaved 2 of 5	VISION_CODETYPE_INTERLEAVED25	Also called ITF
PDF417	VISION_CODETYPE_PDF417	2D code
QR	VISION_CODETYPE_QR	2D code
UPC	VISION_CODETYPE_UPC	
UPC A	VISION_CODETYPE_UPCA	
UPC E	VISION_CODETYPE_UPCE	

## FoundCodeType Result

---

The following example displays the name of the found bar code on the screen.

```
Function main
  Integer code
  VRun CodeTest
  VGet CodeTest.Code01.FoundCodeType, code
  Select code
    Case VISION_CODETYPE_CODABAR
      Print "VISION_CODETYPE_CODABAR"
    Case VISION_CODETYPE_CODE39
      Print "VISION_CODETYPE_CODE39"
    Case VISION_CODETYPE_CODE128
      Print "VISION_CODETYPE_CODE128"
    Case VISION_CODETYPE_DATAMATRIX
      Print "VISION_CODETYPE_DATAMATRIX"
    Case VISION_CODETYPE_EAN8
      Print "VISION_CODETYPE_EAN8"
    Case VISION_CODETYPE_EAN13
      Print "VISION_CODETYPE_EAN13"
    Case VISION_CODETYPE_INTERLEAVED25
      Print "VISION_CODETYPE_INTERLEAVED25"
    Case VISION_CODETYPE_PDF417
      Print "VISION_CODETYPE_PDF417"
    Case VISION_CODETYPE_QR
      Print "VISION_CODETYPE_QR"
    Case VISION_CODETYPE_UPC
      Print "VISION_CODETYPE_UPC"
    Case VISION_CODETYPE_UPCA
      Print "VISION_CODETYPE_UPCA"
    Case VISION_CODETYPE_UPCE
      Print "VISION_CODETYPE_UPCE"
  Send
Fend
```

### See Also

CodeReader Object, CodeType Property, Object Tab

## FoundMajorDiam Result

### Applies To

Vision Object: ArcFinder

### Description

Returns length of the major axis of the ellipse detected by ArcFinder.

### Usage

**VGet** *Sequence.Object*. **FoundMajorDiam**, *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable containing the value of the result.

### Values

Returns length of the major axis of the detected ellipse in millimeters.

### Remarks

Returns length of the major axis of the ellipse detected by ArcFinder in millimeters. To obtain the value in pixels, use the PixelMajorDiam result.

### See Also

ArcFinder Object, ArcSearchType Property, FoundMinorDiam Result, PixelMajorDiam Result, PixelMinorDiam Result, Object Tab

## FoundMinorDiam Result

### Applies To

Vision Object: ArcFinder

### Description

Returns length of the minor diameter of the ellipse detected by ArcFinder.

### Usage

**VGet** *Sequence.Object*. **FoundMajorDiam**, *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable containing the value of the result.

### Values

Returns length of the minor diameter of the detected ellipse in millimeters.

### Remarks

Returns length of the minor diameter of the ellipse detected by ArcFinder in millimeters. To obtain the value in pixels, use the PixelMinorDiam result.

### See Also

ArcFinder Object, ArcSearchType Property, FoundMajorDiam Result, PixelMajorDiam Result, PixelMinorDiam Result, Object Tab

# FoundOnEdge Result

## Applies To

Vision Objects: Blob, Correlation, DefectFinder, Geometric

## Description

Returns 1–True when an object is found too close to the edge of the search window.

## Usage

**VGet** *Sequence.Object.FoundOnEdge* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult.Used for objects that return multiple results.

## Values

0–False The object was at the at the edge of the search window and could not be found

1–True The object was found at the edge of the search window

## Remarks

The FoundOnEdge result is a special result which works only with the Blob, Correlation, Geometric, and DefectFinder objects.

Sometimes the Vision System tries to report that a Blob, Correlation, Geometric, or DefectFinder object was found even though part of the object may be located outside of the Field of View. Rather than report these objects as Found, Vision Guide can set the Found result to return “0–False” when a Blob, Correlation, Geometric, or DefectFinder object is found but part of the object is outside of the Search Window.

If you want to make the object whose FoundOnEdge result is 1–True as “not found”, set the RejectOnEdge property to 1–True.

### Note:

There will be cases where you will use a Correlation or Geometric object and the Found result returns “0–False” even though the Accept property is low. In these situations, check for the FoundOnEdge result and RejectOnEdge property. They may be excluded from the detection result by RejectOnEdge property setting.

## See Also

Blob Object, Correlation Object, Found Result, Geometric Object, DefectFinder Object, Object Tab, RejectOnEdge Property, Score Result

## FoundRadius Result

### Applies To

Vision Object: ArcFinder

### Description

Returns the radius of the found circular object in millimeters.

### Usage

**VGet** *Sequence.Object.FoundRadius*[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real type variable containing a value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult.Used for objects that return multiple results.

### Values

Real number indicating the radius [Unit: mm]

### Remarks

Returns the radius of the found circular object in millimeters.

### See Also

ArcFinder Object, Object Tab

# FOVHeight Result

## Applies To

Vision Calibration

## Description

Returns the FOV (field of view) height of a calibration.

## Usage

**VGet** *Calibration.FOVHeight*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

## Values

Real number in millimeters.

## Remarks

FOVHeight is the height of the camera field of view in millimeters. The calibration must be completed before FOVHeight can be retrieved.

## See Also

FOVWidth Result, XmmPerPixel Result, YmmPerPixel Result

## FOVWidth Result

### Applies To

Vision Calibration

### Description

Returns the FOV (field of view) width of a calibration.

### Usage

**VGet** *Calibration.FOVWidth*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

FOVWidth is the width of the camera field of view in millimeters. The calibration must be completed before FOVWidth can be retrieved.

### See Also

FOVHeight Result, XmmPerPixel Result, YmmPerPixel Result

# Frame Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, ImageOp, Line, LineFinder, LineInspector, OCR, Point, Polar

## Description

Sets the frame for positioning an object's search position.

## Usage

**VGet** *Sequence.Object.Frame*, *var*

**VSet** *Sequence.Object.Frame*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

Any Frame object which is executed in the sequence prior to the current vision object can be specified as the Frame property value.

Default: None

## Remarks

The Frame property is very useful for aligning objects to specific positions with respect to other objects' found positions. For details, refer to the Frame object description in *Vision Objects*.

The Frame property can be used for any Frame objects which have been defined beforehand when the associated Frame object is located prior to the vision object in the Object Execution Step.

The Object Execution Step List is the list shown on the Sequence Tab which shows the execution order for all objects in the current sequence.

## See Also

Blob Object, CodeReader Object, ColorMatch Object, Correlation Object, Edge Object, Frame Object, Geometric Object, ImageOp Object, Line Object, Object Tab, OCR Object, Point Object, Polar Object, BoxFinder Object, Contour Object, CornerFinder Object

## FrameResult Property

### Applies To

Vision Object: Blob, BoxFinder, Contour, CornerFinder, Correlation, Geometric, Edge, Polar, Defect Finder, Arc Finder, Line Finder, Point, Line

### Description

Sets and returns the result number of the frame to be used.

### Usage

**VGet** *Sequence.Object.FrameResult* , *var*

**VSet** *Sequence.Object.FrameResult* , *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

FrameResult: All (use all results)  
1 - n (use the specified result)

Default: 1

### Remarks

Sets the result of the Frame to be used. If the FrameResult property is set to All, the object will be used for all frame results.

### See Also

Frame Object, Object Tab

# Graphics Property

## Applies To

Vision Objects: All

## Description

Specifies which graphics to display at runtime and design time. (i.e. Whether to show graphics for each object, just position information, or nothing at all.) For example, you can restrain the result display by setting the Graphics property of the vision objects which do not need to be displayed to “None”.

## Usage

**VGet** *Sequence.Object.Graphics*, *var*

**VSet** *Sequence.Object.Graphics*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

- |                   |  |
|-------------------|--|
| 1 - All           | Vision constant: VISION_GRAPHICS_ALL<br>Display object labels, line references, and found position |
| 2 - Position Only | Vision constant: VISION_GRAPHICS_POSONLY<br>Display found position only                            |
| 3 - None          | Vision constant: VISION_GRAPHICS_NONE<br>Don't display any graphics at run time                    |

Default: 1 – All

## Remarks

While graphics such as vision object Labels (which show vision object names), lines, Search Windows, and found position crosshairs are very useful they can get in the way if too many are displayed at the same time. The Graphics property can eliminate unnecessary clutter on the Vision Guide Development, Run or Operator Windows by removing those graphics from objects which the designer specifies.

The Graphics property is used to define the graphics display characteristics for each vision object. These will normally be set to values which, when combined with the Graphics Properties of other vision objects, will help reduce screen display clutter. The Graphics property is normally used to set the graphics characteristics exactly as you would like your final vision solution to display graphics on the Run or Operator Window.

The Graphics property settings for all vision objects can be overridden with the Force All Graphics On and Force Labels Off Vision Guide toolbar buttons.

It is important to note that the Graphics property settings apply to both runtime and design modes. (i.e. the Run Window, Operator Window, and Vision Guide Window) This is done to ensure that the graphics display is always the same regardless of if you run a sequence from the Vision Guide Window or from a program.

See Also

Blob Object, CodeReader Object, Correlation Object, Edge Object, Frame Object, Geometric Object, ImageOp Object, Line Object, OCR Object, Point Object, Polar Object, Object Tab

# GridColor Property

## Applies To

Vision Sequence

## Description

Specifies color of grids displayed in the vision sequence.

## Usage

**VGet** *Sequence*. **GridColor**, *var*

**VSet** *Sequence*. **GridColor**, *value*

*Sequence*      String variable containing a sequence name.

*var*            Integer variable that will contain the value of the property.

*value*         Integer expression for the new value of the property.

## Values

Specifies a name of color for grids displayed in sequence execution by a string containing a name of color.

Default: "Dark Gray"

## Remarks

Specifies color of grids displayed in a vision sequence.

## See Also

Vision Sequence, GridPitchX Property, GridPitchY Property, GridShow Property, GridType Property, GridUnits Property, Sequence Tab

## GridPitchX Property

### Applies To

Vision Sequence

### Description

Specifies the X pitch of grids displayed in the vision sequence.

### Usage

**VGet** *Sequence*. **GridPitchX**, *var*

**VSet** *Sequence*. **GridPitchX**, *value*

*Sequence*      String variable containing a sequence name.

*var*             Integer variable that will contain the value of the property.

*value*           Integer expression for the new value of the property.

### Values

Specifies the X pitch of the grids displayed in the vision sequence. The unit depends on the GridUnits property.

Default: 100

### Remarks

Specifies the X pitch of grids displayed in the vision sequence. The unit depends on the GridUnits property.

#### Note:

If millimeter is specified in the GridUnits property, the grids are displayed only when calibration specified by the sequence is completed.

### See Also

Vision Sequence, GridColor Property, GridPitchY Property, GridShow Property, GridType Property, GridUnits Property, Sequence Tab

# GridPitchY Property

## Applies To

Vision Sequence

## Description

Specifies the Y pitch of grids displayed in the vision sequence.

## Usage

**VGet** *Sequence*. **GridPitchY**, *var*

**VSet** *Sequence*. **GridPitchY**, *value*

*Sequence*      String variable containing a sequence name.

*Calibration*      String variable containing a calibration name.

*var*              Integer variable that will contain the value of the property.

*value*            Integer expression for the new value of the property.

## Values

Specifies the Y pitch of grids displayed in the vision sequence. The unit depends on the GridUnits property.

Default: 100

## Remarks

Specifies the Y pitch of grids displayed in the vision sequence. The unit depends on the GridUnits property.

### Note:

If millimeter is specified in the GridUnits property, the grids are displayed only when calibration specified by the sequence is completed.

## See Also

Vision Sequence, GridColor Property, GridPitchX Property, GridShow Property, GridType Property, GridUnits Property, Sequence Tab

## GridShow Property

### Applies To

Vision Sequence

### Description

Specifies whether to display the grids for the vision sequence.

### Usage

**VGet** *Sequence*. **GridShow**, *var*

**VSet** *Sequence*. **GridShow**, *value*

*Sequence*      String variable containing a sequence name.

*var*              Integer variable that will contain the value of the property.

*value*            Integer expression for the new value of the property.

### Values

0 – False      Grids are not displayed.

1 – True        Grids are displayed.

Default: 0 - False

### Remarks

Specifies whether or not to display the grids when executing the sequence.

#### Note:

If millimeter is specified in the GridUnits property, the grids are displayed only when calibration specified by the sequence is completed.

### See Also

Vision Sequence, GridColor Property, GridPitchX Property, GridPitchY Property, GridType Property, GridUnits Property, Sequence Tab

# GridType Property

## Applies To

Vision Sequence

## Description

Specifies a type of grids displayed in the vision sequence.

## Usage

**VGet** *Sequence*. **GridType**, *var*

**VSet** *Sequence*. **GridType**, *value*

*Sequence*      String variable containing a sequence name.

*var*            Integer variable that will contain the value of the property.

*value*          Integer expression for the new value of the property.

## Values

Specifies a type of grids displayed in the sequence execution.

1 – CrossHair    Displays a crosshair at the center of the camera image  
(VISION\_GRIDTYPE\_CROSSHAIR)

2 – Rectangle    Displays grids with specified XY pitches. (VISION\_GRIDTYPE\_RECTANGLE)

Default: 1 - CrossHair

## Remarks

Specifies a type of the grids displayed in the vision sequence.

### Note:

If millimeter is specified in the GridUnits property, the grids are displayed only when calibration specified by the sequence is completed.

## See Also

Vision Sequence, GridColor Property, GridPitchX Property, GridPitchY Property, GridShow Property, GridUnits Property, Sequence Tab

## GridUnits Property

### Applies To

Vision Sequence

### Description

When the grid type for the vision sequence is Rectangle, specifies the unit of GridPitchX and GridPitchY property values.

### Usage

**VGet** *Sequence*. **GridUnits**, *var*

**VSet** *Sequence*. **GridUnits**, *value*

*Sequence*      String variable containing a sequence name.

*var*            Integer variable that will contain the value of the property.

*value*         Integer expression for the new value of the property.

### Values

1 – Pixel        Pixels (VISION\_GRIDUNITS\_PIXEL)

2 – MM         Millimeters (VISION\_GRIDUNITS\_MM)

Default: 1 - Pixel

### Remarks

When the grid type for the vision sequence is Rectangle, specifies the unit of GridPitchX and GridPitchY property values.

### See Also

Vision Sequence, GridColor Property, GridPitchX Property, GridPitchY Property, GridShow Property, GridType Property, Sequence Tab

# HDRMode Property

## Applies To

Vision Sequence  
CV2 firmware Ver. 3.0.0.0 or later

## Description

Displays the captured image as the HDR (High Dynamic Range) image.

## Usage

**VGet** *Sequence.HDRMode*, *var*

**VSet** *Sequence.HDRMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the result.

*value* Boolean expression for the new value of the property.

## Values

0 – False Displays the normal image.

1 – True Displays the HDR image.

Default: 0 – False

## Remarks

Displays the captured image as the HDR (High Dynamic Range) image.  
Brightness of the HDR image can be adjusted in the ExposureTime property.

## See Also

ExposureTime Property

## Holes Result

### Applies To

Vision Object: Blob

### Description

Returns the number of holes found within a Blob object.

### Usage

**VGet** *Sequence.Object.Holes* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the CurrentResult

### Values

Valid values are zero to the number of holes found.

### Remarks

A hole is a blob with opposite polarity located within the blob that was found. Holes that intersect the edge of the blob are not counted.

### See Also

Blob Object, Compactness Result, Object Tab, Perimeter Result, Roughness Result

# ImageBuffer Property

## Applies To

Vision Sequence

## Description

Specifies which image buffer to use for a sequence.

## Usage

**VGet** *Sequence*.**ImageBuffer**, *var*

**VSet** *Sequence*.**ImageBuffer**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0-10.

Default: 0

## Remarks

When a sequence takes a picture or searches for objects, it uses the image buffer specified with the ImageBuffer property. When ImageBuffer is 0, the camera image buffer is used. ImageBuffer 1 – 10 are global image buffers that are shared between all sequences in the project. Using shared image buffers, you can acquire an image with the one sequence and search the image acquired by that sequence with other sequences. When you are only using the image that has already been acquired by another sequence, you must set the sequence RuntimeAcquire property to None, and set the ImageBuffer property as needed.

## Example

In the following example, the robot is moved to five camera positions and a picture is taken at each position into an image buffer. Then, the acquired image is searched by another sequence .Function FindParts

```
Integer i

' Move the camera to 5 positions and grab an image into 5 buffers
For i = 1 to 5
  Go P(100 + i)
  VSet TakePicture.ImageBuffer, i
  VRun TakePicture
Next i

' Signal to other tasks that we are done with the robot
MemOn ScanFinished

' Search for a part in each image
' The SearchPart sequence RuntimeAcquire property is set to None
For i = 1 to 5
  VSet SearchPart.ImageBuffer, i
  VRun SearchPart
  VGet SearchPart.Blob01.Found, g_PartFound(i)
Next i
Fend
```

## ImageBuffer1 Property

### Applies To

Vision Object: ImageOp

### Description

Sets and returns the first source image buffer to be used for the SubtractAbs operation.

### Usage

**VGet** *Sequence.Object.ImageBuffer1* , *var*

**VSet** *Sequence.Object.ImageBuffer1*, *value*

*Sequence*   String variable containing a sequence name.

*Object*       Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var*           Real variable that will contain the value of the property.

*value*        Real expression for the new value of the property.

### Values

ImageBuffer1: 0 to 10

Default: 0

### Remarks

Specifies the image buffer number where the first source image data of the SubtractAbs operation is stored.

### See Also

ImageOpe Object, Object Tab

## ImageBuffer2 Property

### Applies To

Vision Object: `ImgOpe`

### Description

Sets and returns the second source image buffer to be used for the `SubtractAbs` operation.

### Usage

**VGet** *Sequence.Object.ImageBuffer2, var*

**VSet** *Sequence.Object.ImageBuffer2, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

`ImageBuffer2`: 0 to 10

Default: 0

### Remarks

Specifies the image buffer number where the second image data of the `SubtractAbs` operation is stored.

### Example

`ImageOpe Object, Object Tab`

# ImageColor Property

## Applies To

Vision Sequence

## Description

Specifies how the color image is acquired.

## Usage

**VGet** *Sequence*.**ImageColor**, *var*

**VSet** *Sequence*.**ImageColor**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

- 1 – All Vision constant: VISION\_IMAGECOLOR\_ALL  
Acquire all three color bands: Red, Green, and Blue.
- 2 – Red Vision constant: VISION\_IMAGECOLOR\_RED  
Acquire only the red color band.
- 3 – Green Vision constant: VISION\_IMAGECOLOR\_GREEN  
Acquire only the green color band.
- 4 – Blue Vision constant: VISION\_IMAGECOLOR\_BLUE  
Acquire only the blue color band.
- 5 – Grayscale Vision constant: VISION\_IMAGECOLOR\_GRAYSCALE  
Acquire a grayscale image.

Default: 1 - All

## Remarks

Use the ImageColor property to configure which color band(s) to acquire. This property is only available for color cameras.

# ImageFile Property

## Applies To

Vision Sequence

## Description

Sets or returns the image file for the current sequence.

## Usage

**VGet** *Sequence.ImageFile, var*

**VSet** *Sequence.ImageFile, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* String variable that will contain the value of the property.

*value* String expression that contains the new value of the property.

## Values

String containing the path of the image file.

Default: None

## Remarks

Set the ImageFile property to use images stored on disk by the SaveImage property or by VSaveImage for the current vision sequence. When you set ImageFile, ImageSource is automatically set to File.

To set the value to None from the Vision Guide window, select the ImageFile property, then press the Del key.

To specify the file in the USB memory of Compact Vision, put "CVUSB" at the beginning.

Compact Vision only allows the use of ASCII character file names.

Supported formats for bitmap files:

(All of the following conditions must be satisfied.)

Uncompressed Windows Bitmap

Bit depth: either of 8, 16 (RGB555), 24, or 32 (RGB888)

Following formats are not supported.

OS/2

Compressed files

Bit depth: 1, 4

## See Also

Sequence Tab, Vision Sequences, SaveImage Property, ImageSource Property

## ImageFileScale Property

### Applies To

Vision Sequence

### Description

Sets the scale for the image specified with the ImageFile property.

### Usage

**VGet** *Sequence.Object.ImageFileScale*, *var*

**VSet** *Sequence.Object.ImageFileScale*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

### Values

Real value specifying the image scale.

Default: 0

### Remarks

By default (`ImageFileScale = 0`), an image file is automatically scaled to `ImageSize`. Set `ImageFileScale` to specify a scale. For example, if `ImageFileScale` is set to 1, then the image from the file is scaled by a factor of 1 (no change). If `ImageFileScale` is set to 2, then the image is twice as large, and so on. The image is scaled from the top left corner and portions of the image that are outside of `ImageSize` are cropped. If the scaled image is smaller than `ImageSize`, then the remaining portions of the grab image are filled with black.

### See Also

Sequence Tab, ImageFile Property, ImageSize Property

# ImageSize Property

## Applies To

Vision Sequence

## Description

Sets and returns the image size of the search image.

## Usage

**VGet** *Sequence.ImageSize, var*

**VSet** *Sequence.ImageSize, value*

*Sequence* String variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

Specify the constant which represents the image size. For details on available constants, refer to *Remarks*.

## Remarks

Sets the image size for the search image. When ImageSource is Camera, ImageSize specifies the size of the image that is transferred from the camera. This allows for faster image acquisition when using lower resolutions. When ImageSource is File, ImageSize specifies the size of the file image.

You cannot set the ImageSize to a value that is larger than the maximum image size supported by the current camera.

Available constants are as follows:

Vision constant name	Resolution
VISION_IMAGESIZE_320X240	Image width: 320, image height: 240
VISION_IMAGESIZE_640X480	Image width: 640, image height: 480
VISION_IMAGESIZE_800X600	Image width: 800, image height: 600
VISION_IMAGESIZE_1024X768	Image width: 1024, image height: 768
VISION_IMAGESIZE_1280X1024	Image width: 1280, image height: 1024
VISION_IMAGESIZE_1600X1200	Image width: 1600, image height: 1200
VISION_IMAGESIZE_2048X1536	Image width: 2048, image height: 1536
VISION_IMAGESIZE_2560X1920	Image width: 2560, image height: 1920

## See Also

Vision Sequence, ImageFile Property, ImageFileScale Property

## ImageSource Property

### Applies To

Vision Sequence

### Description

Sets or returns the current image input source for the sequence.

### Usage

**VGet** *Sequence.ImageSource, var*

**VSet** *Sequence.ImageSource, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

### Values

1 – Camera (VISION\_IMAGESOURCE\_CAMERA)

2 – File (VISION\_IMAGESOURCE\_FILE)

Default: 1 - Camera

### Remarks

ImageSource sets the input source of the image data. When ImageSource is set to “File”, the image data will be acquired from the image file set in the ImageFile property and the sequence will be executed.

If the ImageFile property is set to a valid image file, then the ImageSource property will automatically be set to 2 - File.

### See Also

Sequence Tab, Vision Sequences, ImageFile Property, ImageSize Property

# ImportFont Property

Design time only

## Applies To

Vision Objects: OCR

## Description

Runs a file dialog from the Vision Guide GUI that allows you to import a font file.

## Remarks

Use the ImportFont property to import a font file that has been previously exported with the ExportFont property. You can import font files created in any project.

## See Also

ExportFont Property, OCR Object

## Index Property

### Applies To

Vision sequence, Vision object

### Description

Returns the index number of the vision sequence and vision object.

### Usage

**VGet** *Sequence*.**Index**, *var*

*Sequence* String variable containing a sequence name.

*var* Integer variable that contains the property.

### Values

Integer equals to or larger than 1

### Remarks

The Index property returns the index number of the vision sequence and vision object. The index number of the vision sequence is given automatically at the time of sequence creation. It will also be displayed on the sequence name drop-down list of the sequence tab. The index numbers of the vision objects are given in the order of the object definitions in the vision sequence and will be the order of object execution. It will also be displayed on the object name drop-down list of the vision object tab.

The Index value is necessary when running sequences from Extended Remote I/O.

### See Also

Sequence Tab, Object Tab

# InspectEndOffset Property

## Applies To

ArcInspector, LineInspector

## Description

Sets / returns the end inspection offset.

## Usage

**VGet** *Sequence.Object.InspectEndOffset, var*

**VSet** *Sequence.Object.InspectEndOffset, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real value in pixels or degrees.

Default: 15

## Remarks

When inspecting a line or arc, you may not always want to inspect the entire line. Use `InspectEndOffset` and `InspectStartOffset` to specify the portion of the line or arc you want to inspect.

For `LineInspector`, the offsets are in pixels. For `ArcInspector`, the offsets are in degrees.

## See Also

Arc Inspector Object, `InspectStartOffset` Property, Line Inspector Object

## InspectStartOffset Property

### Applies To

ArcInspector, LineInspector

### Description

Sets / returns the start inspection offset.

### Usage

**VGet** *Sequence.Object*.InspectStartOffset, *var*

**VSet** *Sequence.Object*.InspectStartOffset, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real value in pixels or degrees.

Default: 15

### Remarks

When inspecting a line or arc, you may not always want to inspect the entire line. Use `InspectEndOffset` and `InspectStartOffset` to specify the portion of the line or arc you want to inspect.

For `LineInspector`, the offsets are in pixels. For `ArcInspector`, the offsets are in degrees.

### See Also

Arc Inspector Object, `InspectEndOffset` Property, Line Inspector Object

# InvalidChar Property

## Applies To

Vision Objects: OCR

## Description

Sets / returns the invalid character used in the Text result.

## Usage

**VGet** *Sequence.Object.InvalidChar*, *var*

**VSet** *Sequence.Object.InvalidChar*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

String for one character.

Default: “?”

## Remarks

The InvalidChar property specifies the substitute character to be displayed after the OCR search if character could not be determined.

## See Also

Object Tab, OCR Object, CharToTeach Property, Text Result

## Iterations Property

### Applies To

Vision Objects: ImageOp

### Description

Sets/returns how many times to execute the image operation.

### Usage

**VGet** *Sequence.Object.Iterations*, *var*

**VSet** *Sequence.Object.Iterations*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

1 ~ 99

Default: 1

### Remarks

The Iterations property affects the following ImageOp operations:

Open, Close, Erode, Dilate, Smooth, Sharpen1, Sharpen2, Thin, Thicken.

### See Also

ImageOp Object, Operation Property

## ITFChecksumEnabled Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to use the checksum for Interleaved 2 of 5 (ITF) code type.

### Remarks

Sets whether to use the checksum for Interleaved 2 of 5 (ITF) code type.

Default: False

### See Also

CodeReading Object, ITFOutputChecksum Property

## ITFOutputChecksum Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the checksum value in an Interleaved 2 of 5 (ITF) Text result.

### Remarks

Sets whether to include the checksum value in an Interleaved 2 of 5 (ITF) Text result.

Default: True

### See Also

CodeReading Object, ITFChecksumEnabled Property

# KeepRGBRatio Property

## Applies To

Vision Objects: ImageOp

## Description

Sets / gets whether to maintain the ratio between R, G, B for the ColorStretch operation.

## Usage

**VGet** *Sequence.Object.KeepRGBRatio*, *var*

**VSet** *Sequence.Object.KeepRGBRatio*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – False Do not maintain the ratio between R, G, B for ColorStretch.

1 – True Maintain the ratio between R, G, B for ColorStretch.

Default: 1 – True

## Remarks

For the ImageOp ColorStretch operation, KeepRGBRatio sets whether to maintain the ratio between R, B, G values when performing the stretch. When KeepRGBRatio is false, the R, G, and B values are stretched individually.

## See Also

ImageOp Object, MaxRGB Property, MinRGB Property

## KernelHeight Property

### Applies To

Vision Objects: DefectFinder

### Description

Sets / returns the intensity of vertical de-noising for DefectFinder.

### Usage

**VGet** *Sequence.Object*.KernelHeight, *var*

**VSet** *Sequence.Object*.KernelHeight, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer from 1 - 15

Default: 3

### Remarks

Sets and returns the intensity of vertical de-noising for DefectFinder. Setting value ranges from 1 to 15. Setting the KernelHeight property to 15 sets the de-noising intensity to maximum. Although a large setting value reduces the influence of noise, it also rejects small defects. Therefore, detectable defect size needs to be larger than the KernelHeight value. Adjust to an appropriate value according to the size of defect to be detected.

### See Also

DefectFinder Object, KernelWidth Property

# KernelWidth Property

## Applies To

Vision Objects: DefectFinder

## Description

Sets / returns the intensity of horizontal de-noising at DefectFinder.

## Usage

**VGet** *Sequence.Object*.KernelWidth, *var*

**VSet** *Sequence.Object*.KernelWidth, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer from 1 - 15

Default: 3

## Remarks

Sets and returns the intensity of horizontal de-noising at DefectFinder. Setting value ranges from 1 to 15. Setting the KernelWidth property to "15" sets the de-noising intensity to maximum. Although a large setting value reduces the influence of noise, it also rejects small defects. Therefore, detectable defect size needs to be larger than the KernelWidth value. Adjust to an appropriate value according to the size of defect to be detected.

## See Also

DefectFinder Object, KernelHeight Property

## LabelBackColor Property

### Applies To

Vision Objects: All

### Description

Sets the background color for an object's label.

### Usage

**VGet** *Sequence.Object.LabelBackColor*, *var*

**VSet** *Sequence.Object.LabelBackColor*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

String containing the name of the label background color.

Default: "Transparent"

### Remarks

With some images, it is difficult to see the label because of the colors or shades of gray in the video image. Use `LabelBackColor` to make the label easier to read.

### See Also

PassColor Property, FailColor Property

# Lamp Property

## Applies To

Vision Calibration

## Description

Sets / returns the output bit used to turn on a lamp at the calibration.

## Usage

**VGet** *Calibration.Lamp, var*

**VSet** *Calibration.Lamp, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value of a valid standard output bit.

Default: None

## Remarks

Use the Lamp property to automatically turn on a lamp for calibration. Use the LampDelay property to allow time for a lamp to turn on before calibration continues.

## See Also

LampDelay Property, UpwardLamp Property

## LampDelay Property

### Applies To

Vision Calibration

### Description

Sets / returns the amount of time to wait for a calibration lamp to turn on.

### Usage

**VGet** *Calibration.LampDelay*, *var*

**VSet** *Calibration.LampDelay*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real number in seconds

### Remarks

Use the LampDelay property to allow time for a lamp to turn on before calibration continues. This is especially useful for a lighting apparatus, such as a fluorescent lamp, which requires time until the light source becomes stable.

### See Also

Lamp Property, MotionDelay Property, UpwardLamp Property

# Length Result

## Applies To

Vision Objects: ArcInspector, Line, LineFinder, LineInspector

## Description

Returns a distance between the starting and ending points of the line in millimeters in the camera coordinate system.

## Usage

**VGet** *Sequence.Object.Length*[(*result*)], *var*

**Sequence** Name of a sequence or string variable containing a sequence name.

**Object** Name of an object or string variable containing an object name. The object must exist in the specified sequence.

**var** Integer variable that will contain the value of the result.

**result** Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult.

## Values

The value returned is always in millimeters and may be from 0 on up. This actual number represents the length of the line in the camera coordinate system and depends upon calibration in order to return a value.

## Remarks

The Length result can only be returned if calibration has been performed because the length is measured in millimeters.

The Length result can be used for inspection and measurement applications where measurements are required. (For example, to measure spark plug gaps.)

For ArcInspector and LineInspector, Length is the length of a defect area.

## See Also

ArcInspector Object, Calibration, Line Object, LineInspector Object, Object Tab, PixelLength Result

## LineDirection Property

### Applies To

Vision Object: Contour  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Sets the direction of contour lines output with the Contour object.

### Usage

**VGet** *Sequence.Object.LineDirection*, *var*

**VSet** *Sequence.Object.LineDirection*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

- 1 – LeftToRight Vision constant: VISION\_LINEDIRECTION\_LEFTTORIGHT  
Outputs the contour from left to right in the search window.
- 2 – RightToLeft Vision constant: VISION\_LINEDIRECTION\_RIGHTTOLEFT  
Outputs the contour from right to left in the search window.

Default: 1

### Remarks

LineDirection sets the direction of contour lines output with the Contour object.

If LineDirection is set to LeftToRight, contour points will be output starting from the left edge when the edge search line is pointing vertically downwards.

If LineDirection is set to RightToLeft, contour points will be output starting from the right edge when the edge search line is pointing vertically downwards.

This property only applies when the ContourMode is set to Line.

### See Also

Contour Object

# LineObject Property

## Applies To

Vision Objects: LineInspector

## Description

Defines a LineFinder object that is used to locate a line for LineInspector to inspect.

## Usage

**VGet** *Sequence.Object.LineObject*, *var*

**VSet** *Sequence.Object.LineObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

None, or the name of a LineFinder object whose step number is prior to the step number of the LineInspector object.

Default: None

## Remarks

By default, the LineInspector inspects the line defined by the X1, Y1, X2, Y2 properties. You can search for the line first using a LineFinder by setting the LineObject property to the LineFinder.

## See Also

LineFinder Object, LineObjResult Property

## LineObjResult Property

### Applies To

Vision Objects: LineInspector

### Description

Specifies the result which the LineObject property uses.

### Usage

**VGet** *Sequence.Object.LineObjResult*, *var*

**VSet** *Sequence.Object.LineObjResult*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

LineObjResult can be set to "All", or you can specify which result to use. By using "All", results will be created for each LineObject result.

Default: 1

### Remarks

LineObjResult enables you to attach several objects to the results of one LineObject.

### See Also

LineInspector Object, LineObject Property

# LineObj1Result Property

## Applies To

Vision Object: Point

## Description

Sets / returns which result to use for the object specified with the LineObject1 property.

## Usage

**VGet** *Sequence.Object.LineObj1Result*, *var*

**VSet** *Sequence.Object.LineObj1Result*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0 – 100.

Default: 1

## Remarks

Sets / returns which result to use for the object specified with the LineObject1 property. The Line object itself does not have more than one result. However, when it is dependent other objects, such as a Frame object, it may have several results. In such cases, this property sets what number of the result values to be used.

## See Also

Line Object, LineObject1 Property, LineObject2 Property, LineObj2Result Property, Point Object, PointType Property, Object Tab

## LineObj2Result Property

### Applies To

Vision Object: Point

### Description

Sets / returns what number of detection results for Line objects set in the LineObject2 property to be used.

### Usage

**VGet** *Sequence.Object.LineObj2Result*, *var*

**VSet** *Sequence.Object.LineObj2Result*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value from 0 – 100.

Default: 1

### Remarks

Sets / returns which result to use for the object specified with the LineObject2 property. The Line object itself does not have more than one result. However, when it is dependent other objects, such as a Frame object, it may have several results. In such cases, this property sets what number of the result values to be used.

### See Also

Line Object, LineObject1 Property, LineObject2 Property, LineObj1Result Property, Point Object, PointType Property, Object Tab

# LineObject1 Property

## Applies To

Vision Objects: Point

## Description

Specifies the 1st Line object to use for defining the position of a Point object. (LineObject1 defines the line which is used by the PointType property for defining the position of the Point object. )

## Usage

**VGet** *Sequence.Object.LineObject1, var*

**VSet** *Sequence.Object.LineObject1, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

Any Line object which is located prior to the Point object can be specified as the LineObject1 property value.

Default: None

## Remarks

When a Point object is first created, the default LineObject1 property is set to None. However, if you want to position a point on the midpoint of a line, then the LineObject1 property defines which Line object to use. In this case the LineObject1 property must be set first and then the PointType property can be set to 1–MidPoint. A MidPoint of a line can only be specified for LineObject1. (i.e. you cannot specify the MidPoint of the LineObject2 property.)

LineObject1 can also be used to define the 1st of 2 lines when you want to define a Point object position as the intersection point between 2 lines. (LineObject2 defines the other Line to use for the intersection point.)

It is important to note that for each specific vision sequence, only those Line objects which are executed prior to the Point object in the vision sequence steps will be available to use as LineObject1. (The order of the vision object execution can be changed from the Sequence Tab.)

Click on the LineObject1 property Value Field and a drop down list will appear showing a list of available Line objects which can be used for the LineObject1 property. Click on one of the choices and the value field will be set accordingly.

## See Also

Line Object, LineObject2 Property, Object Tab, Point Object, PointType Property

## LineObject2 Property

### Applies To

Vision Objects: Point

### Description

Specifies the 2nd Line object to use for defining the position of a Point object when that position is defined by the intersection point of 2 lines. (LineObject1 and LineObject2 together define 2 lines whose intersection point can be defined as the position of the Point object.)

### Usage

**VGet** *Sequence.Object.LineObject2*, *var*

**VSet** *Sequence.Object.LineObject2*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

Any Line object which is located prior to the Point object can be specified as the LineObject2 property value.

Default: None

### Remarks

The LineObject2 property is only required when you want to define the position of a Point object as the position of intersection between 2 Lines. In this case LineObject1 must also specify a Line object before the PointType property can be set. Once the lines are defined for LineObject1 and LineObject2, the PointType property can be set to Intersection. However, if either the LineObject1 or LineObject property is not yet defined then an error will occur when trying to set the PointType property to Intersection.

It is important to note that for each specific vision sequence, only those Line objects which are executed prior to the Point object in the vision sequence steps will be available to use as LineObject2. (The order of the vision object execution can be changed from the Sequence Tab.)

Click on the LineObject2 property Value Field and a drop down list will appear showing a list of available Line objects which can be used for the LineObject2 property. Click on one of the choices and the value field will be set accordingly.

Set the LineObject2 property it is important to note that only those objects which are defined prior to the Point object are displayed in the drop down list. This helps reduce the chances of the user defining a Line object which isn't defined prior to the Point object.

Vision Guide automatically checks which vision objects may be used as LineObject2 and only displays those items in the LineObject2 drop down list.

### See Also

Line Object, LineObject1 Property, Object Tab, Point Object, PointType Property

# LJMMode Property

## Applies To

Vision Calibration

## Description

Sets or returns the mode for appropriately controlling the posture flag of the point data.

## Usage

**VGet** *Calibration.LJMMode*, *var*

**VSet** *Calibration.LJMMode*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Remarks

Appropriately controls the posture flag of the point data to prevent unintended rotation of the wrist. LJMM is not used when this property is set to 0.

Configurable values will vary depending on the type of robot in use. For details, refer to “LJM Functions” in the Spel+ Language Reference.

## See Also

VCal, VDefSetMotionRange Statement

## LuminanceCorrection Property

### Applies To

Vision Object: DefectFinder  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Sets the use of luminance correction preprocessing.

### Usage

**VGet** *Sequence.Object.LuminanceCorrection*, *var*

**VSet** *Sequence.Object.LuminanceCorrection*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

1 – None Vision constant: VISION\_LUMINANCECORRECTION\_NONE

No pre-processing

2 – Histogram Vision constant: VISION\_LUMINANCECORRECTION\_HISTGRAM

Luminance correction preprocessing using histogram mapping

Default: 1 - None

### Remarks

The LuminanceCorrection property sets the use of DefectFinder preprocessing.

When set to 1 - None, the detection process is performed without preprocessing.

When set to 2 - Histogram, brightness correcting preprocessing is performed by means of luminance correction preprocessing using histogram mapping.

Set this property to 2 - Histogram to prevent fluctuations in brightness from being treated as defects.

### See Also

DefectFinder Object

# MajorDiameter Result

## Applies To

Vision Objects: Blob, DefectFinder

## Description

Returns the major axis of the detected blob approximated to an ellipse.

## Usage

**VGet** *Sequence.Object.MajorDiameter*[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult.

## Values

Real number representing the pixel length of the major axis

## Remarks

Returns the major axis of the detected blob approximated to an ellipse. The unit is pixel.

The minor axis can be acquired by the MinorDiameter result.

## See Also

Blob Object, Area Result, Extrema Result, MinorDiameter Result

## MaxArea Property

### Applies To

Vision Objects: ArcInspector, Blob, Contour, DefectFinder, LineInspector

### Description

Defines the upper limit of detection for the an object. Blobs which exceed the MaxArea property value cannot be found.

### Usage

**VGet** *Sequence.Object.MaxArea*, *var*

**VSet** *Sequence.Object.MaxArea*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

### Values

1 - (the area of the search window)

Default: 100,000

### Remarks

For Blob and DefectFinder objects, the MinArea and MaxArea Properties set a range for the Blob object detection. Blobs which exceed the MaxArea property value cannot be found. (i.e. the Found result is returned as False.)

For ArcInspector and LineInspector objects, the MinArea and MaxArea Properties set a range for the area of a defect. Defects which exceed the MaxArea property value cannot be found. (i.e. the Found result is returned as False.)

When a new Blob object is created, the range between the MinArea property and MaxArea property is quite large because the default values are set at 25 and 100,000 respectively. This means that in most situations the Blob object will return a Found result as 1–True since the range for Blobs is so wide. In most applications it is useful to set a tighter range between the MinArea and MaxArea Properties but of course these values will vary from application to application. Set the MinArea and MaxArea Properties according to each application.

Do not set the range between MinArea and MaxArea too large. If the range is too large, it may result in false detection.

### See Also

Area Result, Blob Object, Contour Object, DefectFinder Object, LineInspector Object, ArcInspector Object, MinArea Property, MinMaxArea Property, Object Tab

## MaxError Result

### Applies To

Vision Objects: ArcFinder, BoxFinder, CornerFinder, LineFinder

### Description

Returns a distance from the found line or circular object to the furthest detection edge position.

### Usage

**VGet** *Sequence*.MaxError, *var*

*Sequence* String variable containing a sequence name.

*var* Integer variable that will contain the value of the result.

### Values

Actual value equals to or larger than 0 [Unit: pixel]

### Remarks

Returns a distance from the found line or circular object to the furthest detection edge position.

### See Also

LineFinder Object, ArcFinder Object, BoxFinder Object, CornerFinder, Object

## MaxFeretDiameter Result

### Applies To

Vision Object: Blob

### Description

Returns the maximum feret diameter of the found blob.

### Usage

**VGet** *Sequence*.MaxFeretDiameter, *var*

*Sequence* String variable containing a sequence name.

*var* Integer variable that will contain the value of the result.

### Values

Actual value equals to or larger than 0 [Unit: pixel]

### Remarks

Returns the maximum feret diameter (the maximum value among the feret diameters when dividing the direction into eight)

### See Also

Blob Object, Area Result, Angle Result, Extrema Result, MajorDiameter Result, MinorDiameter Result

# MaxLength Property

## Applies To

Vision Objects: Line

## Description

Defines the upper length limit for the Line object. For a Line to be found it must have a Length result shorter than the value set for MaxLength property.

## Usage

**VGet** *Sequence.Object.MaxLength*, *var*

**VSet** *Sequence.Object.MaxLength* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

0 or higher

Default: 1000

## Remarks

The MinLength and MaxLength Properties set a range of length of lines to be found for the Line object. (i.e. the Found result is returned as 0–False.)

This is useful when you want to gauge the length of a line in millimeters. To gauge a line in pixels, use the MinPixelLength and MaxPixelLength properties.

## See Also

Line Object, MinLength Property, MinPixelLength Property, MaxPixelLength Property, Object Tab

## MaxMoveDist Property

### Applies To

Vision Calibration

### Description

Sets or returns the maximum distance of move during calibration.

### Usage

**VGet** *Calibration.MaxMoveDist*, *var*

**VSet** *Calibration.MaxMoveDist*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real value from 0 to 500 (Unit: mm)

Default: 200

### Remarks

Limits distance of arm move during calibration. Distance of arm move is not restricted when this is set to 0.

### See Also

VCal, VDefSetMotionRange Statement

# MaxPixelLength Property

## Applies To

Vision Objects: Line

## Description

Sets the upper pixel length limit for the Line object. For a Line to be found it must have a PixelLength result shorter than the value set for MaxPixelLength property.

## Usage

**VGet** *Sequence.Object.MaxPixelLength*, *var*

**VSet** *Sequence.Object.MaxPixelLength* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

0 or higher in pixels

Default: 1000

## Remarks

The MinPixelLength and MaxPixelLength Properties set a range of length of lines to be found for the Line object. (i.e. the Found result is returned as 0–False.)

This is useful when you want to gauge the length of a line in pixels. To gauge a line in Millimeters, use the Length and MaxLength properties.

## See Also

Line Object, MinLength Property, MaxLength Property, MinPixelLength Property, Object Tab

## MaxRGB Property

### Applies To

Vision Objects: ImageOp

### Description

Defines the upper color limits for the ImageOp ColorStretch operation.

### Usage

**VGet** *Sequence.Object.MaxRGB*, *var*

**VSet** *Sequence.Object.MaxRGB* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

### Values

Long values formatted as RGB colors ( &Hrrggbb).

Default: &HFFFFFF (White)

### Remarks

MinRGB and MaxRGB are used to specify the minimum and maximum RGB values to use for the ColorStretch operation of ImageOpe.

### See Also

ImageOp Object, MinRGB Property

## MaxX Result

### Applies To

Vision Objects: Blob, DefectFinder, OCR

### Description

Returns the maximum X pixel coordinate of the blob extrema.

### Usage

**VGet** *Sequence.Object.MaxX* [(*result*)] , *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

The value returned is always in pixels and may be from 1 - video width.

### Remarks

The MinX, MaxX, MinY, and MaxY results return a blob's smallest enclosing rectangle that is aligned with the coordinate axes and completely encloses the blob. This rectangle is known as the Extrema.

### See Also

Area Result, Blob Object, Extrema Result, MaxY Result, MinX Result, MinY Result, DefectFinder Object, OCR Object, Object Tab

## MaxY Result

### Applies To

Vision Objects: Blob, DefectFinder, OCR

### Description

Returns the maximum Y pixel coordinate of the blob extrema.

### Usage

**VGet** *Sequence.Object.MaxY* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

### Values

Real number in pixels.

### Remarks

The `MinX`, `MaxX`, `MinY`, and `MaxY` results return a blob's smallest enclosing rectangle that is aligned with the coordinate axes and completely encloses the blob. This rectangle is known as the Extrema.

### See Also

Area Result, Blob Object, Extrema Result, MaxX Result, MinX Result, MinY Result, DefectFinder Object, OCR Object, Object Tab

# MinArea Property

## Applies To

Vision Objects: ArcInspector, Blob, Contour, DefectFinder, LineInspector

## Description

Sets the lower limit for the Blob object detection. Blobs which are smaller than the MinArea property value cannot be found.

## Usage

**VGet** *Sequence.Object.MinArea*, *var*

**VSet** *Sequence.Object.MinArea* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

## Values

1 - MaxArea in pixels

Default: 25

## Remarks

For Blob and DefectFinder objects, the MinArea and MaxArea Properties set a range for the Blob object detection. Blobs which exceed the MaxArea property value cannot be found. (i.e. the Found result is returned as False.)

For ArcInspector and LineInspector objects, the MinArea and MaxArea Properties set a range for the area of a defect. Defects which exceed the MaxArea property value cannot be found. (i.e. the Found result is returned as False.)

When a new Blob object is created, the range between the MinArea property and MaxArea property is quite large because the default values are set at 25 and 100,000 respectively. This means that in most situations the Blob object will return a Found result as 1–True since the range is large. In most applications it is useful to set a tighter range between the MinArea and MaxArea Properties but of course there values will vary from application to application. Set the MinArea and MaxArea Properties according to each application.

Do not set the range between MinArea and MaxArea too large. If the range is too large, it may result in false detection.

## See Also

Area Result, Blob Object, Contour Object, DefectFinder Object, LineInspector Object, ArcInspector Object, MaxArea Property, Object Tab

## MinorDiameter Result

### Applies To

Vision Objects: Blob, DefectFinder

### Description

Returns the minor axis of the detected blob approximated to an ellipse.

### Usage

**VGet** *Sequence.Object.MinorDiameter*[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the CurrentResult.

### Values

Real representing the pixel length of the minor axis

### Remarks

Returns the minor axis of the detected blob approximated to an ellipse. The unit is pixel.

The major axis can be acquired by the MajorDiameter result.

### See Also

Blob Object, Area Result, Extrema Result, MajorDiameter Result

# MinLength Property

## Applies To

Vision Objects: Line

## Description

Defines the lower length limit for the Line object. For a Line to be found it must have a Length result above the value set for MinLength property.

## Usage

**VGet** *Sequence.Object.MinLength*, *var*

**VSet** *Sequence.Object.MinLength* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real number in millimeters

Default: 0

## Remarks

The purpose of the MinLength and MaxLength Properties is to set a range for the Line object such that if a line length does not fit within the range then it is considered not found.

This is useful when you want to gauge the length of a line in millimeters. To gauge a line in pixels, see the MinPixelLength and MaxPixelLength properties.

## See Also

Line Object, MaxLength Property, MinPixelLength Property, MaxPixelLength Property, Object Tab

## MinMaxArea Property

Runtime only

### Applies To

Vision Objects: Blob

### Description

Defines the lower and upper Area limits for the Blob object. For a Blob to be found it must have an Area result greater than the MinArea property and less than the MaxArea property. (MinMaxArea property was added to allow easy manipulation of both the MinArea and MaxArea Properties from one function call in the SPEL<sup>+</sup> language.)

### Usage

**VGet** *Sequence.Object.MinMaxArea, minVar, maxVar*

**VSet** *Sequence.Object.MinMaxArea, minVar, maxVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*minVar* Long variable containing the minimum area to get from or set to the MinArea property

*maxVar* Long variable containing the maximum area to get from or set to the MaxArea property

### Values

All values are in pixels.

For details, refer to MaxArea Property or MinArea Property.

### Remarks

The purpose of the MinMaxArea property is to provide a single function call from the SPEL<sup>+</sup> language to allow the setting of both the MinArea and MaxArea Properties.

Do not set the MinMaxArea setting too large. If the range is too large, it may result in false detection.

### See Also

Area Result, Blob Object, MaxArea Property, MinArea Property, Object Tab

# MinPixelLength Property

## Applies To

Vision Objects: Line

## Description

Defines the lower length limit for the Line object. For a Line to be found it must have a PixelLength result above the value set for MinPixelLength property.

## Usage

**VGet** *Sequence.Object.MinPixelLength*, *var*

**VSet** *Sequence.Object.MinPixelLength* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

0 or higher actual number in pixels

Default: 0

## Remarks

The purpose of the MinPixelLength and MaxPixelLength Properties is to set a range for the Line object such that if a line pixel length does not fit within the range then it is considered not found.

This is useful when you want to gauge the length of a line in pixels. To gauge a line in Millimeters, see the MinLength and MaxLength properties. The default settings allow most lines to be found.

## See Also

Line Object, MaxLength property, MinLength property, MaxPixelLength property, Object Tab

## MinRGB Property

### Applies To

Vision Objects: ImageOp

### Description

Defines the lower color limits for the ImageOp ColorStretch operation.

### Usage

**VGet** *Sequence.Object.MinRGB*, *var*

**VSet** *Sequence.Object.MinRGB* , *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

### Values

Long values formatted as RGB colors ( &Hrrggbb).

Default: &H000000 (Black)

### Remarks

MinRGB and MaxRGB are used to specify the minimum and maximum RGB values to use for the ColorStretch operation of ImageOpe.

### See Also

ImageOp Object, KeepRGBRatio Property, MaxRGB Property

# MinX Result

## Applies To

Vision Objects: Blob, DefectFinder, OCR

## Description

Returns the minimum X pixel coordinate of the blob extrema.

## Usage

**VGet** *Sequence.Object.MinX* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Integer number in pixels

## Remarks

The MinX, MaxX, MinY, and MaxY results together return a blob's smallest enclosing rectangle that is aligned with the coordinate axes and completely encloses the blob. This rectangle is known as the extrema.

## See Also

Area Result, Blob Object, MaxX Result, MaxY Result, MinY Result, DefectFinder Object, OCR Object, Object Tab

## MinY Result

### Applies To

Vision Objects: Blob, DefectFinder, OCR

### Description

Returns the minimum Y pixel coordinate of the blob extrema.

### Usage

**VGet** *Sequence.Object.MinY* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

### Values

Integer number in pixels

### Remarks

The `MinX`, `MaxX`, `MinY`, and `MaxY` results together return a blob's smallest enclosing rectangle that is aligned with the coordinate axes and completely encloses the blob. This rectangle is known as the Extrema.

### See Also

Area Result, Blob Object, MaxX Result, MaxY Result, MinX Result, DefectFinder Object, OCR Object, Object Tab

# MissingEdgeType Property

## Applies To

Vision Object: ArcFinder, LineFinder, ArcInspector, LineInspector, BoxFinder, CornerFinder

## Description

Sets / returns how to handle missing edges.

## Usage

**VGet** *Sequence.Object.MissingEdgeType, var*

**VSet** *Sequence.Object.MissingEdgeType, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 – Interpolate. Determine the edge location from neighbor edges.

2 – StartPoint. Set the edge position to the start of the edge search.

3 – EndPoint. Set the edge position to the end of the edge search.

4 – Zero. Set the edge position to zero (on the line or arc).

Default: 1

## Remarks

Use MissingEdgeType to specify how to handle a missing edge during the edge search used to find or inspect a line or arc.

## See Also

ArcFinder Object, LineFinder Object, ArcInspector Object, LineInspector Object, BoxFinder Object, CornerFinder Object

## ModelColor Property

Runtime Only

### Applies To

Vision Objects: ColorMatch, ImageOp

### Description

Gets / sets the color of a model.

### Usage

**VGet** *Sequence.Object.ModelColor*, *var*

**VSet** *Sequence.Object.ModelColor*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression that will contain the new value of the property.

### Values

The color of the model in the format &Hrrggbb (red, green, blue) when ColorMode = RGB, and &Hhhssvv when ColorMode = HSV.

### Remarks

The ModelColor property is used to set the color of a model at runtime. Before setting the ModelColor, you must first set the CurrentModel property to the desired model.

### See Also

ColorMatch Object, CurrentModel Property, ImageOp Object, ModelName Property

# ModelColorTol Property

Runtime Only

## Applies To

Vision Objects: ColorMatch, ImageOp

## Description

Gets / sets the color tolerance of a model.

## Usage

**VGet** *Sequence.Object.ModelColorTol*, *var*

**VSet** *Sequence.Object.ModelColorTol*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the property.

*value* Long expression that will contain the new value of the property.

## Values

The color tolerance of the model. The value depends on the ColorMode setting.

If ColorMode = RGB, then the tolerance is a long value from 0 – 442. For RGB, the tolerance is the radius for a color circle where the red, green, and blue values of the ModelColor are at the center of the circle.

The default is 0 for ColorMatch and 10 for ImageOp ColorFilter.

If ColorMode = HSV, then the tolerance is in the format *&Hhhssvv*, where *hh* is the hue tolerance from 0 – 359, *ss* is the saturation tolerance from 0 – 255, and *vv* is the value tolerance from 0 – 255.

The default is 50 (0,0,50).

## Remarks

The ModelColorTol property is used to set the color tolerance of a model at runtime.

## See Also

ImageOp Object, ColorMatch Object, ModelColor Property

## ModelName Property

Runtime Only

### Applies To

Vision Objects: ColorMatch

### Description

Gets / sets the name of a model.

### Usage

**VGet** *Sequence.Object.ModelName*, *var*

**VSet** *Sequence.Object.ModelName*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression that will contain the new value of the property.

### Values

The name of the model.

### Remarks

The ModelName property is used to set the name of a model at runtime. Before you can set the ModelName, you must first set CurrentModel to the desired model.

### See Also

ColorMatch Object, CurrentModel Property, ModelColor Property

# ModelObject Property

## Applies To

Vision Objects: ColorMatch, Correlation, Geometric, Polar

## Description

Determines which model to use for searching.

## Usage

**VGet** *Sequence.Object.ModelObject*, *var*

**VSet** *Sequence.Object.ModelObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

**Self** Use the model for this object to search with.

**objectName** Use the model for this object.

Default: Self

## Remarks

The ModelObject property enables you to use one model for several objects of the same type. For example, if you have 5 polar objects that all search for the same part, you can teach the model for the first polar object, then set the ModelObject for the remaining polar objects to "Polar01" (the first polar object).

Note that you cannot set the ModelObject property to an object whose ModelObject property is not 'Self'.

## See Also

ColorMatch Object, Correlation Object, Geometric Object, Polar object, Object Tab

## ModelOK Property

Runtime only

### Applies To

Vision Objects: ColorMatch, Correlation, DefectFinder, Geometric, OCR, Polar

### Description

Returns the status of an object's model.

### Usage

**VGet** *Sequence.Object.ModelOK*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

### Values

0 – False

1 – True

### Remarks

If the model has been taught, ModelOK will return 1–True. You can ensure that the sequence will run before running a sequence.

### See Also

Correlation Object, Geometric Object, ModelObject Property, OCR Object, Polar object, DefectFinder Object, VTeach

# ModelOrgAutoCenter Property

## Applies To

Vision Objects: Correlation, Geometric

## Description

A model has a fixed reference point by which a location of the model in the image is indicated. This point is referred to as the model origin. The ModelOrgAutoCenter property sets the model origin at the center of the model window automatically.

## Usage

**VGet** *Sequence.Object.ModelOrgAutoCenter, var*

**VSet** *Sequence.Object.ModelOrgAutoCenter, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Do not cause the model origin to be automatically positioned

1 – True Automatically position the model origin at the center of the model window

Default: 1 – True

## Remarks

The model origin can be set anywhere within the region defined by the model window. The origin's coordinates define the model origin relative to the model window's upper left corner, that is, relative to the location of element [0][0] of the model window that defines the model.

The ModelOrgAutoCenter property sets the model origin at the center of the model window whenever the model window is moved or resized.

If the ModelOrgAutoCenter property is set to 1–True, then the ModelOrgX and ModelOrgY Properties cannot be used to reposition the model origin.

## See Also

Anatomy of a Vision Object, Correlation Object, Geometric Object, ModelOrgX Property, ModelOrgY Property, Object Tab

## ModelOrgFindCenter Property

### Applies To

Vision Object: Geometric

### Description

A model has a fixed reference point by which a location of the model in the image is indicated. This point is referred to as the model origin. The ModelOrgFindCenter property sets the model origin at the rotation center of the model edge automatically.

### Usage

**VGet** *Sequence.Object. ModelOrgFindCenter*, *var*

**VSet** *Sequence.Object. ModelOrgFindCenter*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not set the model origin at the center of the model edge

1 – True Set the model origin at the center of the model edge

Default: 0 – False

### Remarks

The model origin can be set anywhere within the region defined by the model window. The origin's coordinates define the model origin relative to the model window's upper left corner, that is, relative to the location of element [0][0] of the model window that defines the model.

The ModelOrgFindCenter property sets the model origin at the rotation center of the model edge. The model origin is not updated automatically when the model window is moved or resized. Use this property to set the model origin at the rotation center of the mode edge as needed.

If this property is set to “True” when the ModelOrgAutoCenter property is “1–True”, then the ModelOrgAutoCentere property will be automatically set to “0 - False”.

If this property is set to “True”, the model origin is updated, and then the property value returns to “False”. That is, the value obtained by VGet is always “False”.

### See Also

Anatomy of a Vision Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgX Property, ModelOrgY Property, Object Tab

# ModelOrgX Property

## Applies To

Vision Objects: Correlation, Geometric

## Description

A model has a fixed reference point by which a location of the model in the image is indicated. This point is referred to as the model origin. The ModelOrgX property contains the X coordinate value of the model origin.

## Usage

**VGet** *Sequence.Object.ModelOrgX*, *var*

**VSet** *Sequence.Object.ModelOrgX*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real number or expression for the new value of the property.

## Values

The ModelOrgX property can be set between 0–2047. It should be noted that the ModelOrgX defines the X coordinate of the model origin with respect to the model's upper left corner.

Default: The model origin is set in the center of the model window when a new object is created.

## Remarks

The model origin can be set anywhere within the model window Region. The origin's coordinates define the model origin relative to the model's upper left corner, that is, relative to the location of element [0][0] of the image that defines the model.

When a new Correlation is created, the model origin is set to the center of the model window. However, the user may modify this position by entering new X and Y positions to the ModelOrgX and ModelOrgY properties or by simply clicking on the model origin (the crosshair shown in the middle of the model window) and moving it to the desired position.

The model origin also can be automatically changed by setting the ModelOrgAutoCenter property to “1–True”. If the ModelOrgAutoCenter property is set to “1–True” then the model origin is automatically set to the center of the model window.

If the ModelOrgAutoCenter property is set to “1–True”, the ModelOrgX property cannot be used to reposition the model origin.

## See Also

Anatomy of a Vision Object, Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgFindCenter Property, ModelOrgY Property, Object Tab

## ModelOrgY Property

### Applies To

Vision Objects: Correlation, Geometric

### Description

A model has a fixed reference point by which a location of the model in the image is indicated. This point is referred to as the model origin. The ModelOrgY property contains the Y coordinate value of the model origin.

### Usage

**VGet** *Sequence.Object.ModelOrgY*, *var*

**VSet** *Sequence.Object.ModelOrgY*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real number or expression for the new value of the property.

### Values

Basically, the ModelOrgY property can be set between 0-2047. However, it should be noted that the ModelOrgY defines the Y coordinate of the model origin with respect to the model's upper left corner.

Default: The model origin is set in the center of the model window when a new object is created.

### Remarks

The model origin can be set anywhere within the model's bounds. The origin's coordinates define the model origin relative to the model's upper left corner, that is, relative to the location of element [0][0] of the image that defines the model.

When a new Correlation is created, the model origin is set to the center of the model window. However, the user may modify this position by entering new X and Y positions to the ModelOrgX and ModelOrgY Properties or by simply clicking on the model origin (the crosshair shown in the middle of the model window) and moving it to the desired position.

The model origin also can be automatically changed by setting the ModelOrgAutoCenter property to "1-True". If the ModelOrgAutoCenter property is set to "1-True" then the model origin is automatically set to the center of the model window.

If the ModelOrgAutoCenter property is set to "1-True", the ModelOrgY property cannot be used to reposition the model origin.

### See Also

Anatomy of a Vision Object , Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgFindCenter Property, ModelOrgX Property, Object Tab

# ModelWin Property

Runtime only

## Applies To

Vision Objects: Correlation, Geometric, ImageOp, OCR

## Description

Defines the position and size of the model window.

## Usage

**VGet** *Sequence.Object.ModelWin, LeftVar, TopVar, WidthVar, HeightVar*

**VSet** *Sequence.Object.ModelWin, LeftVar, TopVar, WidthVar, HeightVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*LeftVar* Integer variable representing the leftmost position of the model window (in Pixels).

*TopVar* Integer variable representing the uppermost position of the model window (in Pixels).

*WidthVar* Integer variable representing the width of the model window (in Pixels).

*HeightVar* Integer variable representing the height of the model window (in Pixels).

## Values

All Values are in Pixels. See the ModelWinTop, ModelWinLeft, ModelWinWidth, and ModelWinHeight Properties for exact value data.

## Remarks

The ModelWin property was added to provide easy access to the ModelWinTop, ModelWinLeft, ModelWinWidth and ModelWinHeight Properties from the SPEL<sup>+</sup> Language. The ModelWin property allows the setting of all 4 Properties. There are cases where the user may want to define the position and size of the model window dynamically and for that reason the ModelWin property was created.

The ModelWin property can be applied to the Correlation, Geometric, ImageOp, and OCR objects. Each of these object types have rectangular model windows used to define the position and size of the model. For ImageOp, the Operation property must first be set to ColorFilter.

## See Also

Correlation Object, Geometric Object, ImageOp Object, ModelWinHeight Property, ModelWinLeft Property, ModelWinTop Property, ModelWinWidth Property, Object Tab, OCR Object

## ModelWinAngle Property

### Applies To

Vision Objects: Correlation, Geometric

### Description

Sets / returns the model window angle.

### Usage

**VGet** *Sequence.Object.ModelWinAngle*, *var*

**VSet** *Sequence.Object.ModelWinAngle*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

### Values

Real value within a range of  $\pm 180$  degrees

### Remarks

Sets and returns the model window angle. Setting values are valid only when ModelWinType is set to RotatedRectangle.

### See Also

ModelWinCenterX Property, ModelWinCenterY Property, ModelWinType Property, SearchWinCenterX Property, SearchWinCenterY Property, SearchWinType Property

# ModelWinCenterX Property

## Applies To

Vision Objects: Correlation, Geometric

## Description

Sets and returns the X coordinate value on the center of the model window.

## Usage

**VGet** *Sequence.Object.ModelWinCenterX, var*

**VSet** *Sequence.Object.ModelWinCenterX, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer ranging from 0 to “search window width - 1” in pixels

## Remarks

Sets and returns the X coordinate value on the center of the model window. Setting values are automatically updated when the model window is removed or resized.

## See Also

ModelWinAngle Property, ModelWinCenterX Property, ModelWinCenterY Property, ModelWinType Property, SearchWinCenterX Property, SearchWinCenterY Property, SearchWinType Property, SearchWinHeight Property, SearchWinWidth Property

## ModelWinCenterY Property

### Applies To

Vision Objects: Correlation, Geometric

### Description

Sets and returns the Y coordinate value on the center of the model window.

### Usage

**VGet** *Sequence.Object.ModelWinCenterY, var*

**VSet** *Sequence.Object.ModelWinCenterY, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer ranging from 0 to “search window height - 1” in pixels

### Remarks

Sets and returns the Y coordinate value on the center of the model window. Setting values are automatically updated when the model window is removed or resized.

### See Also

ModelWinAngle Property, ModelWinCenterX Property, ModelWinCenterY Property, ModelWinType Property, SearchWinCenterX Property, SearchWinCenterY Property, SearchWinType Property, SearchWinHeight Property, SearchWinWidth Property

# ModelWinHeight Property

## Applies To

Vision Objects: Correlation, Geometric, ImageOp, OCR

## Description

Defines the height of the model window.

## Usage

**VGet** *Sequence.Object.ModelWinHeight*, *var*

**VSet** *Sequence.Object.ModelWinHeight*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

Integer number in pixels from 10 – 2048.

## Remarks

The model window can be set in the Search Window.

The Correlation, Geometric, ImageOp, and OCR objects have rectangular model windows which define the position and size of the model to be taught. The ModelWinHeight property is set automatically when the user drags the upper or lower horizontal sides of the model window.

Keep in mind that a larger model window causes the taught model to be larger, which in turn can cause the execution time of the object to increase.

The ModelWinHeight property can be set by directly inputting the value from the Vision Guide window Object tab and by using the SPEL<sup>+</sup> Language. Also, it can be set by clicking on the upper or lower horizontal sides of the model window and then dragging them vertically.

To set a value from the property tab, click on the ModelWinHeight property Value Field and simply enter in the value. Once you move the cursor off the value field, the ModelWinHeight will be adjusted for the associated vision object.

## See Also

Anatomy of a Vision Object , Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgY Property, ModelOrgY Property, ModelWin Property, ModelWinLeft Property, ModelWinTop Property, ModelWinWidth Property, Object Tab, OCR Object

## ModelWinLeft Property

### Applies To

Vision Objects: Correlation, Geometric, ImageOp, OCR

### Description

Defines the left coordinate of the upperleft corner of the model window.

### Usage

**VGet** *Sequence.Object.ModelWinLeft, var*

**VSet** *Sequence.Object.ModelWinLeft, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

### Values

Integer number in pixels from 0- video width

### Remarks

The ModelWinLeft property is available for the Correlation, Geometric, ImageOp, and OCR objects only since those are the only vision objects which use a model to define a pattern to search for. The model window can be set in the Search Window.

The Correlation, Geometric, ImageOp, and OCR objects have rectangular model windows which define the position and size of the model to be taught. The ModelWinLeft property is set automatically when the user drags the entire model window to a new location or when the leftmost side of the model window is dragged to resize the model window.

The ModelWinLeft property can be set by directly inputting the value from the Vision Guide window Object tab and by using the SPEL<sup>+</sup> Language. Also, it can be set by clicking on the upper or lower horizontal sides of the model window and then dragging them vertically. You can also click on the leftmost vertical side of the model window at the center of the vertical line where the leftmost window handle is (the small square on the left vertical side of the Model window). You will see the mouse pointer change to a two direction horizontal arrow. Now drag the leftmost vertical side of the model window and you will see the size of the model change. Release the mouse button when you want to set the position.

To set a value from the property tab, click on the ModelWinLeft property's Value Field and simply enter in the value. Once you move the cursor off the value field, the ModelWinLeft will be adjusted for the associated vision object.

### See Also

Anatomy of a Vision Object, Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgX Property, ModelOrgY Property, ModelWin Property, ModelWinHeight Property, ModelWinTop Property, ModelWinWidth Property, Object Tab, OCR Object

# ModelWinTop Property

## Applies To

Vision Objects: Correlation, Geometric, ImageOp, OCR

## Description

Defines the top coordinate of the upperleft corner of the model window.

## Usage

**VGet** *Sequence.Object.ModelWinTop*, *var*

**VSet** *Sequence.Object.ModelWinTop*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

Integer number in pixels from 0 - video height

## Remarks

The ModelWinTop property is available for the Correlation, Geometric, ImageOp, and OCR objects only since those are the only vision objects which use a model to define a pattern to search for. The model window can be set in the Search Window.

The Correlation, Geometric, ImageOp, and OCR objects have rectangular model windows which define the position and size of the model to be taught. The ModelWinTop property is set automatically when the user drags the entire model window to a new location or when the topmost side of the model window is dragged to resize the model window.

The ModelWinTop property can be set by directly inputting the value from the Vision Guide window Object tab and by using the SPEL<sup>+</sup> Language. Also, it can be set by clicking on the upper or lower horizontal sides of the model window and then dragging them vertically.

You can also click on the uppermost horizontal side of the model window at the center of the vertical line where the uppermost side window handle is (the small square on the upper horizontal side of the Model window). You will see the mouse pointer change to a two direction vertical arrow. Now drag the uppermost horizontal side of the model window and you will see the size of the model change. Release the mouse button when you want to set the position.

To set a value from the property tab, click on the ModelWinTop property's Value Field and simply enter in the value. Once you move the cursor off the value field, the ModelWinLeft will be adjusted for the associated vision object.

## See Also

Anatomy of a Vision Object, Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgX Property, ModelOrgY Property, ModelWin Property, ModelWinHeight Property, ModelWinLeft Property, ModelWinWidth Property, Object Tab, OCR Obejct

# ModelWinType Property

## Applies To

Vision Object: Correlation, Geometric

## Description

Sets / returns the model window type.

## Usage

**VGet** *Sequence.Object.ModelWinType, var*

**VSet** *Sequence.Object.ModelWinType, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Following integer values representing the model window type

1 - Rectangle (VISION\_WINTYPE\_RECTANGLE)

2 - RotatedRectangle (VISION\_WINTYPE\_ROTATEDRECT)

3 - Circle (VISION\_WINTYPE\_CIRCLE)

## Remarks

Sets and returns the model window type. Available values are as follows:

1 - Rectangle                      Rectangular model window (angle designation invalid)

2 - RotatedRectangle            Rectangular model window (angle designation valid)

3 - Circle                          Circular model window

## See Also

ModelWinAngle Property, ModelWinCenterX Property, ModelWinCenterY Property, ModelWinType Property, SearchWinCenterX Property, SearchWinCenterY Property, SearchWinHeight Property, SearchWinWidth Property

# ModelWinWidth Property

## Applies To

Vision Objects: Correlation, Geometric, ImageOp, OCR

## Description

Defines the width of a model window.

## Usage

**VGet** *Sequence.Object.ModelWinWidth*, *var*

**VSet** *Sequence.Object.ModelWinWidth*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression that contains the new value of the property.

## Values

Integer number in pixels from 10 – 2048.

## Remarks

The model window can be set in the Search Window.

The ModelWinWidth property is set automatically when the user drags the entire model window to a new location or when one of the horizontal sides of the model window is dragged to resize the model window.

Keep in mind that larger model windows cause the taught Model to be larger, which in turn may cause the execution time of the object to increase.

The ModelWinWidth property can be set by directly inputting the value from the Vision Guide window Object tab and by using the SPEL<sup>+</sup> Language. Also, it can be set by clicking on the upper or lower horizontal sides of the model window and then dragging them vertically.

You can also click on the left or right vertical side of the model window at the center of the vertical line where the window handle is (the small square on the upper horizontal side of the Model window). You will see the mouse pointer change to a two direction horizontal arrow. Now drag the side of the model window and you will see the of the model change in width. Release the mouse button when you want to set the position.

To set a value from the property tab, click on the ModelWinWidth property's Value Field and simply enter in the value. Once you move the cursor off the value field, the ModelWinLeft will be adjusted for the associated vision object.

## See Also

Anatomy of a Vision Object, Correlation Object, Geometric Object, ModelOrgAutoCenter Property, ModelOrgX Property, ModelOrgY Property, ModelWin Property, ModelWinHeight Property, ModelWinLeft Property, ModelWinTop Property, Object Tab, OCR Obejct

## MotionDelay Property

### Applies To

Vision Calibration

### Description

Sets / returns the amount of time to wait after each robot motion during the calibration cycle.

### Usage

**VGet** *Calibration.MotionDelay*, *var*

**VSet** *Calibration.MotionDelay*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number in milliseconds

Default: 500

### Remarks

Use the MotionDelay property to allow settling time after the robot is moved during a calibration cycle. During calibration, it is important that the robot, tooling, and table are not moving when the vision system is acquiring images. It is recommended that values below 500 milliseconds should not be used.

### See Also

LampDelay Property, RobotAccel Property, RobotSpeed Property

# Name Property

## Applies To

Vision Sequence  
Vision Calibration  
Vision Objects: All

## Description

All vision objects, sequences, and calibrations must have a name. The name is then used to refer to the individual vision object, sequence or calibration.

## Usage

**VGet** *Sequence.Object.Name*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

## Remarks

Upon the creation of a new vision sequence or calibration, a dialog is displayed requesting a name. This name is then set as the Name property value. The user can modify this name from the Vision Guide GUI by changing the Name property value.

Upon the creation of a new vision object such as a Blob, Correlation, etc., a name is automatically assigned to the object. The name used is based on the object type with a numeric value appended to the end of the name. For example, the following names might have been created for a specific vision sequence: Blob01, Corr01, Blob02, Blob03, Corr02, Line01. You can modify the name by changing the Name property value from the Vision Guide GUI.

You cannot change the name of a sequence, calibration, or object at runtime.

## See Also

Blob Object, CodeReader Object, Correlation Object, Edge Object, Frame Object, Geometric Object, ImageOp Object, Line Object, Object Tab, OCR Object, Point Object, Polar Object, Sequence Tab

## NumberFound Result

### Applies To

Vision Objects: ArcInspector, Blob, CodeReader, Contour, Correlation, DefectFinder, Edge, Geometric, LineInspector, OCR

### Description

Returns the number of features found within a single Search Window.

### Usage

**VGet** Sequence.Object.**NumberFound**, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

### Values

Valid number found for all objects is 0 - NumberToFind property value.

### Remarks

Blob, Correlation, Edge, and Geometric objects support the finding of multiple features within a single Search Window. The NumberToFind property defines how many features to search for.

The NumberFound result returns how many features were actually found.

The NumberFound result is a special result. It always returns the number of features that were found for the specified vision object regardless of the setting of the CurrentResult property. The result list shows the results selected by the CurrentResult property.

Blob results are ordered by largest found blob to smallest found blob. (i.e. result record 1 (CurrentResult = 1) contains the results for the largest blob.)

Correlation results are ordered by highest Score result to lowest Score result. (i.e. result Record 1 (CurrentResult =1) contains the results for the feature with the highest score.)

The order of detection can be changed by the Sort property setting.

### See Also

ArcInspector Object, Blob Object, Contour Objec, Correlation Object, CurrentResult Property, DefectFinder Object, Edge Object, Found Result, Geometric Object, LineInspector Object, NumberToFind Property, Sort Property, CodeReader Object, OCR Object, Object Tab

## Example

The following SPEL<sup>+</sup> language example runs a vision sequence called *mtest* which contains a Correlation object called *Corr01*. *Corr01* has been defined to find multiple features (3).

The following program will run the sequence and make sure that the proper number of features (3) was found for *Corr01* and then print the Score result in descending order.

```
Function main

  #define NUM_TO_FIND 3

  Boolean numfound
  Integer score

  VRun mtest
  VGet mtest.Corr01.NumberFound, numfound
  If numfound = NUM_TO_FIND Then
    Print "The Proper Number of features(3) were found"
  Else
    Print "Only (", numfound, ") features were found"
    Exit Function
  EndIf
  VGet mtest.Corr01.Score(1), score
  Print "1st feature score (Best):  ", score

  VGet mtest.Corr01.Score(2), score
  Print "2nd feature score (Medium): ", score

  VGet mtest.Corr01.Score(3), score
  Print "3rd feature score (Worst):  ", score
Fend
```

## NumberOfEdges Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, LineFinder, LineInspector

### Description

Sets and returns the number of edges detected when detecting line segments and circular arcs.

### Usage

**VGet** *Sequence.Object.NumberOfEdges*, *var*

**VSet** *Sequence.Object.NumberOfEdges*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer from 5 - 99

Default: 5 for LineFinder, ArcFinder, BoxFinder, CornerFinder

20 for LineInspector and ArcInspector

### Remarks

Sets the number of segments to perform edge detection in equal divisions within the search range.

By adding segments, detection becomes more robust to the change in the input image (lighting and noise). However, it takes longer detection time. Set the property according to the actual system.

### See Also

ArcFinder Object, LineFinder Object, ArcInspector Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object, EdgeRobotXYU Result

# NumberOfModels Property

Runtime Only

## Applies To

Vision Objects: ColorMatch, ImageOp

## Description

Gets / sets the number of models used by the object.

## Usage

**VGet** *Sequence.Object.NumberOfModels*, *var*

**VSet** *Sequence.Object.NumberOfModels*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

The number of models defined for the object.

## Remarks

The NumberOfModels property is used at runtime to set the number of models for a ColorMatch or ImageOp object. After setting NumberOfModels, you can use CurrentModel and VTeach to teach each color model.

## See Also

CurrentModel Property, ColorMatch Object, ImageOp Object, VTeach

## NumberOfResults Property

Runtime Only

### Applies To

Vision Objects: All

### Description

Gets the number of results for an object.

### Usage

**VGet** Sequence.Object.**NumberOfResults**, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

### Values

The number of results for the object.

### Remarks

The NumberOfResults property is used at runtime to determine the total number of results (found and not found).

### See Also

CurrentResult Property, NumberFound Result

# NumberToFind Property

## Applies To

Vision Objects: ArcInspector, Blob, CodeReader, Contour, Correlation, DefectFinder, Edge, Geometric, LineInspector

## Description

Defines the number of features to search for within a single search window.

## Usage

**VGet** *Sequence.Object.NumberToFind*, *var*

**VSet** *Sequence.Object.NumberToFind*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Valid entries are from 0 (All) to 100 (for the CodeReader object, the maximum is 8).

Default: 1

## Remarks

The ArcInspector, Blob, Correlation, DefectFinder, Edge, Geometric, and LineInspector objects support finding multiple features within a single Search Window. The NumberToFind property defines how many.

Since many applications require that only 1 feature be found within a Search Window, the default value of the NumberToFind property is set to 1.

When working in the Vision Guide Development Environment, you will notice that the Results List on the Object tab will display a heading like "Result (1 of 15)". This means that the system tried to find 15 features (as defined by the NumberToFind property) and the Result List will display the results for item 1.

If you want to see the results for one of the other results, just change the CurrentResult property value to indicate which result you want to examine.

Blob results are ordered according to the SizeToFind and Sort properties.

If NumberToFind is set to "0 - All", then all possible results are found, up to the maximum detection number. The maximum detection number varies for each object as shown below.

Object name	Max. detection number
CodeReader	4
Other	100

Correlation and Geometric results are ordered by highest Score result to lowest Score result when Sort is None. (i.e. result Record 1 (CurrentResult =1) contains the results for the feature with the highest score.)

## See Also

ArcInspector Object, Blob Object, Contour Object, Correlation Object, CurrentResult property, DefectFinder Object, Edge Object, Found Result, Geometric Object, LineInspector Object, NumberFound Result, Object Tab

### Example

The following SPEL<sup>+</sup> language example runs a vision sequence called *mtest* which contains a Correlation object called *Corr01*. The *NumberToFind* value for *Corr01* is set using *VSet*.

The following program will run the sequence and make sure that the proper number of features (3) was found for *Corr01* and then print the *Score* result in descending order.

```
Function main

    #define NUM_TO_FIND 3

    Boolean numfound
    Integer score

    VSet mtest.Corr01.NumberToFind, NUM_TO_FIND
    VRun mtest
    VGet mtest.Corr01.NumberFound, numfound
    If numfound = NUM_TO_FIND Then
        Print "The Proper Number of features(3) were found"
    Else
        Print "Only (", numfound, ") features were found"
        Exit Function
    EndIf
    VGet mtest.Corr01.Score(1), score
    Print "1st feature score (Best):  ", score

    VGet mtest.Corr01.Score(2), score
    Print "2nd feature score (Medium): ", score

    VGet mtest.Corr01.Score(3), score
    Print "3rd feature score (Worst):  ", score
Fend
```

# Objects Property

Runtime Only

## Applies To

Vision Sequence

## Description

Array of objects in a sequence. Used to access object properties and results with an index.

## Usage

**VGet** *Sequence*.**Objects**(*index*).*Property*, *var*

**VGet** *Sequence*.**Objects**(*index*).*Result*, *var*

**VSet** *Sequence*. **Objects**(*index*) .*Property*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*index* Integer expression containing the index of an object in the specified sequence.

*Property* Name of the object property to access.

*Result* Name of the object result to access.

*var* Variable that will contain the value of the property or result. The data type depends on which property or result is specified.

*value* Expression for the new value of the property. The data type depends on which property is specified.

## Remarks

Use the Objects property to access the objects in a sequence by using an index instead of a name.

## See Also

Count Property, Type Property

# Operation Property

## Applies To

Vision Objects: ImageOp

## Description

Sets which image operation to perform.

## Usage

**VGet** *Sequence.Object.Operation*, *var*

**VSet** *Sequence.Object.Operation*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

- |                  |   |
|------------------|---|
| 1 - Open         | Vision Constant: VISION_OPERATION_OPEN<br>Performs an opening-type morphological operation. This is an erosion followed by a dilation. The number of iterations is determined by the Iterations property. |
| 2 - Close        | Vision Constant: VISION_OPERATION_CLOSE<br>Performs a closing-type morphological operation. This is a dilation followed by an erosion. The number of iterations is determined by the Iterations property. |
| 3 - Erode        | Vision Constant: VISION_OPERATION_ERODE<br>Performs an erosion-type morphological operation. The number of iterations is determined by the Iterations property.   |
| 4 - Dilate       | Vision Constant: VISION_OPERATION_DILATE<br>Performs a dilation-type morphological operation. The number of iterations is determined by the Iterations property.  |
| 5 - Smooth       | Vision Constant: VISION_OPERATION_SMOOTH<br>Performs a smoothing type convolution operation. The number of iterations is determined by the Iterations property.   |
| 6 - Sharpen1     | Vision Constant: VISION_OPERATION_SHARPEN1<br>Performs a sharpen type convolution operation. The number of iterations is determined by the Iterations property.   |
| 7 - Sharpen2     | Vision Constant: VISION_OPERATION_SHARPEN2<br>Performs a sharpen type convolution operation. The number of iterations is determined by the Iterations property.   |
| 8 - HorizEdge    | Vision Constant: VISION_OPERATION_HORIZEDGE<br>Highlights horizontal edges.   |
| 9 - VertEdge     | Vision Constant: VISION_OPERATION_VERTEDGE<br>Highlights vertical edges.  |
| 10 - EdgeDetect1 | Vision Constant: VISION_OPERATION_EDGEDETECT1<br>Highlights edges.  |
| 11 - EdgeDetect2 | Vision Constant: VISION_OPERATION_EDGEDETECT2<br>Highlights edges.  |

12 - LaPlaceEdge1	Vision Constant: VISION_OPERATION_LAPLACE1 Highlights edges.
13 - LaPlaceEdge2	Vision Constant: VISION_OPERATION_LAPLACE2 Highlights edges.
14 - Thin	Vision Constant: VISION_OPERATION_THIN Thins blobs in the image.
15- Thicken	Vision Constant: VISION_OPERATION_THICKEN Thickens blobs in the image.
16 - Binarize	Vision Constant: VISION_OPERATION_BINARIZE Binarizes the image according to the ThresholdLow and ThresholdHigh settings.
17 - Rotate	Vision Constant: VISION_OPERATION_ROTATE Rotates the image according to the AngleObject or RotationAngle settings. If AngleObject is Screen, then the rotation angle is determined by the RotationAngle property. Otherwise, the rotation angle is determined by the Angle result of the AngleObject. The rotation is counter-clockwise for positive angles.
18 - FlipHoriz	Vision Constant: VISION_OPERATION_FLIPHORIZ Flips the image from left to right.
19 - FlipVert	Vision Constant: VISION_OPERATION_FLIPVERT Flips the image from top to bottom.
20 - FlipBoth	Vision Constant: VISION_OPERATION_FLIPBOTH Flips the image horizontally and vertically.
21 - ColorFilter	Vision Constant: VISION_OPERATION_COLORFILTER Filters the image using the color models.
22 - SubtractAbs	Vision Constant: VISION_OPERATION_SUBTRACTABS Returns absolute difference image of input image 1 and 2.
23 - Zoom	Vision Constant: VISION_OPERATION_ZOOM Enlarges or reduces the image.
24 - ColorStretch	Vision Constant: VISION_OPERATION_COLORSTRETCH Stretches the color values in the image between MinRGB and MaxRGB.
25-Shift	Vision Constant: VISION_OPERATION_SHIFT
26- DetectFocus	Vision Constant: VISION_OPERATION_DETECTFOCUS

Default: 1 – Open

## Remarks

The Operation settings can be grouped as follows:

### Morphology

Open, Close, Erode, Dilate

The morphological operations use grayscale morphology. The Polarity property determines which shade to operate on: Dark or Light. For example, if you have dark objects on a light background, then you should set the Polarity property to 1–DarkOnLight. If you were to set Polarity to 2–LightOnDark for the same image, then executing Erode will look like a Dilate, because the light objects will be eroded, making the dark objects dilated. The Iterations property determines how many times to execute the operation.

### Convolution

Smooth, Sharpen1, Sharpen2, HorizEdge, VertEdge, EdgeDetect1 EdgeDetect2, LaPlaceEdge1, LaPlaceEdge2, Thin, Thicken

The Polarity property determines which shade to operate on for the Thin and Thicken operations. The Iterations property determines how many times to execute the operation.

### Image Manipulation

Rotate, FlipHoriz, FlipVert, FlipBoth

### Binarize

ThresholdLow and ThresholdHigh are the boundaries for determining which gray values will be black and which values will be white. All gray values in between the thresholds will be black and all others will be white.

### Pixel to Pixel Operation

SubtractAbs

Calculates the difference image (absolute value) of the image buffers set to the ImageBuffer1 property and the ImageBuffer2 property.

### Color Filter

One or more filter colors and a background color can be taught. At runtime, the ImageOp tool checks each pixel color in the image ROI. If a pixel color is within the specified tolerance of one of the filter colors, then the pixel is unchanged. Otherwise, the pixel color is set to the specified background color.

### Color Stretch

This operation changes the color values in an image by mapping the RGB values between MinRGB and MaxRGB to 0 – 255. The KeepRGBRatio property also affects how the values are stretched.

### See Also

ImageOp Object, Iterations Property, MinRGB Property, MaxRGB Property, KeepRGBRatio Property

# Orientation Property

## Applies To

Vision Object: CodeReader

## Description

Sets / returns the orientation of the bar code.

## Usage

**VGet** *Sequence.Object.Orientation*, *var*

**VSet** *Sequence.Object.Orientation*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Sets the expected orientation of the bar code by following values

1 – Vertical and horizontal (VISION\_ORIENT\_BOTH)

2 - Horizontal (VISION\_ORIENT\_HORIZ)

3 - Vertical (VISION\_ORIENT\_VERT)

## Remarks

Sets and returns the expected orientation of the bar code.

## See Also

CodeReader Object

## OriginAngleEnabled Property

### Applies To

Vision Objects: Frame

### Description

Unlike the two-point frame which rotates the frame based on the vector rotation between the OriginPoint property and the YaxisPoint property, the OriginAngleEnabled property enables a single point frame to rotate based on the angle of the origin object.

### Usage

**VGet** *Sequence.Object.OriginAngleEnabled*, *var*

**VSet** *Sequence.Object.OriginAngleEnabled*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

0 – False Frame is not influenced by the OriginPoint object angle

1 – True Frame rotates with the OriginPoint object angle

Default 0 – False

### Remarks

Setting the OriginAngleEnabled property to True enable a frame to rotate by the angle of the origin object. For example, you can set the OriginPoint to a Polar object and set OriginAngleEnabled to 1–True. The frame will rotate according to the angle of the Polar object.

If the YAxisObject is set to a value other than Screen, the YAxisObject setting is given preference.

### See Also

Frame Object, Object Tab

# OriginPntObjResult Property

## Applies To

Vision Objects: Frame

## Description

Specifies which result to use from the OriginPointObject.

## Usage

**VGet** *Sequence.Object.OriginPntObjResult*, *var*

**VSet** *Sequence.Object.OriginPntObjResult*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

The value can range from 1 to the NumberToFind value for the OriginPointObject.

If the OriginPointObject is "Screen", then the value is always "1".

## Remarks

Use the OriginPntObjResult property to specify a result number other than "1" for a Frame Object's OriginPoint.

## See Also

Frame Object, Object Tab, OriginPoint Property, YAxisPoint Property, YAxisObjResult Property

# OriginPoint Property

## Applies To

Vision Objects: Frame

## Description

Defines the vision object to be used as the origin point for a Frame object.

## Usage

**VGet** *Sequence.Object.OriginPoint*, *var*

**VSet** *Sequence.Object.OriginPoint*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property. Valid vision objects for the OriginPoint property are ArcFinder, ArcInspector, Blob, Correlation, DefectFinder, Edge, Geometric, LineInspector, Point and Polar objects. The OriginPoint may also be based on the Screen position of the Frame.

## Values

Screen or any object that runs prior to the frame and returns PixelX and PixelY results.

Default: Screen

## Remarks

When a Frame object is first drag-and-dropped onto the Image Display area of the Vision Guide Window, the default OriginPoint property is set to Screen. Frame objects are normally attached to other Vision objects. This is the purpose of the OriginPoint and YAxisPoint. Through these 2 properties the user can define a frame of reference for other objects to have their position based upon. This capability is useful when specific features can be used to find reference points on a part and then other vision objects can be located on the image with respect to the frame position defined.

The OriginPoint and YAxisPoint properties are used together to define a vision frame which has an origin at the OriginPoint and a Y Axis direction defined by the YAxisPoint property.

It is important to note that for each specific vision sequence, only those vision objects which are executed prior to the Frame object in the vision sequence steps will be available to use as an OriginPoint. (The order of the vision object execution can be adjusted from the Sequence Tab.)

When using the GUI to change the OriginPoint property Value, a drop down list will appear showing a list of available vision objects (along with the default value Screen) which can be used to define the Origin of the Frame. Click on one of the choices and the value field will be set accordingly.

When using the Object tab to set the OriginPoint property it is important to note that only those objects which are defined prior to the Frame object are displayed in the drop down list. This helps reduce the chances of the user defining an OriginPoint which isn't defined prior to the Frame object.

Vision Guide automatically checks which vision objects can be used as the OriginPoint and displays only those object Names in the drop down list.

## See Also

Frame Object, Object Tab, OriginPntObjResult Property, YAxisPoint Property

## Overlapped Result

### Applies To

Vision Object: Geometric

### Description

Returns whether found objects overlap each other or not.

### Usage

**VGet** *Sequence.Object.Overlapped, var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean integer to acquire the result value

### Values

True: Overlap

False: Not overlap

### Remarks

Returns whether found objects overlap each other or not. Overlapping will be found based on overlapping of model windows. When `NumberToFind` is set to 2 or larger and some work pieces are found, the result whether the work pieces overlap each other or not will be returned.

### See Also

Geometric Object, `NumberToFind` Property, `RejectOnEdge` Property

## PassColor Property

### Applies To

Vision Object: All vision objects

### Description

Sets / returns the color of an object when it is Passed.

### Usage

**VGet** *Sequence.Object.PassColor*, *var*

**VSet** *Sequence.Object.PassColor*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String or string expression for the new value of the property.

### Values

String representing the name of color

Default: "LightGreen"

### Remarks

Specifies the color for the object when the object result is Passed. To configure how an object is passed, set the PassType property.

### See Also

Found Result, Graphics Property, FailColor Property, LabelBackColor Property, PassType Property

## Passed Result

### Applies To

Vision Object

### Description

Returns whether the result of an object has passed.

### Usage

**VGet** *Sequence.Object.Passed*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean integer to acquire the result value

### Values

0 – False Object did not pass.

1 – True Object passed.

### Remarks

Returns whether the specified object passed. To specify how an object is considered passed, set the PassType property.

For example, if the PassType of the Blob object is set to “AllFound”, this result will return True when the number of objects specified in “NumberToFind” were all found.

### See Also

AllPassed Result, PassColor Property, PassType Property, Sequence Tab, Vision Sequence

# PassType Property

## Applies To

Vision Object: All vision objects

## Description

Sets / returns the condition that specifies how an object is considered to be passed or failed.

## Usage

**VGet** *Sequence.Object.PassType*, *var*

**VSet** *Sequence.Object.PassType*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

## Values

- 1- SomeFound Vision Constant: `VISION_PASSTYPE_SOMEFOUND`  
Sets the result as Passed if one or more objects were found.
- 2- AllFound Vision Constant: `VISION_PASSTYPE_ALLFOUND`  
Sets the result as Passed if the expected number of objects (`NumberToFind`) was found.
- 3- SomeNotFound Vision Constant: `VISION_PASSTYPE_SOMENOTFOUND`  
This sets instances where one or more of the expected number of objects (`NumberToFind`) are not found as a Pass.
- 4- AllNotFound Vision Constant: `VISION_PASSTYPE_ALLNOTFOUND`  
Sets the result as Passed if no objects were found.

Default: 1 – SomeFound

For DefectFinder, ArcInspector, LineInspector: 4 - AllNotFound

For all other objects: 1 - SomeFound

## Remarks

Sets / returns the condition that specifies how an object is considered to be passed or failed. If all object results in the vision sequence are Passed, the AllPassed result of the sequence will be True.

## See Also

AllPassed Result, Found Result, Graphics Property, FailColor Property, PassColor Property

# PDFScanInterval Property

Design time only

## Applies To

Vision Object: CodeReader

## Description

Sets the scanning line pitch for PDF417 codes.

## Remarks

Sets the scanning line pitch for PDF417 codes. Decreasing the value increases the detection accuracy, but detection time will be longer.

Values range from 1 to height/width of the PDF417 code to be scanned.

Default: 10

## See Also

CodeReader Object

## Perimeter Result

### Applies To

Vision Objects: Blob, BoxFinder, DefectFinder

### Description

Returns the perimeter of a blob in pixels.

### Usage

**VGet** *Sequence.Object.Perimeter* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

### Values

Real number in pixels.

### Remarks

Returns the perimeter of the found blob in pixels. (including the edges of holes)

### See Also

Blob Object, Compactness Result, DefectFinder Object, BoxFinder Object, Holes Result, Object Tab, Roughness Result

# PixelLength Result

## Applies To

Vision Objects: ArcInspector, Line, LineInspector

## Description

Returns the length in pixels of the distance between the starting and ending point of the line for Line objects, and the length of a defect for ArcInspector and LineInspector objects.

## Usage

**VGet** *Sequence.Object.PixelLength*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in pixels

## Remarks

Unlike the Length result, the PixelLength result returns a value even if calibration has not yet been performed. This is because the units are in pixels and no calibration is needed for pixel unit based calculations. If the user needs a length in millimeters then perform a standalone or robot based camera calibration and use the Length result.

## Statistics

For the PixelLength result, the following statistics are available.

PixelLengthMax, PixelLengthMean, PixelLengthMin, PixelLengthStdDev.

Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

ArcInspector Object, Length Result, Line Object, LineInspector Object, Object Tab

## PixelLine Result

Runtime only

### Applies To

Vision Objects: ArcInspector, Line, LineInspector

### Description

Run time only result which returns the pixel coordinate position data of the starting (X1, Y1) and ending (X2, Y2) points of the specified object.

### Usage

**VGet** *Sequence.Object.PixelLine, X1, Y1, X2, Y2*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*X1* Real variable representing the X coordinate position of the starting point of the Line object specified by *Object*.

*Y1* Real variable representing the Y coordinate position of the starting point of the Line object specified by *Object*.

*X2* Real variable representing the X coordinate position of the ending point of the Line object specified by *Object*.

*Y2* Real variable representing the Y coordinate position of the ending point of the Line object specified by *Object*.

### Values

*X1, X2* Real variable specified in pixels from 1 - video width.

*Y1, Y2* Real variable specified in pixels from 1 - video height.

### Remarks

The PixelLine result is a runtime result which provides the X1, Y1, X2 and Y2 pixel coordinate data for the starting and ending points of the specified Line object.

The PixelLine result returns the same information as the PixelX1, PixelY1, PixelX2, and PixelY2 results. However, it returns this information with 1 function call rather than 4 separate calls.

### See Also

ArcInspector Object, Line Object, LineInspector Object, Object Tab, PixelX1 Result, PixelX2 Result, PixelY1 Result, PixelY2 Result, RobotXYU Result, RobotU Result, RobotX Result, RobotY Result

## PixelMajorDiam Result

### Applies To

Vision Object: ArcFinder

### Description

Returns length of the major axis of the ellipse found by ArcFinder.

### Usage

**VGet** *Sequence.Object*. **PixelMajorDiam**, *var*

*Sequenc* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable containing a value of the result.

### Values

Returns length of the major axis of the found ellipse in pixels.

### Remarks

Returns length of the major axis of the ellipse found by ArcFinder in pixels. To obtain the value in millimeters, use the FoundMajorDiam result.

### See Also

ArcFinder Object, ArcSearchType Property, FoundMajorDiam Result, FoundMinorDiam Result, PixelMinorDiam Result, Object Tab

## PixelMinorDiam Result

### Applies To

Vision Object: ArcFinder

### Description

Returns length of the minor axis of the ellipse found by ArcFinder.

### Usage

**VGet** *Sequence.Object*. **PixelMinorDiam**, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable containing a value of the result.

### Values

Returns length of the minor axis of the found ellipse in pixels.

### Remarks

Returns length of the minor axis of the ellipse found by ArcFinder in pixels. To obtain the value in millimeters, use the FoundMinorDiam result.

### See Also

ArcFinder Object, ArcSearchType Property, FoundMajorDiam Result, FoundMinorDiam Result, PixelMajorDiam Result, Object Tab

# PixelRadius Result

## Applies To

Vision Object: ArcFinder

## Description

Returns the radius of the found circular object in pixels.

## Usage

**VGet** *Sequence.Object.PixelRadius*(*result*), *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable containing a value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number indicating the radius [Unit: pixels]

## Remarks

Returns the radius of the found circular object by a actual number. [Unit: pixels]

If the user needs the radius in millimeters, then use the FoundRadius result.

## See Also

ArcFinder Object, Object Tab, FoundRadius Result

# PixelX Result

## Applies To

Vision Objects: ArcInspector, Blob, BoxFinder, CodeReader, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineInspector, OCR, Point, Polar, Text

## Description

Returns the X position coordinate of the found part's position in pixel coordinates.

## Usage

**VGet** *Sequence.Object.PixelX* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

## Values

Min: 0

Max Video width- 1

## Remarks

The PixelX result is the X coordinate of the objects position in the image coordinate system. The value is a actual number that has a fractional component because of the sub-pixeling feature.

## Statistics

For the PixelX result, the following statistics are available.

PixelXMax, PixelXMean, PixelXMin, PixelXStdDev.

Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

Angle Result, Blob Object, CameraX Result, CameraXYU Result, Correlation Object, Edge Object, Found Result, Geometric Object, Point Object, Polar Object, PixelXYU Result, RobotX Result, RobotXYU Result, CodeReader Object, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

# PixelX1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

Line, LineFinder: Returns the pixel X coordinate of the starting point of a Line object.

BoxFinder: Returns the X (X1) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelX1*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Min: 0

Max Video width- 1

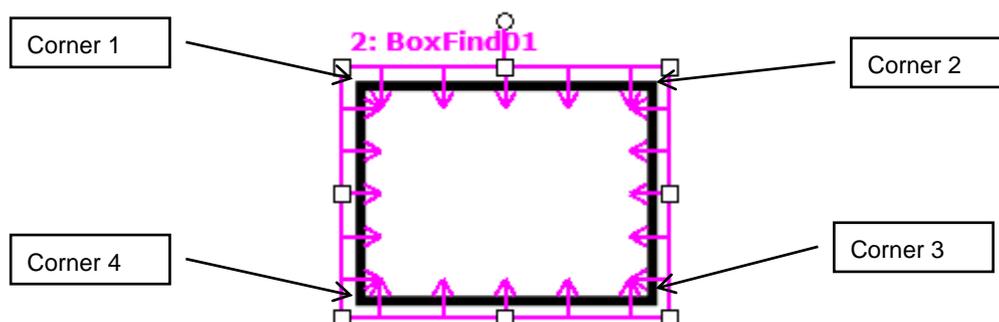
## Remarks

For Line, LineFinder

Every line has a starting point and ending point. The PixelX1 and PixelX2 results represent the X coordinate position of the starting (X1,Y1) and end points (X2,Y2) of the specified Line object. Since Line object starting and end points can be assigned to other vision objects, the (PixelX1, PixelY1) and (PixelX2, PixelY2) coordinate pairs can actually be pixel coordinate positions which match the PixelX and PixelY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (PixelX, PixelY) results from the Correlation object will match the (PixelX1, PixelY1) results for the Line object.)

For BoxFinder

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelX1 is used to retrieve the X coordinate for the Corner1 point shown in the diagram below.



See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX2 Result, PixelY Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelY3 Result, PixelX4 Result, PixelY4 Result, RobotX Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# PixelX2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

Line, LineFinder: Returns the pixel X coordinate of the end point of a Line object.

BoxFinder: Returns the X (X2) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelX2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Min: 0

Max Video width- 1

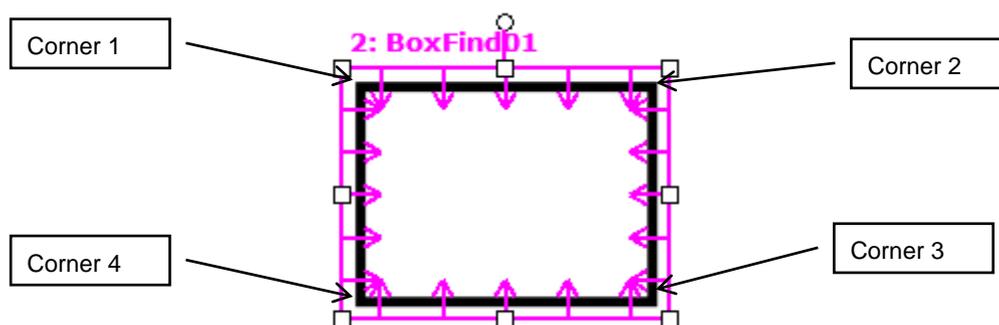
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The PixelX1 and PixelX2 results represent the X coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (PixelX1, PixelY1) and (PixelX2, PixelY2) coordinate pairs can actually be pixel coordinate positions which match the PixelX and PixelY results for other vision objects. (In other words if a Line object's endpoint is defined by a Correlation object, then the (PixelX, PixelY) results from the Correlation object will match the (PixelX2, PixelY2) results for the Line object.)

For BoxFinder

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelX2 is used to retrieve the X coordinate for the Corner2 point shown in the diagram below.



See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, PixelY Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelY3 Result, PixelX4 Result, PixelY4 Result, RobotX Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# PixelX3 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the X (X3) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelX3*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

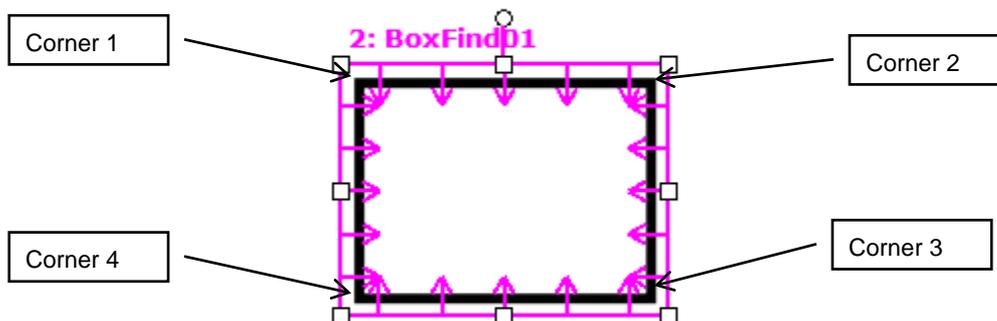
## Values

Min: 0

Max Video width – 1

## Remarks

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelX3 is used to retrieve the X coordinate for the Corner3 point shown in the diagram below.



## See Also

PixelX1 Result, PixelX2 Result, PixelY1 Result, PixelY2 Result, PixelY3 Result, PixelX4 Result, PixelY4 Result, BoxFinder Object, Object Tab

# PixelX4 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the X (X4) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelX4*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

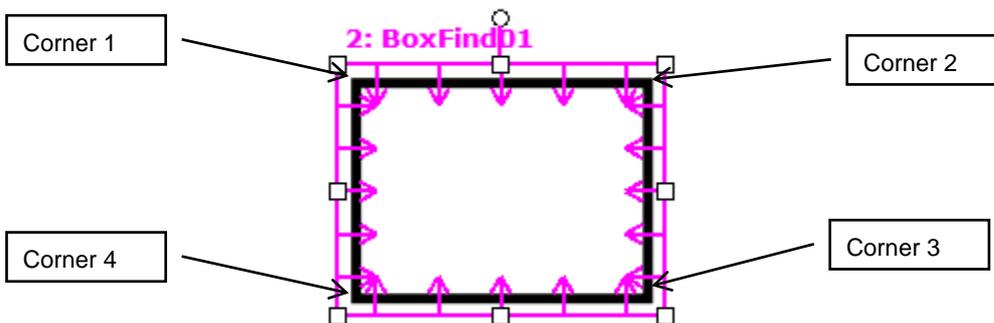
## Values

Min: 0

Max Video width – 1

## Remarks

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelX4 is used to retrieve the X coordinate for the Corner4 point shown in the diagram below.



## See Also

PixelX1 Result, PixelX2 Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelY3 Result, PixelY4 Result, BoxFinder Object, Object Tab

## PixelXYU Result

Runtime only

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Blob, CornerFinder, Correlation, DefectFinder, Edge, Geometric, Point, Polar, LineInspector

### Description

Returns the PixelX, PixelY and Angle coordinates of the found part's position in the image coordinate system.

### Usage

**VGet** *Sequence.Object.PixelXYU*[(*result*)], *found*, *xVar*, *yVar*, *uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the part you are looking for was found.

*xVar* Real variable representing the X pixel coordinate position of the part.

*yVar* Real variable representing the Y pixel coordinate position of the part.

*uVar* Real variable representing the angular position (rotation) of the part with respect to the image coordinate system

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

*found* Boolean value which is either True or False

*xVar* Real number in pixels

*yVar* Real number in pixels

*uVar* Real number in degrees

### Remarks

The PixelXYU result returns coordinates in the image coordinate system.

### See Also

Angle Result, ArcFinder Object, ArcInspector Object, Blob Object, CameraX Result, CameraY Result, CameraXYU Result, Correlation Object, Edge Object, Found Result, Geometric object, LineInspector Object, PointObject, Polar Object, BoxFinder Object, CornerFinder Object, RobotX Result, RobotY Result, RobotU Result, RobotXYU Result

# PixelY Result

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, Point, Polar, LineInspector, CodeReader, OCR, Text

## Description

Returns the Y position coordinate of the found part's position in pixel coordinates.

## Usage

**VGet** *Sequence.Object.PixelY* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Min: 0

Max: Video height- 1

## Remarks

The PixelY result is the Y coordinate of the objects position in the image coordinate system. The value is a actual number that has a fractional component because of the sub-pixeling feature.

## Statistics

For the PixelY result, the following statistics are available. PixelYMax, PixelYMean, PixelYMin, PixelYStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

Angle Result, Blob Object, CameraXYU Result, CameraY Result, Correlation Object, Edge Object, Found Result, Geometric Object, Object Tab, PixelXYU Result, Point Object, Polar Object, RobotY Result, RobotXYU Result, CodeReader Object, OCR Object, BoxFinder Object Contour Object, CornerFinder Object, Text Object

# PixelY1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the pixel Y coordinate of the starting point of a Line object.

**BoxFinder:** Returns the Y (Y1) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelY1*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Min: 0

Max: ImageSize height- 1

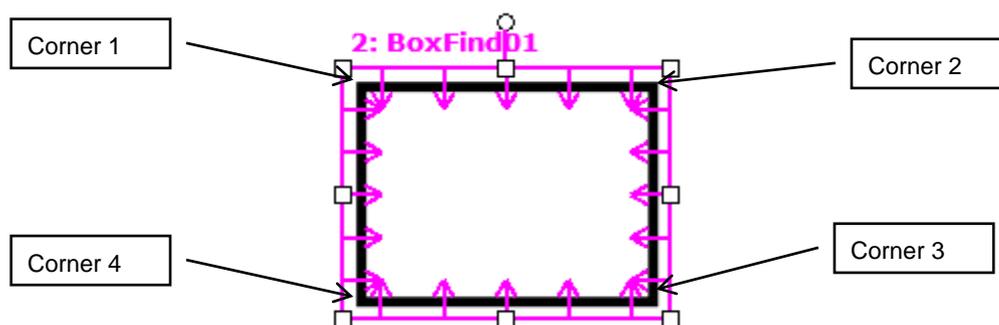
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The PixelY1 and PixelY2 results represent the Y coordinate position starting (Y1) and endpoints (Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (PixelX1, PixelY1) and (PixelX2, PixelY2) coordinate pairs can actually be pixel coordinate positions which match the PixelX and PixelY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (PixelX, PixelY) results from the Correlation object will match the (PixelX1, PixelY1) results for the Line object.)

For BoxFinder

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelY1 is used to retrieve the Y coordinate for the Corner1 point shown in the diagram below.



See Also

Angle Result, Line Object, Object Tab, PixelX Result, PixelX1 Result, PixelY Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelY3 Result, PixelX4 Result, PixelY4 Result, RobotY Result, RobotXYU Result, X1 Property, X2 Property, Y1 Property, Y2 Property

# PixelY2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the pixel Y coordinate of the end point of a Line object.

**BoxFinder:** Returns the Y (Y2) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelY2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Min: 0

Max: ImageSize height - 1

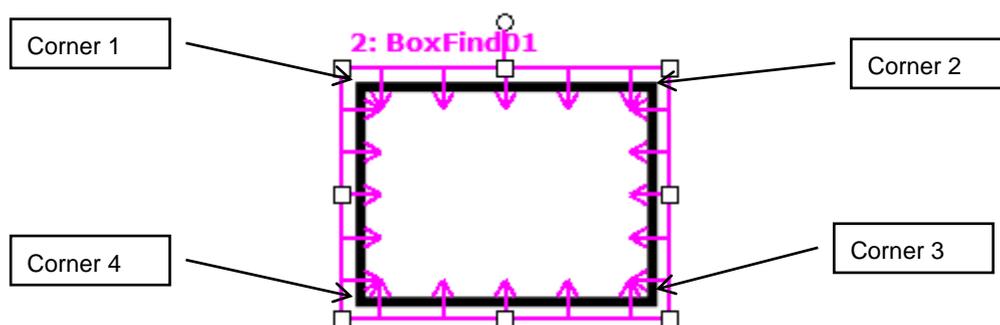
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The PixelY1 and PixelY2 results represent the Y coordinate position starting (Y1) and endpoints (Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (PixelX1, PixelY1) and (PixelX2, PixelY2) coordinate pairs can actually be pixel coordinate positions which match the PixelX and PixelY results for other vision objects. (In other words if a Line object's endpoint is defined by a Correlation object, then the (PixelX, PixelY) results from the Correlation object will match the (PixelX2, PixelY2) results for the Line object.)

For BoxFinder

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelY2 is used to retrieve the Y coordinate for the Corner2 point shown in the diagram below.



See Also

Angle Result, Line Object, Object Tab, PixelX Result, PixelX1 Result, PixelX2 Result, PixelX3 Result, PixelY3 Result, PixelX4 Result, PixelY4 Result, PixelY Result, PixelY1 Result, RobotXYU Result, RobotY Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# PixelY3 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y3) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelY3*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

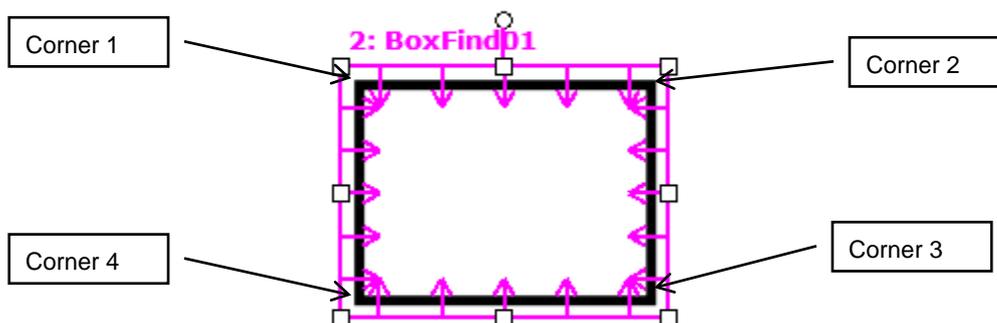
## Values

Min: 0

Max Video width – 1

## Remarks

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelY3 is used to retrieve the Y coordinate for the Corner3 point shown in the diagram below.



## See Also

PixelX1 Result, PixelX2 Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelX4 Result, PixelY4 Result, BoxFinder Object, Object Tab

# PixelY4 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y4) coordinate position for corner points of rectangles detected in a pixel coordinate system.

## Usage

**VGet** *Sequence.Object.PixelY4*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

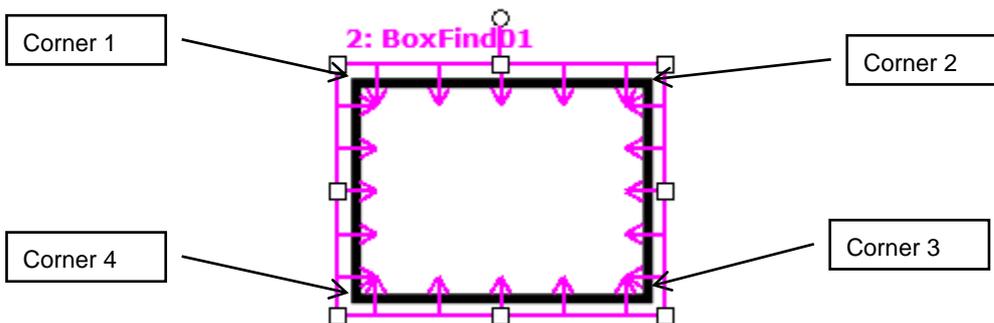
## Values

Min: 0

Max Video width – 1

## Remarks

The pixel coordinates for the four corners of a rectangle can be retrieved as Pixel X1, 2, 3, 4 results and Pixel Y1, 2, 3, 4 results. PixelY4 is used to retrieve the Y coordinate for the Corner4 point shown in the diagram below.



## See Also

PixelX1 Result, PixelX2 Result, PixelY1 Result, PixelY2 Result, PixelX3 Result, PixelY3 Result, PixelX4 Result, BoxFinder Object, Object Tab

# PointsTaught Property

## Applies To

Vision Calibration

## Description

Returns the teach state of a vision calibration's points.

## Usage

**VGet** *Calibration.PointsTaught*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Boolean variable that will contain the value of the result.

## Values

0 – False Points have not been taught.

1 – True Points have been taught.

## Remarks

PointsTaught must be True before you can execute a calibration. If you teach the calibration points from the Vision Guide GUI Calibration tab, then this property will automatically be set to 1–True.

## See Also

CalComplete Result, ShowConfirmation Property

# PointType Property

## Applies To

Vision Objects: Point

## Description

Sets / returns the type of a point.

## Usage

**VGet** *Sequence.Object.PointType*, *var*

**VSet** *Sequence.Object.PointType*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

- 0 - Screen Sets the Point object's position to be based upon the position on the screen
- 1 - MidPoint Sets the Point object's position to be based upon the Midpoint of the Line object selected as LineObject1 for this Point.
- 2 - Intersection Sets the Point objects position to be based upon the point were 2 lines intersect as defined by LineObject1 and LineObject2 for this point.

Default: 0 - Screen

## Remarks

Point objects are useful to define the midpoint or intersection point of a line or lines. This is their primary purpose. The PointType property is used to define the what the position for a Point object will be based upon. As mentioned before there are 3 choices.

0 - Screen: This is the default value for a Point object when it is first created. This type is useful when you want to specify a static point in the image or in a frame.

1 - MidPoint: A point position can be set to coincide with the midpoint of the line defined by the LineObject1 property. If the LineObject1 property does not specify a line, then an error will occur if you try to set the PointType to 1–MidPoint informing you that LineObject1 does not exist. (i.e. you cannot define a point as the midpoint of a non-existent line.)

2 - Intersection: A point position can be set to coincide with the intersection of 2 lines defined by the LineObject1 and LineObject2 Properties. If the either the LineObject1 or LineObject2 property does not specify a Line then an error dialog will appear if you try to set the PointType to 2–Intersection informing you that one of the 2 lines required to form an intersection does not exist.

The intersection of 2 lines does not have to appear directly between the starting and ending points for the lines. The intersection could occur somewhere along the imaginary extension of either or both lines.

## See Also

LineObject1 Property, LineObject2 Property, Object Tab, Point Object

# Polarity Property

## Applies To

Vision Objects: ArcInspector, ArcFinder, Blob, BoxFinder, CodeReader, Contour, CornerFinder, DefectFinder, Edge, ImageOp, LineFinder, LineInspector, OCR

## Description

For Blob, ImageOp, OCR objects, Polarity defines the differentiation between objects and background.

The Polarity property defines the edge direction for Edge, LineFinder, ArcFinder, LineInspector, ArcInspector, BoxFinder and CornerFinder objects.

When using DefectFinder, the Polarity property defines the polarity of defects detected.

## Usage

**VGet** *Sequence.Object.Polarity*, *var*

**VSet** *Sequence.Object.Polarity*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Edge:	1 - LightToDark	Vision Constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision Constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light
Blob:	1 - DarkOnLight	Vision Constant: VISION_POLARITY_DARK Find a dark blob on a light background
	2 - LightOnDark	Vision Constant: VISION_POLARITY_LIGHT Find a light blob on a dark background
ImageOp:	1 - DarkOnLight	Vision Constant: VISION_POLARITY_DARK The operation will be performed on dark objects.
	2 - LightOnDark	Vision Constant: VISION_POLARITY_LIGHT The operation will be performed on light objects.
CodeReader:	1 - DarkOnLight	Vision Constant: VISION_POLARITY_DARK Find a dark code on a light background
	2 - LightOnDark	Vision Constant: VISION_POLARITY_LIGHT Find a light code on a dark background
LineFinder:	1 - LightToDark	Vision Constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision Constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light
ArcFinder:	1 - LightToDark	Vision Constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision Constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light

LineInspector:	1 - LightToDark	Vision Constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision Constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light
ArcInspector:	1 - LightToDark	Vision Constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision Constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light
OCR:	1 - DarkOnLight	Vision Constant: VISION_POLARITY_DARK Find dark characters on a light background
	2 - LightOnDark	Vision Constant: VISION_POLARITY_LIGHT Find light characters on a dark background
BoxFinder:	1 - LightToDark	Vision constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light
CornerFinder:	1 - LightToDark	Vision constant: VISION_POLARITY_DARK Searching for an Edge transition from light to dark
	2 - DarkToLight	Vision constant: VISION_POLARITY_LIGHT Searching for an Edge transition from dark to light

The default setting for the abovementioned objects: 1

DefectFinder:	1 - DarkOnLight	Vision constant: VISION_POLARITY_DARK Detects a dark defect on a light background.
	2 - LightOnDark	Vision constant: VISION_POLARITY_LIGHT Detects a light defect on a dark background.
	3 - Both	Vision constant: VISION_POLARITY_BOTH Detects both dark defects on light backgrounds, and light defects on dark backgrounds.

Default setting for DefectFinder: 3

### Remarks

The Polarity property is important for both the Edge and Blob objects because it defines one of the core parameters for each.

In the case of the Edge object, the Polarity defines the edge transition along the direction of the edge search.

When using the Blob object, Polarity is critical. The Vision System must be told whether or not to look for light objects on a dark background or dark objects on a light background. Without the proper setting for the Polarity property, the Blob object will return strange results. Keep in mind that if a Blob object can find a dark object on a light background it can also find a light object on a dark background. The ThresholdHigh property and ThresholdLow property will also have an impact on the Blob object 's ability to find blobs. Please refer to *ThresholdHigh Property* and *ThresholdLow Property* for more information.

**See Also**

Blob Object, CodeReader Object, Contour Object, DefectFinder Object, Direction Property, Edge Object, ImageOp Object, LineFinder Object, ArcFinder Object, LineInspector Object, ArcInspector Object, OCR Object, Object Tab, BoxFinder Object, CornerFinder Object, ThresholdLow Property, ThresholdHigh Property

## QRLargeSize Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Set this property to True when the QR code is large in the field of view.

### Remarks

Setting this property to True when the QR code is large for the field of view may increase search speed.

Setting range: True / False

Default: False

### See Also

CodeReader Object, QRMinContrast Property, QRMinLength Property, QRNarrowQuietZone Property, QROutputID Property

# QRMinContrast Property

Design time only

## Applies To

Vision Object: CodeReader

## Description

Sets the minimum contrast for scanning QR codes.

## Remarks

Reducing the value enables the detection of low-contrast codes. However, it increases the chance for false detection and takes longer.

Setting range: 30 – 255

Default: 64

## See Also

CodeReader Object, QRLargeSize Property, QRMinLength Property, QRNarrowQuietZone Property, QROutputID Property

## QRMinLength Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets the minimum QR code size.

### Remarks

Sets the minimum QR code size. Reducing the value may enable detection of small QR codes. However, it may increase the search time.

Setting range: 36 - 999 pixels

Default: 46

### See Also

CodeReader Object, QRLargeSize Property, QRMinContrast Property, QRNarrowQuietZone Property, QROutputID Property

## QRNarrowQuietZone Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets the quiet zone width (Normal or Narrow) of QR codes.

### Remarks

Setting this value to True sets the quiet zone narrow and setting it to False sets the quiet zone to normal.

Setting range: True / False

Default: False

### See Also

CodeReader Object, QRLargeSize Property, QRMinContrast Property, QRMinLength Property, QRNarrowQuietZone Property

## QROutputID Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the data carrier identifier in the QR code Text result.

### Remarks

For this version, the data carrier identifier is fixed at “JQ0”.

Setting range: True / False

Default: False

### See Also

CodeReader Object, QRLargeSize Property, QRMinContrast Property, QRMinLength Property, QRNarrowQuietZone Property, Text Result

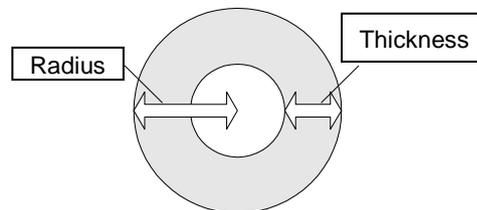
# Radius Property

## Applies To

Vision Objects: ArcInspector, ColorMatch, Polar

## Description

Defines the radius for an object. The diagram below shows a Polar object.



## Usage

**VGet** *Sequence.Object.Radius*, *var*

**VSet** *Sequence.Object.Radius*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*value* Integer expression for the new value of the property.

*var* Integer variable that will contain the value of the property.

## Values

The radius in pixels

Default: 50

## Remarks

Use the Radius property to set the radius for the object.

It is important to remember that the Polar object is used to process images that are circular in nature. The Radius property defines the size of the circle used for the Polar object. This means that the Radius property along with the Thickness property defines the size of the Search Window for the Polar object.

The size required for a Polar object Search very much depends upon what the Polar object is being used for. For example, if the Polar object is being used to Inspect Gear Teeth then the Polar object should be made just a little larger than the Gear to be inspected. However, if the Polar object is used just to find an angular position of a specific part of an image, then the Polar object may be smaller in size. Keep in mind that the smaller the Polar object search window, the faster the execution time for the Polar search.

## See Also

CenterPoint Property, CenterX Property, CenterY Property, ColorMatch Object, Object Tab, Polar Object, Thickness Property

## RadiusInner Property

### Applies To

Vision Object: ArcFinder, ArcInspector, Contour

### Description

Sets and returns the inner diameter of the detection area.

### Usage

**VGet** *Sequence.Object*.**RadiusInner**, *var*

**VSet** *Sequence.Object*.**RadiusInner**, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

### Values

The value is in pixels and must be less than or equal to RadiusOuter - 5.

Default: 50

### Remarks

Sets and returns the inner diameter of the search area. RadiusInner can also be changed by dragging the search area using a mouse in the Vision Guide GUI. In this case, the property value will be updated automatically.

### See Also

ArcFinder Object, ArcInspector Object, Contour Object, RadiusOuter Property, Direction Property

# RadiusOuter Property

## Applies To

Vision Object: ArcFinder, ArcInspector Contour

## Description

Sets and returns the outer diameter of the detection area.

## Usage

**VGet** *Sequence.Object*.**RadiusOuter**, *var*

**VSet** *Sequence.Object*.**RadiusOuter**, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

## Values

The value is in pixels and must be greater than or equal to RadiusInner + 5.

Default: 100

## Remarks

Sets and returns the outer diameter of the search area. RadiusOuter can also be changed by dragging the search area using a mouse in the Vision Guide GUI. In this case, the property value will be updated automatically.

## See Also

ArcFinder Object, ArcInspector Object, Contour Object, RadiusInner Property, Direction Property

## RejectOnEdge Property

### Applies To

Vision Objects: Blob, Contour, Correlation, DefectFinder, Geometric

### Description

Determines if an object is rejected if found on the edge of the search window.

### Usage

**VGet** *Sequence.Object.RejectOnEdge*, *var*

**VSet** *Sequence.Object.RejectOnEdge*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not reject the object if found on edge of search window.

1 – True Reject the object if found on edge of search window.

Default: 0 – False

### Remarks

When searching for objects that can fall outside the search window, setting RejectOnEdge to 1–True will prevent these objects from being found. For example, if you are trying to locate the center of a blob, and it falls partially outside the search window will not report the correct center of mass. Therefore, you should use RejectOnEdge to reject the result.

### See Also

Blob Object, Contour Object, Correlation Object, FoundOnEdge Result, Geometric Object, Object Tab

# ReferenceType Property

## Applies To

Vision Calibration

## Description

Sets / returns the reference type for a calibration.

## Usage

**VGet** *Calibration.ReferenceType, var*

**VSet** *Calibration.ReferenceType, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 – TaughtPoints            Vision Constant: VISION\_REFTYPE\_TAUGHTPOINTS  
Taught points

2 – UpwardCamera        Vision Constant: VISION\_REFTYPE\_UPWARDCAMERA  
Upward camera

Default: 1 – TaughtPoints

## Remarks

The calibration reference is a point defined in the robot coordinate system. When taught points are specified, one or two points are taught using a tool on the robot end effector during the teaching process for calibration points. When upward camera is specified, an upward camera that has already been calibrated is used to find the reference target. This method is the most accurate.

## See Also

CameraOrientation Property, PointsTaught Result, TwoPointReference Property

## ResultObject Property

### Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Specifies the vision object including the result you wish to render as a character string.

### Usage

**VGet** *Sequence.Object.ResultObject, var*

**VSet** *Sequence.Object.ResultObject, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

The name of the vision object, or “None”

Default: None

### Remarks

Specify the vision object name in the ResultObject property. Specify the vision object containing the result you wish to render as a character string. The vision object specified must be executed before the Text object.

### See Also

Text Object, ResultText1-3 Property, ShowLabel Property

# ResultText1 Property

## Applies To

Vision Object: Text

## Description

Specifies the result you wish to render as a character string.

## Usage

**VGet** *Sequence.Object.ResultText1, var*

**VSet** *Sequence.Object.ResultText1, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – None	Vision constant: VISION_TEXTRESULT_NONE Do not render result.
1 – Angle	Vision constant: VISION_TEXTRESULT_AGNLE Renders Angle result values.
2 – Angle1	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
3 – Angle2	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
4 – Area	Vision constant: VISION_TEXTRESULT_AREA Renders Area result values.
5 – CameraXY	Vision constant: VISION_TEXTRESULT_CAMERAXY Renders CameraX result and Camera Y result values.
6 – CameraXY1	Vision constant: VISION_TEXTRESULT_CAMERAXY1 Renders CameraX1 result and Camera Y1 result values.
7– CameraXY2	Vision constant: VISION_TEXTRESULT_CAMERAXY2 Renders CameraX2 result and Camera Y2 result values.
8– CameraXY3	Vision constant: VISION_TEXTRESULT_CAMERAXY3 Renders CameraX3 result and Camera Y3 result values.
9– CameraXY4	Vision constant: VISION_TEXTRESULT_CAMERAXY4 Renders CameraX4 result and Camera Y4 result values.
10 – ColorName	Vision constant: VISION_TEXTRESULT_COLORNAME Renders ColorName result values.
11 – ColorValue	Vision constant: VISION_TEXTRESULT_COLORVALUE Renders ColorValue result values.
12 – Compactness	Vision constant: VISION_TEXTRESULT_COMPACTNESS Renders Compactness result values.
13 – Contrast	Vision constant: VISION_TEXTRESULT_CONTRAST Renders Contrast result values.

14 – DefectLevel	Vision constant: VISION_TEXTRESULT_DEFECTLEVEL Renders DefectLevel result values.
15 – FitError	Vision constant: VISION_TEXTRESULT_FITERROR Renders FitError result values.
16 – FocusValue	FocusValue constant: VISION_TEXTRESULT_FOCUSVALUE Renders FocusValue alue result values.
17 – Holes	Vision constant: VISION_TEXTRESULT_HOLES Renders Holes result values.
18 – Length	Vision constant: VISION_TEXTRESULT_LENGTH Renders Length result values.
19 – MaxError	Vision constant: VISION_TEXTRESULT_MAXERROR Renders MaxError result values.
20 – MaxX	Vision constant: VISION_TEXTRESULT_MAXX Renders MaxX result values.
21 – MaxY	Vision constant: VISION_TEXTRESULT_MAXY Renders MaxY result values.
22 – MinX	Vision constant: VISION_TEXTRESULT_MINX Renders MinX result values.
23 – MinY	Vision constant: VISION_TEXTRESULT_MINY Renders MinY result values.
24 – Passed	Vision constant: VISION_TEXTRESULT_PASSED Renders Passed result values.
25 – Perimeter	Vision constant: VISION_TEXTRESULT_PERIMETER Renders Perimeter result values.
26 – PixelLength	Vision constant: VISION_TEXTRESULT_PIXELLENGTH Renders PixelLength result values.
27 – PixelRadius	Vision constant: VISION_TEXTRESULT_PIXELRADIUS Renders PixelRadius result values.
28 – PixelXY	Vision constant: VISION_TEXTRESULT_PIXELXY Renders PixelX result and PixelY result values.
29 – PixelXY1	Vision constant: VISION_TEXTRESULT_PIXELXY1 Renders PixelX1 result and PixelY1 result values.
30 – PixelXY2	Vision constant: VISION_TEXTRESULT_PIXELXY2 Renders PixelX2 result and PixelY2 result values.
31 – PixelXY3	Vision constant: VISION_TEXTRESULT_PIXELXY3 Renders PixelX3 result and PixelY3 result values.
32 – PixelXY4	Vision constant: VISION_TEXTRESULT_PIXELXY4 Renders PixelX4 result and PixelY4 result values.
33 – RobotXY	Vision constant: VISION_TEXTRESULT_ROBOTXY Renders RobotX result and RobotY result values.
34 – RobotXY1	Vision constant: VISION_TEXTRESULT_ROBOTXY1 Renders RobotX1 result and RobotY1 result values.
35 – RobotXY2	Vision constant: VISION_TEXTRESULT_ROBOTXY2 Renders RobotX2 result and RobotY2 result values.
36 – RobotXY3	Vision constant: VISION_TEXTRESULT_ROBOTXY3 Renders RobotX3 result and RobotY3 result values.

37 – RobotXY4	Vision constant: VISION_TEXTRESULT_ROBOTXY4 Renders RobotX4 result and RobotY4 result values.
38 – Roughness	Vision constant: VISION_TEXTRESULT_ROUGHNESS Renders Roughness result values.
39 – Scale	Vision constant: VISION_TEXTRESULT_SCALE Renders Scale result values.
40 – Score	Vision constant: VISION_TEXTRESULT_SCORE Renders Score result values.
41 – Strength	Vision constant: VISION_TEXTRESULT_STRENGTH Renders Strength result values.
42 – Text	Vision constant: VISION_TEXTRESULT_TEXT Renders Text result values.

**Remarks**

Specify the type of result rendered as a character string in the TextView object. The results available for selection will vary depending on the vision object type specified in the TextObj property.

**See Also**

Text Object, ResultObject Property, ShowLabel Property

# ResultText2 Property

## Applies To

Vision Object: Text

## Description

Specifies the result you wish to render as a character string.

## Usage

**VGet** *Sequence.Object.ResultText2, var*

**VSet** *Sequence.Object.ResultText2, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Renders Angle result values.

0 – None	Vision constant: VISION_TEXTRESULT_NONE Do not render result.
1 – Angle	Vision constant: VISION_TEXTRESULT_AGNLE Renders Angle result values.
2 – Angle1	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
3 – Angle2	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
4 – Area	Vision constant: VISION_TEXTRESULT_AREA Renders Area result values.
5 – CameraXY	Vision constant: VISION_TEXTRESULT_CAMERAXY Renders CameraX result and Camera Y result values.
6 – CameraXY1	Vision constant: VISION_TEXTRESULT_CAMERAXY1 Renders CameraX1 result and Camera Y1 result values.
7– CameraXY2	Vision constant: VISION_TEXTRESULT_CAMERAXY2 Renders CameraX2 result and Camera Y2 result values.
8– CameraXY3	Vision constant: VISION_TEXTRESULT_CAMERAXY3 Renders CameraX3 result and Camera Y3 result values.
9– CameraXY4	Vision constant: VISION_TEXTRESULT_CAMERAXY4 Renders CameraX4 result and Camera Y4 result values.
10 – ColorName	Vision constant: VISION_TEXTRESULT_COLORNAME Renders ColorName result values.
11 – ColorValue	Vision constant: VISION_TEXTRESULT_COLORVALUE Renders ColorValue result values.
12 – Compactness	Vision constant: VISION_TEXTRESULT_COMPACTNESS Renders Compactness result values.

13 – Contrast	Vision constant: VISION_TEXTRESULT_CONTRAST Renders Contrast result values.
14 – DefectLevel	Vision constant: VISION_TEXTRESULT_DEFECTLEVEL Renders DefectLevel result values.
15 – FitError	Vision constant: VISION_TEXTRESULT_FITERROR Renders FitError result values.
16 – FocusValue	FocusValue constant: VISION_TEXTRESULT_FOCUSVALUE Renders FocusValue alue result values.
17 – Holes	Vision constant: VISION_TEXTRESULT_HOLES Renders Holes result values.
18 – Length	Vision constant: VISION_TEXTRESULT_LENGTH Renders Length result values.
19 – MaxError	Vision constant: VISION_TEXTRESULT_MAXERROR Renders MaxError result values.
20 – MaxX	Vision constant: VISION_TEXTRESULT_MAXX Renders MaxX result values.
21 – MaxY	Vision constant: VISION_TEXTRESULT_MAXY Renders MaxY result values.
22 – MinX	Vision constant: VISION_TEXTRESULT_MINX Renders MinX result values.
23 – MinY	Vision constant: VISION_TEXTRESULT_MINY Renders MinY result values.
24 – Passed	Vision constant: VISION_TEXTRESULT_PASSED Renders Passed result values.
25 – Perimeter	Vision constant: VISION_TEXTRESULT_PERIMETER Renders Perimeter result values.
26 – PixelLength	Vision constant: VISION_TEXTRESULT_PIXELLENGTH Renders PixelLength result values.
27 – PixelRadius	Vision constant: VISION_TEXTRESULT_PIXELRADIUS Renders PixelRadius result values.
28 – PixelXY	Vision constant: VISION_TEXTRESULT_PIXELXY Renders PixelX result and PixelY result values.
29 – PixelXY1	Vision constant: VISION_TEXTRESULT_PIXELXY1 Renders PixelX1 result and PixelY1 result values.
30 – PixelXY2	Vision constant: VISION_TEXTRESULT_PIXELXY2 Renders PixelX2 result and PixelY2 result values.
31 – PixelXY3	Vision constant: VISION_TEXTRESULT_PIXELXY3 Renders PixelX3 result and PixelY3 result values.
32 – PixelXY4	Vision constant: VISION_TEXTRESULT_PIXELXY4 Renders PixelX4 result and PixelY4 result values.
33 – RobotXY	Vision constant: VISION_TEXTRESULT_ROBOTXY Renders RobotX result and RobotY result values.
34 – RobotXY1	Vision constant: VISION_TEXTRESULT_ROBOTXY1 Renders RobotX1 result and RobotY1 result values.
35 – RobotXY2	Vision constant: VISION_TEXTRESULT_ROBOTXY2 Renders RobotX2 result and RobotY2 result values.

## ResultText2 Property

---

36 – RobotXY3	Vision constant: VISION_TEXTRESULT_ROBOTXY3 Renders RobotX3 result and RobotY3 result values.
37 – RobotXY4	Vision constant: VISION_TEXTRESULT_ROBOTXY4 Renders RobotX4 result and RobotY4 result values.
38 – Roughness	Vision constant: VISION_TEXTRESULT_ROUGHNESS Renders Roughness result values.
39 – Scale	Vision constant: VISION_TEXTRESULT_SCALE Renders Scale result values.
40 – Score	Vision constant: VISION_TEXTRESULT_SCORE Renders Score result values.
41 – Strength	Vision constant: VISION_TEXTRESULT_STRENGTH Renders Strength result values.
42 – Text	Vision constant: VISION_TEXTRESULT_TEXT Renders Text result values.

### Remarks

Specify the type of result rendered as a character string in the TextView object. The results available for selection will vary depending on the vision object type specified in the TextObj property.

### See Also

Text Object, ResultObject Property, ShowLabel Property

# ResultText3 Property

## Applies To

Vision Object: Text

## Description

Specifies the result you wish to render as a character string.

## Usage

**VGet** *Sequence.Object.ResultText3, var*

**VSet** *Sequence.Object.ResultText3, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Renders Angle result values.

0 – None	Vision constant: VISION_TEXTRESULT_NONE Do not render result.
1 – Angle	Vision constant: VISION_TEXTRESULT_AGNLE Renders Angle result values.
2 – Angle1	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
3 – Angle2	Vision constant: VISION_TEXTRESULT_ANGLE1 Renders Angle1 result values.
4 – Area	Vision constant: VISION_TEXTRESULT_AREA Renders Area result values.
5 – CameraXY	Vision constant: VISION_TEXTRESULT_CAMERAXY Renders CameraX result and Camera Y result values.
6 – CameraXY1	Vision constant: VISION_TEXTRESULT_CAMERAXY1 Renders CameraX1 result and Camera Y1 result values.
7– CameraXY2	Vision constant: VISION_TEXTRESULT_CAMERAXY2 Renders CameraX2 result and Camera Y2 result values.
8– CameraXY3	Vision constant: VISION_TEXTRESULT_CAMERAXY3 Renders CameraX3 result and Camera Y3 result values.
9– CameraXY4	Vision constant: VISION_TEXTRESULT_CAMERAXY4 Renders CameraX4 result and Camera Y4 result values.
10 – ColorName	Vision constant: VISION_TEXTRESULT_COLORNAME Renders ColorName result values.
11 – ColorValue	Vision constant: VISION_TEXTRESULT_COLORVALUE Renders ColorValue result values.
12 – Compactness	Vision constant: VISION_TEXTRESULT_COMPACTNESS Renders Compactness result values.

13 – Contrast	Vision constant: VISION_TEXTRESULT_CONTRAST Renders Contrast result values.
14 – DefectLevel	Vision constant: VISION_TEXTRESULT_DEFECTLEVEL Renders DefectLevel result values.
15 – FitError	Vision constant: VISION_TEXTRESULT_FITERROR Renders FitError result values.
16 – FocusValue	FocusValue constant: VISION_TEXTRESULT_FOCUSVALUE Renders FocusValue alue result values.
17 – Holes	Vision constant: VISION_TEXTRESULT_HOLES Renders Holes result values.
18 – Length	Vision constant: VISION_TEXTRESULT_LENGTH Renders Length result values.
19 – MaxError	Vision constant: VISION_TEXTRESULT_MAXERROR Renders MaxError result values.
20 – MaxX	Vision constant: VISION_TEXTRESULT_MAXX Renders MaxX result values.
21 – MaxY	Vision constant: VISION_TEXTRESULT_MAXY Renders MaxY result values.
22 – MinX	Vision constant: VISION_TEXTRESULT_MINX Renders MinX result values.
23 – MinY	Vision constant: VISION_TEXTRESULT_MINY Renders MinY result values.
24 – Passed	Vision constant: VISION_TEXTRESULT_PASSED Renders Passed result values.
25 – Perimeter	Vision constant: VISION_TEXTRESULT_PERIMETER Renders Perimeter result values.
26 – PixelLength	Vision constant: VISION_TEXTRESULT_PIXELLENGTH Renders PixelLength result values.
27 – PixelRadius	Vision constant: VISION_TEXTRESULT_PIXELRADIUS Renders PixelRadius result values.
28 – PixelXY	Vision constant: VISION_TEXTRESULT_PIXELXY Renders PixelX result and PixelY result values.
29 – PixelXY1	Vision constant: VISION_TEXTRESULT_PIXELXY1 Renders PixelX1 result and PixelY1 result values.
30 – PixelXY2	Vision constant: VISION_TEXTRESULT_PIXELXY2 Renders PixelX2 result and PixelY2 result values.
31 – PixelXY3	Vision constant: VISION_TEXTRESULT_PIXELXY3 Renders PixelX3 result and PixelY3 result values.
32 – PixelXY4	Vision constant: VISION_TEXTRESULT_PIXELXY4 Renders PixelX4 result and PixelY4 result values.
33 – RobotXY	Vision constant: VISION_TEXTRESULT_ROBOTXY Renders RobotX result and RobotY result values.
34 – RobotXY1	Vision constant: VISION_TEXTRESULT_ROBOTXY1 Renders RobotX1 result and RobotY1 result values.
35 – RobotXY2	Vision constant: VISION_TEXTRESULT_ROBOTXY2 Renders RobotX2 result and RobotY2 result values.

---

36 – RobotXY3	Vision constant: VISION_TEXTRESULT_ROBOTXY3 Renders RobotX3 result and RobotY3 result values.
37 – RobotXY4	Vision constant: VISION_TEXTRESULT_ROBOTXY4 Renders RobotX4 result and RobotY4 result values.
38 – Roughness	Vision constant: VISION_TEXTRESULT_ROUGHNESS Renders Roughness result values.
39 – Scale	Vision constant: VISION_TEXTRESULT_SCALE Renders Scale result values.
40 – Score	Vision constant: VISION_TEXTRESULT_SCORE Renders Score result values.
41 – Strength	Vision constant: VISION_TEXTRESULT_STRENGTH Renders Strength result values.
42 – Text	Vision constant: VISION_TEXTRESULT_TEXT Renders Text result values.

#### Remarks

Specify the type of result rendered as a character string in the TextView object. The results available for selection will vary depending on the vision object type specified in the TextObj property.

#### See Also

Text Object, ResultObject Property, ShowLabel Property

## Reversed Result

### Applies To

Vision Object: Geometric

### Description

Returns the polarity of the found object.

### Usage

**VGet** *Sequence.Reversed*[(*result*)], *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable representing the result value

*result* Optional result number. If omitted, the result number is the `CurrentResult`. Used for objects that return multiple results.

### Values

True : An object was found whose polarity is the reverse of the original model.

False : An object was found whose polarity is the same as the original model.

### Remarks

Returns the polarity of the found object. When the model is a black work piece on a white background, this result returns False if the same black work piece on a white background is found, and returns True if the white work piece on a black background is found.

If the `SearchPolarity` property is set to "1 - Find objects with same polarity or reverse polarity", this result will always return false.

### See Also

Geometric Object, SearchPolarity Property

# RobotAccel Property

## Applies To

Vision Calibration

## Description

Sets / returns the robot point to point motion acceleration used during the calibration.

## Usage

**VGet** *Calibration.RobotAccel*, *var*

**VSet** *Calibration.RobotAccel*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 1 to 99%.

Default: 10

## Remarks

Use RobotAccel along with RobotSpeed to configure the speed during a calibration. For more delicate systems, a slow speed and accel should be used. The robot must not cause any vibration of the camera that could affect calibration accuracy.

## See Also

Accel Statement, MotionDelay Property, RobotSpeed Property, Speed Statement

## RobotArm Property

### Applies To

Vision Calibration

### Description

Sets / returns the robot arm used when teaching points for a vision calibration.

### Usage

**VGet** *Calibration*.**RobotArm**, *var*

**VSet** *Calibration*.**RobotArm**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value from 0 - 15.

Default: 0

### Remarks

RobotArm defines the arm definition used during the teaching process for a vision calibration.

### See Also

RobotLocal, RobotTool

# RobotLimZ Property

## Applies To

Vision Calibration

## Description

Sets / returns the robot LimZ value used during the calibration cycle for a mobile camera.

## Usage

**VGet** *Calibration.RobotLimZ*, *var*

**VSet** *Calibration.RobotLimZ*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Real value from -999 mm to 999 mm.

Default: 0

## Remarks

Use RobotLimZ to specify the LimZ value used for the first motion used in a mobile camera calibration cycle (SCARA robot only). During mobile calibration, when the robot is moved to the first camera calibration point, a Jump command is used. RobotLimZ can be used to limit the distance that the robot moves up in Z for that Jump command.

## See Also

Accel Statement, RobotSpeed Property, Speed Statement

## RobotLocal Property

### Applies To

Vision Calibration

### Description

Sets / returns the local coordinate system used for a vision calibration.

### Usage

**VGet** *Calibration*.**RobotLocal**, *var*

**VSet** *Calibration*.**RobotLocal**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value from 0 - 15.

Default: 0

### Remarks

RobotLocal defines the local robot coordinate system used for a vision calibration. The RobotLocal is used during teach and also at runtime, where all robot coordinates are in the specified local coordinate system. The local coordinate system must be defined before it can be used by the calibration.

### See Also

Local Statement, RobotArm, RobotTool

# RobotNumber Property

## Applies To

Vision Calibration

## Description

Sets / returns the robot number associated with a vision calibration.

## Usage

**VGet** *Calibration.RobotNumber, var*

**VSet** *Calibration.RobotNumber, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 1 to the number of robots in the system.

## Remarks

RobotNumber specifies which robot the vision calibration is used for.

## See Also

RobotArm, RobotLocal, RobotTool

## RobotPlacePos Result

Runtime only

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, Correlation, DefectFinder Edge, Geometric, LineInspector Point, Polar

### Description

Returns a point that can be used to place a part after finding it with an upward camera.

### Usage

**VGet** *Sequence.Object*.**RobotPlacePos**[(*result*)], *found*, *placePoint*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the part was found.

*placePoint* Point variable that will contain the place position.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

*found* True or False

*placePoint* Point containing the coordinates needed to place the part.

### Remarks

The RobotPlacePos result returns a point for placing a part after it has been found with an upward camera.

Using RobotPlacePos eliminates the need to define a tool for the robot when placing parts found with an upward camera. Before you can use RobotPlacePos, you must calibrate it using the CalRobotPlacePos wizard from the Vision Guide GUI, or by setting CalRobotPlacePos at runtime.

Note: Tool and Arm settings at acquisition of this result and the settings at moving the robot to the acquired position must be the same.

### See Also

CalRobotPlacePos Property, RobotPlaceTargetPos Property, RobotPos Property

# RobotPlaceTargetPos Property

Runtime only

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, Correlation, DefectFinder Edge, Geometric, LineInspector Point, Polar

## Description

Sets / gets the target place position for a part.

## Usage

**VGet** *Sequence.Object.RobotPlaceTargetPos,targetPoint*

**VSet** *Sequence.Object.RobotPlaceTargetPos,targetPoint*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*targetPoint* Point variable that will contain the target place position.

## Values

*targetPoint* Point containing the coordinates of the part in the place position.

## Remarks

The RobotPlaceTargetPos property sets or gets the actual place position of a part. Normally, RobotPlaceTargetPos is not needed. The system automatically calculates it during calibration of RobotPlacePos. RobotPlaceTargetPos can be used to adjust the part place position after calibration, if necessary.

## See Also

CalRobotPlacePos Property, RobotPlacePos Result

## RobotPos Property

### Applies To

Vision Sequence

### Description

Sets robot position and orientation used in the vision sequence result.

### Usage

**VSet** *Sequence*.**RobotPos**, *position*

*Sequence* String variable containing a sequence name.

*position* Point variable that will contain the value of the property.

### Values

Point data representing the robot position and orientation at VRun execution.

### Remarks

The RobotPos property specifies the robot position and orientation at the time of image acquisition executed by VRun. Normally, this property does not need to be changed since the robot position and the orientation at VRun execution are the same as those at the time of image acquisition. However, when the vision system is used with the real time I/O, the robot position and orientation at VRun execution differ from those at the time of imaging. In this case, use this property to set actual robot position and orientation at the time of image acquisition.

### See Also

*EPSON RC+ 7.0 User's Guide Real Time I/O*

# RobotSpeed Property

## Applies To

Vision Calibration

## Description

Sets / returns the robot point to point motion speed used during the calibration.

## Usage

**VGet** *Calibration.RobotSpeed*, *var*

**VSet** *Calibration.RobotSpeed*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 1 to 100%.

Default: 10

## Remarks

Use RobotSpeed along with RobotAccel to configure the speed during a calibration. For more delicate systems, a slow speed and accel should be used. The robot must not cause any vibration of the camera that could affect calibration accuracy.

## See Also

Accel Statement, MotionDelay Property, RobotAccel Property, Speed Statement

## RobotTool Property

### Applies To

Vision Calibration

### Description

Sets / returns the robot tool used when teaching points for a vision calibration.

### Usage

**VGet** *Calibration*.**RobotTool**, *var*

**VSet** *Calibration*.**RobotTool**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value from 0 - 15.

Default: 0

### Remarks

RobotTool defines the tool used during the teaching process for a vision calibration. The tool must be defined before it can be used by the calibration.

### See Also

RobotArm, RobotLocal, TLSet Statement

# RobotToolXYU Result

Runtime only

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, Correlation, DefectFinder Edge, Geometric, LineInspector Point, Polar

## Description

Returns the tool X, tool Y and tool U values of the found part's position as a robot tool.

## Usage

**VGet** *Sequence.Object.RobotToolXYU[(result)],found, xVar, yVar, uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the part you are looking for was found.

*xVar* Real variable that will contain the X offset of the tool.

*yVar* Real variable that will contain the Y offset of the tool.

*uVar* Real variable that will contain the angular rotation of the tool.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

*found* True or False

*xVar* Real number in millimeters

*yVar* Real number in millimeters

*uVar* Real number in degrees

## Remarks

The RobotToolXYU result returns the tool offsets of the found part with an upward camera and therefore can be used to define a tool for robot guidance applications. The RobotToolXYU result *xVar* and *yVar* values are always returned in millimeters. The *uVar* value is always returned in degrees. When used for a Point object, *uVar* always returns 0.

It should be noted that the RobotToolXYU result can only be calculated for vision sequences which have been calibrated with the robot coordinate system with the CameraOrientation set to Fixed Upward. If an invalid calibration has been assigned to the vision sequence then the RobotToolXYU result cause an error to occur.

## Example

```
VGet Seq. Geom01. RobotToolXYU, f, tx, ty, tu

If f = True then
    Tlset 1, xY(tx,ty,u,tu)
    Tool1
EndIf

Jump Placepos,
```

## See Also

ArcFinder Object, ArcInspector Object, Blob Object, CameraX Result, CameraY Result, CameraXYU Result, Correlation Object, DefectFinder Object, Edge Object, Found Result, Geometric Object, LineInspector Object, PixelXYU Result, Point Object, Polar Object, RobotPos Property, RobotX Result, RobotY Result, RobotU Result

## RobotU Result

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, CornerFinder, Correlation, DefectFinder, Geometric, Line, LineFinder, LineInspector, Polar

### Description

Returns the U angle of the found part's position in the robot coordinate system.

### Usage

**VGet** *Sequence.Object*.**RobotU** [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real value representing in degrees (error if no calibration).

### Remarks

The RobotU result is similar to the Angle result except that the position results are returned with reference to the robot coordinate system. This means that the RobotU result is suited for robot guidance applications. However, keep in mind that a special result called the RobotXYU result is most often used for robot guidance because it returns not only the U, but also the X, and Y coordinate positions as well as whether or not the part was found. See *RobotXYU Result*.

It should be noted that the RobotU result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then the RobotU result will return 0.

You can add an offset to the RobotU result by using the RobotUOffset property.

### Statistics

For the RobotU Result, the following statistics are available. RobotUMax, RobotUMean, RobotUMin, RobotUStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

### See Also

Angle Result, Blob Object, CameraXYU Result, Correlation Object, Found Result, Geometric Object, Line Object, PixelXYU Result, Polar Object, RobotUOffset Property, RobotX Result, RobotY Result, RobotXYU Result, LineFinder Object, ArcFinder Object, DefectFinder Object, CodeReader Object, BoxFinder Object, CornerFinder Object

# RobotUOffset Property

## Applies To

Vision Calibration

## Description

Sets / returns the robot U axis offset which is added to RobotU object results.

## Usage

**VGet** *Calibration*.**RobotUOffset**, *var*

**VSet** *Calibration*. **RobotUOffset**, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real value from -999 degrees to 999 degrees.

Default: 0

## Remarks

Robot world calibrations can accurately return angle in the robot world (RobotU), but the end effector is normally not aligned exactly in the robot world, so a constant offset is necessary. RobotUOffset allows you to specify this constant offset. You can set the value directly, or you can use the RobotUOffset wizard from the RobotUOffset calibration property in the Vision Guide window.

## See Also

RobotU Result, RobotXYU Result

## RobotX Result

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineInspector, Point, Polar, OCR

### Description

Returns the X position coordinate of the found part's position in the robot coordinate system.

### Usage

**VGet** *Sequence.Object.RobotX* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

### Values

Real number in millimeters.

### Remarks

The RobotX result returns an X coordinate in the robot coordinate system, and can therefore be used for robot guidance applications. However, keep in mind that a special result called the RobotXYU result is most often used for robot guidance because it returns not only the X, but also the Y, and U coordinate positions as well as whether or not the part was found. See *RobotXYU Result*.

The RobotX Result is always returned in millimeters.

It should be noted that the RobotX result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotX result will cause an error to occur.

### Statistics

For the RobotX Result, the following statistics are available. RobotXMax, RobotXMean, RobotXMin, RobotXStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

### See Also

Angle Result, Blob Object, CameraXYU Result, Correlation Object, DefectFinder Object, Edge Object, Found Result, Geometric Object, PixelXYU Result, Point Object, Polar Object, RobotY Result, RobotU Result, RobotXYU Result, CodeReader Object, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object

# RobotX1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the X coordinate of the starting point position (X1) of a Line object in the robot coordinate system.

**BoxFinder:** Returns the X (X1) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object*.**RobotX1**[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

## Remarks

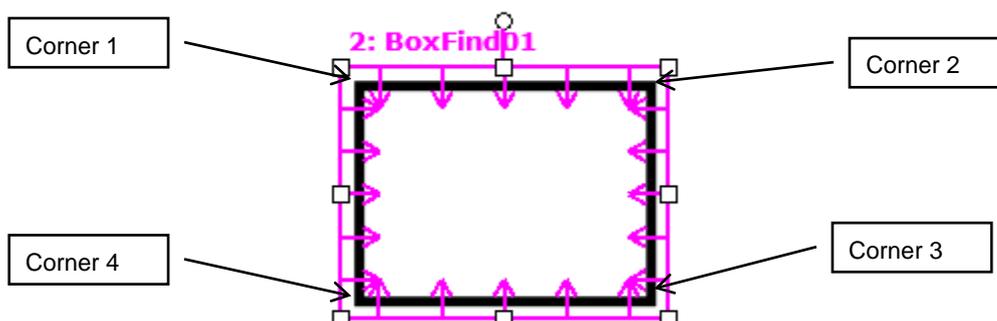
For Line, LineFinder

Every line must have a starting point and ending point. The RobotX1 and RobotX2 results represent the X coordinate position starting (X1,Y1) and endpoints (X2,Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (RobotX1, RobotY1) and (RobotX2, RobotY2) coordinate pairs can actually be Robot coordinate positions which match the RobotX and RobotY results for other vision objects. In other words if a Line object's starting point is defined by a Correlation object, then the (RobotX, RobotY) results from the Correlation object will match the (RobotX1, RobotY1) results for the Line object.

The RobotX1 result is always in millimeters in the robot coordinate system.

For BoxFinder

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotX1 is used to retrieve the X coordinate for the Corner1 point shown in the diagram below.



It should be noted that the RobotX1 result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotX1 result will cause an error to occur.

### See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, PixelX2 Result, PixelY Result, PixelY1 Result, PixelY2 Result, RobotX Result, RobotX2 Result, RobotXYU Result, RobotY Result, RobotY1 Result, RobotY2 Result, RobotX3 Result, RobotY3 Result, RobotX4 Result, RobotY4 Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# RobotX2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the X coordinate of the ending point position (X2) of a Line object in the robot coordinate system.

**BoxFinder:** Returns the X (X2) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotX2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

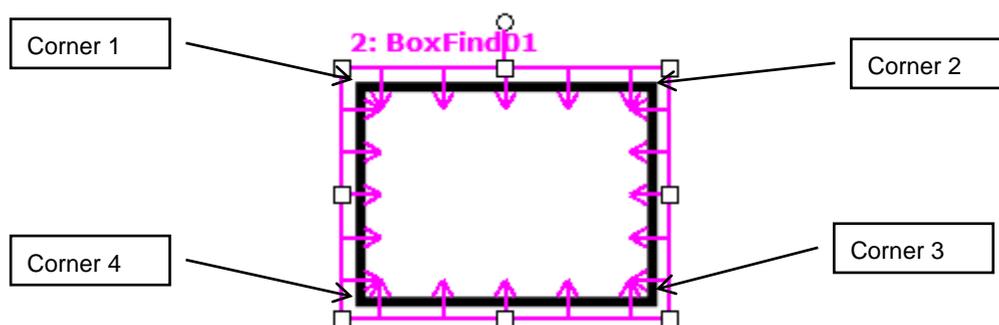
## Remarks

For Line, LineFinder

Every line has a starting point and ending point. The RobotX1 and RobotX2 results represent the X coordinates of the line starting point (X1, Y1) and end point (X2, Y2) of the specified Line object. Since Line object starting and end points can be assigned to other vision objects, the (RobotX1, RobotY1) and (RobotX2, RobotY2) coordinate pairs can actually be robot coordinate positions which match the RobotX and RobotY results for other vision objects. (In other words if a Line object's endpoint is defined by a Correlation object, then the (RobotX, RobotY) results from the Correlation object will match the (RobotX2, RobotY2) results for the Line object.)

For BoxFinder

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotX2 is used to retrieve the X coordinate for the Corner2 point shown in the diagram below.



The RobotX2 result is always in millimeters in the robot coordinate system.

It should be noted that the RobotX2 result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotX2 result will cause an error to occur.

### See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, PixelY Result, PixelY1 Result, PixelY2 Result, RobotX Result, RobotX1 Result, RobotXYU Result, RobotY Result, RobotY1 Result, RobotY2 Result, RobotX3 Result, RobotY3 Result, RobotX4 Result, RobotY4 Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# RobotX3 Result

## Applies To

Vision Object: BoxFinder

## Description

Returns the X (X3) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotX3*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

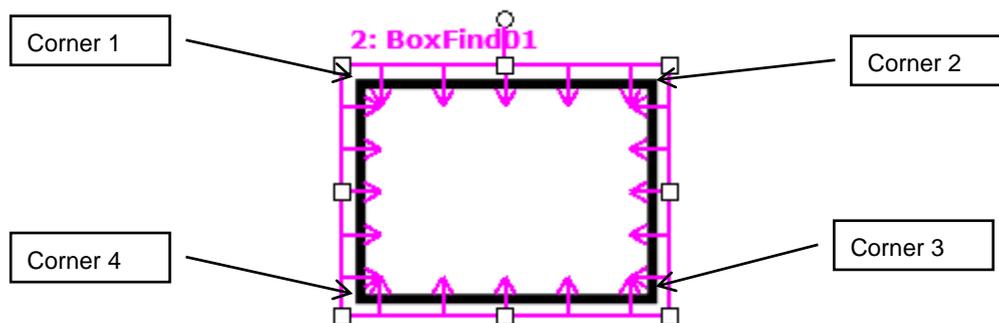
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

## Remarks

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotX3 is used to retrieve the X coordinate for the Corner3 point shown in the diagram below.



RobotX3 results always show the position in the robot coordinate system in mm units.

RobotX3 results are only calculated for vision sequences pre-calibrated for the robot coordinate system. RobotX3 results will return an error for uncalibrated vision sequences.

## See Also

RobotX1 Result, RobotX2 Result, RobotY1 Result, RobotY2 Result, RobotY3 Result, RobotX4 Result, RobotY4 Result, BoxFinder Object

# RobotX4 Result

## Applies To

Vision Object: BoxFinder

## Description

Returns the X (X4) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object*.**RobotX4**[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

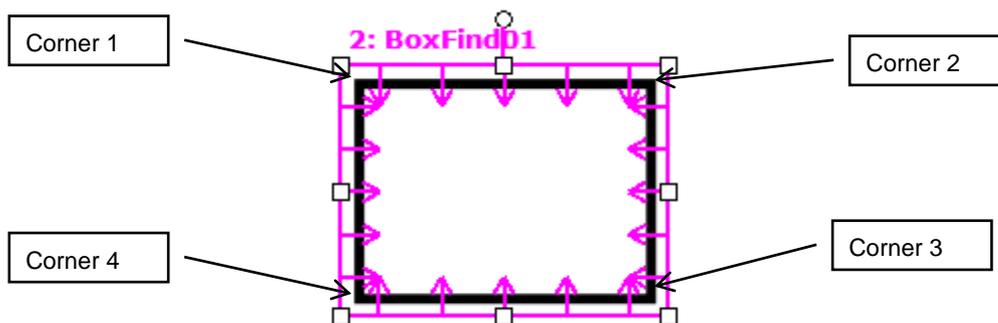
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

## Remarks

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotX4 is used to retrieve the X coordinate for the Corner4 point shown in the diagram below.



RobotX4 results always show the position in the robot coordinate system in mm units.

RobotX4 results are only calculated for vision sequences pre-calibrated for the robot coordinate system. RobotX4 results will return an error for uncalibrated vision sequences.

## See Also

RobotX1 Result, RobotX2 Result, RobotY1 Result, RobotY2 Result, RobotX3 Result, RobotY3 Result, RobotY4 Result, BoxFinder Object

# RobotXYU Result

Runtime only

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, ColorMatch, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineInspector, Point, Polar

## Description

Returns the RobotX, RobotY and RobotU position coordinates of the found part's position with respect to the robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotXYU* [(*result*)], *found*, *xVar*, *yVar*, *uVar*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*found* Boolean variable representing whether or not the part you are looking for was found.

*xVar* Real variable that will contain the X coordinate position of the part.

*yVar* Real variable that will contain the Y coordinate position of the part.

*uVar* Real variable that will contain the angular position (rotation) of the part.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

*found* True or False

*xVar* Real number in millimeters

*yVar* Real number in millimeters

*uVar* Real number in degrees

## Remarks

The RobotXYU result returns a position in the robot coordinate system and therefore can be used for robot guidance applications. The RobotXYU result *xVar* and *yVar* values are always returned in millimeters. The *uVar* value is always returned in degrees. When used for a Point object, *uVar* always returns 0.

It should be noted that the RobotXYU result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then the RobotXYU result cause an error to occur.

## See Also

Blob Object, CameraX Result, CameraY Result, CameraXYU Result, Correlation Object, DefectFinder Object, Edge, Found Result, Geometric Object, PixelXYU Result, Point Object, Polar Object, BoxFinder Object, CornerFinder Object, RobotUOffset Property, RobotX Result, RobotY Result, RobotU Result, RobotToolXYU Result

# RobotY Result

## Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineInspector, Point, Polar, OCR

## Description

Returns the Y coordinate of the found part's position in the robot coordinate system.

## Usage

**VGet** *Sequence.Object*.**RobotY** [(*result*)], *var*

**Sequence** Name of a sequence or string variable containing a sequence name.

**Object** Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

*var* Real number in millimeters.

## Remarks

The RobotX result returns an Y coordinate in the robot coordinate system, and can therefore be used for robot guidance applications. However, keep in mind that a special result called the RobotXYU result is most often used for robot guidance because it returns not only the Y, but also the X, and U coordinate positions as well as whether or not the part was found. See *RobotXYU Result*.

The RobotY Result is always returned in millimeters.

It should be noted that the RobotY result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotY result will cause an error to occur.

## Statistics

For the RobotY Result, the following statistics are available. RobotYMax, RobotYMean, RobotYMin, RobotYStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

Angle Result, Blob Object, CameraXYU Result, Correlation Object, DefectFinder Object, Edge Object, Found Result, Geometric, PixelXYU Result, Point Object, Polar Object, RobotX Result, RobotU Result, RobotXYU Result, CodeReader Object, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object

# RobotY1 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the Y coordinate of the starting point position (Y1) of a Line object in the robot coordinate system.

**BoxFinder:** Returns the Y (Y1) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotY1*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters

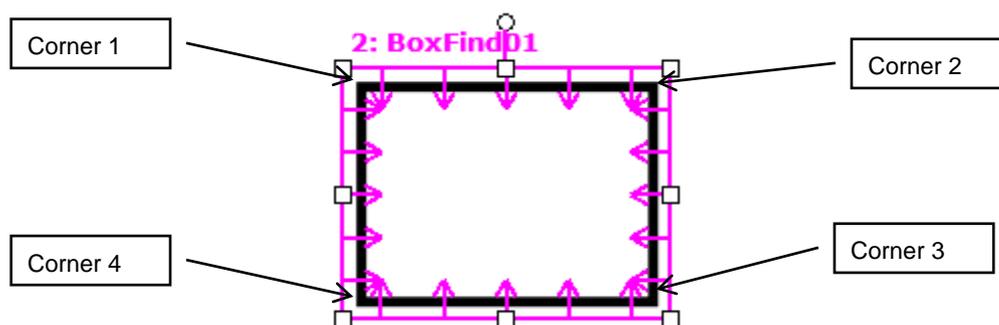
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The RobotY1 and RobotY2 results represent the Y coordinate position starting (Y1) and endpoints (Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (RobotX1, RobotY1) and (RobotX2, RobotY2) coordinate pairs can actually be Robot coordinate positions which match the RobotX and RobotY results for other vision objects. (In other words if a Line object's starting point is defined by a Correlation object, then the (RobotX, RobotY) results from the Correlation object will match the (RobotX1, RobotY1) results for the Line object.)

For BoxFinder

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotY1 is used to retrieve the Y coordinate for the Corner1 point shown in the diagram below.



The RobotY1 result is always in millimeters in the robot coordinate system.

It should be noted that the RobotY1 result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotY1 result will cause an error to occur.

### See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, PixelY Result, PixelY1 Result, PixelY2 Result, RobotX Result, RobotX1 Result, RobotX2 Result, RobotXYU Result, RobotY Result, RobotY2 Result, RobotX3 Result, RobotY3 Result, RobotX4 Result, RobotY4 Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# RobotY2 Result

## Applies To

Vision Objects: Line, LineFinder, BoxFinder

## Description

**Line, LineFinder:** Returns the Y coordinate of the ending point position (Y2) of a Line object in the robot coordinate system.

**BoxFinder:** Returns the Y (Y2) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotY2*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters

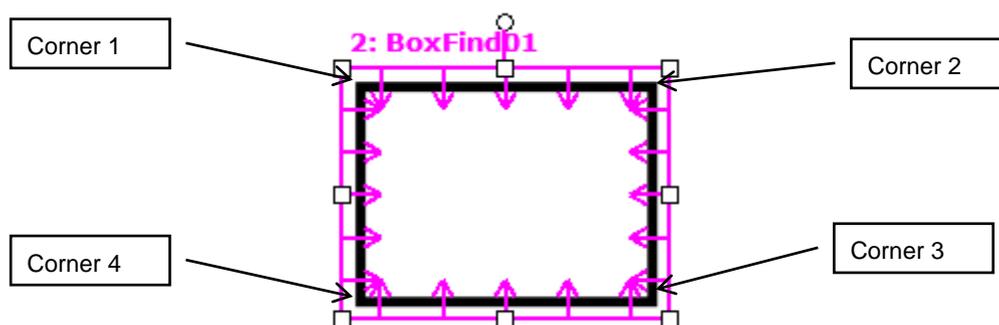
## Remarks

For Line, LineFinder

Every line must have a starting point and ending point. The RobotY1 and RobotY2 results represent the Y coordinate position starting (Y1) and endpoints (Y2) of the specified Line object. Since Line object starting and endpoints can be assigned to other vision objects, the (RobotX1, RobotY1) and (RobotX2, RobotY2) coordinate pairs can actually be robot coordinate positions which match the RobotX and RobotY results for other vision objects. (In other words if a Line object's endpoint is defined by a Correlation object, then the (RobotX, RobotY) results from the Correlation object will match the (RobotX2, RobotY2) results for the Line object.)

For BoxFinder

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotY2 is used to retrieve the Y coordinate for the Corner2 point shown in the diagram below.



The RobotY2 result is always represented in millimeters with respect to the robot coordinate system.

It should be noted that the RobotY2 result can only be calculated for vision sequences which have been calibrated with the robot coordinate system. If no calibration has been assigned to the vision sequence then using VGet to retrieve the RobotY2 result will cause an error to occur.

### See Also

Angle Result, Line Object, LineFinder Object, Object Tab, PixelX Result, PixelX1 Result, PixelX2 Result, PixelY Result, PixelY1 Result, PixelY2 Result, RobotX Result, RobotX1 Result, RobotX2 Result, RobotX3 Result, RobotX4 Result, RobotXYU Result, RobotY Result, RobotY1 Result, RobotY2 Result, RobotY3 Result, RobotY4 Result, X1 Property, X2 Property, Y1 Property, Y2 Property, BoxFinder Object

# RobotY3 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y3) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object*.**RobotY3**[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

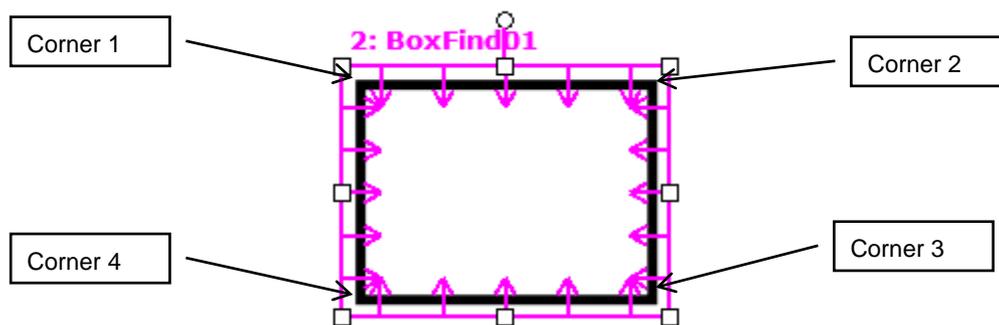
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

## Remarks

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotY3 is used to retrieve the Y coordinate for the Corner3 point shown in the diagram below.



RobotY3 results always show the position in the robot coordinate system in mm units.

RobotY3 results are only calculated for vision sequences pre-calibrated for the robot coordinate system. RobotY3 results will return an error for uncalibrated vision sequences.

## See Also

RobotX1 Result, RobotX2 Result, RobotY1 Result, RobotY2 Result, RobotX3 Result, RobotX4 Result, RobotY4 Result, BoxFinder Object

# RobotY4 Result

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Returns the Y (Y4) coordinate position for corner points of rectangles detected in a robot coordinate system.

## Usage

**VGet** *Sequence.Object.RobotY4*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

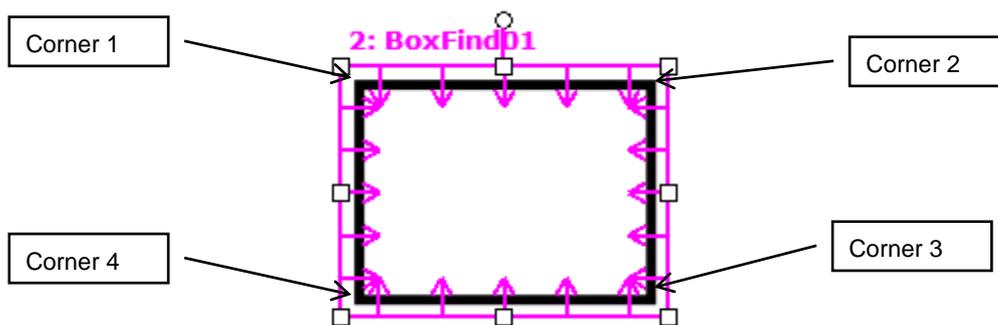
*result* Optional result number. If omitted, the result number is the CurrentResult. Used for objects that return multiple results.

## Values

Real number in millimeters.

## Remarks

The robot coordinates for the four corners of a rectangle can be retrieved as Robot X1, 2, 3, 4 results and Robot Y1, 2, 3, 4 results. RobotY4 is used to retrieve the Y coordinate for the Corner4 point shown in the diagram below.



RobotY4 results always show the position in the robot coordinate system in mm units.

RobotY4 results are only calculated for vision sequences pre-calibrated for the robot coordinate system.  
RobotY4 results will return an error for uncalibrated vision sequences.

## See Also

RobotX1 Result, RobotX2 Result, RobotY1 Result, RobotY2 Result, RobotX3 Result, RobotY3 Result, RobotX4 Result, BoxFinder Object

# RotationAngle Property

## Applies To

Vision Objects: ImageOp

## Description

Sets/returns the angle of rotation for the ImageOp object Rotate operation.

## Usage

**VGet** *Sequence.Object.RotationAngle*, *var*

**VSet** *Sequence.Object.RotationAngle*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Default: 0 degrees

## Remarks

RotationAngle is used to determine how many degrees to rotate the image when the AngleObject property is set to Screen. If AngleObject is not set to Screen, the RotationAngle setting has no effect.

If RotationDirection is set to CCW, the direction of rotation is counter-clockwise when the angle is a positive value.

If RotationDirection is set to CW, the direction of rotation is clockwise when the angle is a positive value.

Pixels that are not in the rotation are set to 0 (black).

## See Also

AngleObject Property, ImageOp Object, Operation Property, RotationDirection Property

# RotationDirection Property

## Applies To

Vision Objects: ImageOp, Contour

## Description

For ImageOp, this property sets or returns the direction of rotation when rotating.

For Contour, this property sets the direction of rotation for contour lines to be output.

## Usage

**VGet** *Sequence.Object.RotationDirection*, *var*

**VSet** *Sequence.Object.RotationDirection*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

1 – CCW Vision constant: VISION\_ROTATIONDIR\_CCW

2 – CW Vision constant: VISION\_ROTATIONDIR\_CW

Default: 1

## Remarks

For ImageOp, the RotationDirection sets the direction of rotation for the image.

If RotationDirection is set to CCW, the direction of rotation is counter-clockwise when the angle set as the RotationAngle is a positive value. If RotationDirection is set to CW, the direction of rotation is clockwise when the angle set as the RotationAngle is a positive value.

For Contour, the RotationDirection sets the direction of rotation for contour lines.

This only applies when the ContourMode is set to Blob or Arc. If RotationDirection is set to CCW, a trajectory traced in a counter-clockwise direction around the work piece is output. If RotationDirection is set to CW, a trajectory traced in a clockwise direction around the work piece is output.

## See Also

Contour Object

# Roughness Result

## Applies To

Vision Objects: Blob, DefectFinder

## Description

Returns the roughness of a blob.

## Usage

**VGet** *Sequence.Object.Roughness* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult.

## Values

Real value equals to or larger than 1.0.

## Remarks

Roughness is a measure of unevenness or irregularity of a blob's surface. It is the ratio of the true perimeter to the convex perimeter of a blob. The convex perimeter is the length of a line connecting all the extremities of the blob directly, while the true perimeter is the length of a line connecting every pixel along the blob's edge (Perimeter result). Smooth convex blobs have a roughness of 1.0 (the minimum), whereas rough blobs have a higher value than 1 because their true perimeter is bigger than their convex perimeter.

## See Also

Blob Object, DefectFinder Object, Compactness Result, Holes Result, Perimeter Result

# RuntimeAcquire Property

## Applies To

Vision Sequence

## Description

The RuntimeAcquire property tells the vision sequence which method to use to acquire an image for use with that sequence.

## Usage

**VGet** *Sequence.Object.RuntimeAcquire*, *var*

**VSet** *Sequence.Object.RuntimeAcquire*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - None Vision constant: `VISION_ACQUIRE_NONE`  
 This tells the vision system not to acquire an image before processing the sequence. Simply use the image which is already in the frame buffer specified by the ImageBuffer property. This is useful when a series of vision sequences need to work from the same image. For example, you could acquire an image in sequence #1. During this sequence you could also execute some vision objects. Next, assume that you want to use another vision sequence on the same image. Simply set the RuntimeAcquire property to None for the 2nd vision sequence and the same image will be used for both sequences.

1 – Stationary Vision constant: `VISION_ACQUIRE_STATIONARY`  
 The camera is stationary (not moving). A new image is acquired at the start of the vision sequence. This is the normal method for running a vision sequence. Each time a new vision sequence is executed, a new image is acquired at the start of the sequence. The ExposureTime property affects how stationary images are acquired. See ExposureTime for details.

2 - Strobed Vision constant: `VISION_ACQUIRE_STROBED`  
 The image acquisition starts by the trigger input. Also the strobe output is output. This is the mechanism for setting up strobed lighting for capturing moving images within the image buffer. See the remarks section below for more details.

Default: 1 - Stationary

## Remarks

The RuntimeAcquire property is very important to understand. There are 3 settings for the RuntimeAcquire property as explained in the Values section. The most common of the 3 is the 1–Stationary setting since in most cases you will want to acquire a new image at the beginning of each vision sequence.

However, you may also use the same image for more than 1 sequence. Simply acquire an image for the 1st sequence and then use the same image in the 2nd sequence making sure to set the RuntimeAcquire property to 0–None for the 2nd sequence.

The 3rd Acquisition method is called 2–Strobed. This acquires an image as follows.

When the vision sequence is run, the sequence will wait for an input trigger. At the instance the trigger input goes active, the vision sequence will initiate an acquisition, thus capturing the image at the same time as the strobe of the light source. Please see *Image Acquisition* in the *Vision Guide 6.0* manual for more details.

## See Also

Object Tab, RuntimeFreeze Property, Vision Sequences

## RuntimeContour Property

### Applies To

Vision Object: Contour  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Specifies whether to output contour lines of a work piece when executing a Contour object.

### Usage

**VGet** *Sequence.Object.RuntimeContour*, *var*

**VSet** *Sequence.Object.RuntimeCotour*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 - False Outputs a pre-taught contour.

1 - True Outputs the work piece contour when executing an object.

Default: 1

### Remarks

RuntimeContour specifies whether to output contour lines of a work piece when executing an object.

When set to False, a pre-taught contour will be output. When set to True, at runtime a contour will be traced and output from data reflected in the image.

### See Also

Contour Object

# RuntimeFreeze Property

## Applies To

Vision Sequence

## Description

Defines whether or not to freeze the display of an image acquired during a vision sequence.

## Usage

**VGet** *Sequence.RuntimeFreeze*, *var*

**VSet** *Sequence.RuntimeFreeze*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Do not freeze the image. (Image display area will show live image)

1 – True Freeze the image. (Image display area will show frozen image)

Default: 1 – True

## Remarks

The RuntimeFreeze property lets you choose whether to show the image acquired during a sequence, or show live video after the sequence runs.

Note that when RuntimeFreeze is false and live video is displayed until the next sequence runs, it can slow down vision processing, because when a sequence needs to grab an image, it must wait for the current live grab to complete. For fastest processing, use RuntimeFreeze = False only when necessary.

## See Also

RuntimeAcquire Property, Sequence Tab, Vision Sequences

# SamplingPitchProperty

## Applies To

Vision Object: Contour  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Specifies the degree to which contour points are thinned out.

## Usage

**VGet** *Sequence.Object.SamplingPitch, var*

**VSet** *Sequence.Object. SamplingPitch, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0-100.

## Remarks

With regard to Contour objects, after tracing object contours, the number of contour points is reduced according to the SamplingPitch property value. Increase the SamplingPitch value to reduce the number of contour points. Further, note that if the ContourTolerance property value is low, the number of contour points may not be thinned out along certain segments in order to acquire an accurate trajectory for the work piece contour. If unnecessary contour points are output, increase the ContourTolerance property value.

## See Also

Contour Object, ContourTolerance Property

# SaveImage Property

Design-time Only

## Applies To

Vision Sequence

## Description

Saves the currently displayed image on disk.

## Remarks

SaveImage allows you to save images to disk that can be used by the ImageFile property. The file can be saved in the following formats:

BMP (default format), TIF, or JPG.

## See Also

Object Tab, ImageFile Property, ImageSource Property, VSaveImage Statement

## Scale Result

### Applies To

Vision Objects: Geometric, Correlation

### Description

Returns the scale of the object found.

### Usage

**VGet** *Sequence.Object.Scale* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult.

### Values

Real number representing the scale factor of the object found.

### Remarks

You can use the Scale result to determine the size of the object found compared to the size of the model trained. When ScaleEnable is False, a small variation in scale is tolerated, so the Scale result may not always be exactly 1. To ensure Scale to be only 1, you must set ScaleEnable to True, and set ScaleFactorMin = 1.0, and ScaleFactorMax = 1.0.

### See Also

Geometric Object, Correlation Object, ScaleEnable Property, ScaleFactorMax Property, ScaleFactorMin Property, ScaleTarget Property, Object Tab

# ScaleEnable Property

## Applies To

Vision Objects: Geometric

## Description

Enables a greater range of scale detection.

## Usage

**VGet** *Sequence.Object.ScaleEnable*, *var*

**VSet** *Sequence.Object.ScaleEnable*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Allow only a small variation in scale.

1 – True Do allow a greater range of scale during search.

Default: 0 – False

## Remarks

Use `ScaleEnable` to search for objects within a specified scale range. Use `ScaleFactorMin` and `ScaleFactorMax` to set the range, and `ScaleTarget` to specify the target scale. When `ScaleEnable` is `False`, a small variation in scale is tolerated, so the `Scale` result may not always be exactly 1. To ensure `Scale` to be only 1, you must set `ScaleEnable` to `True`, and set `ScaleFactorMin` = 1.0, and `ScaleFactorMax` = 1.0.

## See Also

Geometric Object, Object Tab, Scale Result, ScaleFactorMax Property, ScaleFactorMin Property, ScaleTarget Property, Vision Sequences

## ScaleFactorMax Property

### Applies To

Vision Objects: Geometric

### Description

Sets / returns the maximum scale factor applied to the ScaleTarget value.

### Usage

**VGet** *Sequence.Object.ScaleFactorMax*, *var*

**VSet** *Sequence.Object.ScaleFactorMax*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

1.0 - 2.0

Default: 2.0

### Remarks

ScaleFactorMax and ScaleFactorMin determine the scale range to search for as applied to the ScaleTarget property. The maximum scale found is ScaleFactorMax \* ScaleTarget.

To use ScaleFactorMax and ScaleFactorMin, you must set the ScaleEnabled property to 1–True.

### See Also

Geometric Object, Object Tab, Scale Result, ScaleEnable Property, ScaleFactorMin Property, ScaleTarget Property

# ScaleFactorMin Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns the minimum scale factor applied to the ScaleTarget value.

## Usage

**VGet** *Sequence.Object.ScaleFactorMin*, *var*

**VSet** *Sequence.Object.ScaleFactorMin*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

0.5 - 1.0

Default: 0.5

## Remarks

ScaleFactorMax and ScaleFactorMin determine the scale range to search for as applied to the ScaleTarget property. The minimum scale found is  $\text{ScaleFactorMin} * \text{ScaleTarget}$ .

To use ScaleFactorMax and ScaleFactorMin, you must set the ScaleEnabled property to 1–True.

## See Also

Geometric Object, Object Tab, Scale Result, ScaleEnable Property, ScaleFactorMax Property, ScaleTarget Property

# ScaleTarget Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns the expected scale of the object you are searching for.

## Usage

**VGet** *Sequence.Object.ScaleTarget, var*

**VSet** *Sequence.Object.ScaleTarget, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

0.5 - 2.0

Default: 1.0

## Remarks

To use `ScaleTarget`, you must set the `ScaleEnabled` property to 1–True. The actual scale range is determined by `ScaleTarget`, `ScaleFactorMin`, and `ScaleFactorMax`.

The range is determined as follows:

minimum scale = `ScaleFactorMin * ScaleTarget`

maximum scale = `ScaleFactorMax * ScaleTarget`

## See Also

Geometric Object, Object Tab, Scale Result, ScaleEnable Property, ScaleFactorMax Property, ScaleFactorMin Property

# ScaleTargetPriority Property

## Applies To

Vision Object: Geometric

## Description

Sets / returns whether to find objects whose scale is near ScaleTarget preferentially during object search.

## Usage

**VGet** *Sequence.Object.ScaleTargetPriority*, *var*

**VSet** *Sequence.Object.ScaleTargetPriority*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean value or expression for the new value of the property.

## Values

True : Search while giving a priority to objects whose scale is near ScaleTarget.

False : Search the range between ScaleFactorMax and ScaleFactorMin evenly.

Default:True

## Remarks

By setting this property to True when scale variation for the object to be found is small, search speed may increase. When the variation in scale for the object to be found is large, set this property to False. Objects will be found with either setting, but setting the value according to the variation in scale may increase search speed.

## See Also

Geometric Object, ScaleFactorMax Property, ScaleFactorMin Property, ScaleTarget Property

## Score Result

### Applies To

Vision Objects: ColorMatch, Correlation, Edge, Geometric, Polar

### Description

Returns an integer value which represents the level at which the runtime object was found. For Correlation, Geometric, and Polar objects, the score indicates how well an object matches the model for which it is searching. For the Edge object, the Score result measures the level at which a contrast between Light to Dark or Dark to Light transition takes place. For ColorMatch objects, this refers to the matching degree of colors in a color space.

### Usage

**VGet** *Sequence.Object.Score* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional result number. If omitted, the result number is the CurrentResult.

### Values

0 - 1000

### Remarks

The Score result is the basic value used to measure how well a feature in the search area matches a previously taught model. If the Score is not greater than or equal to the Accept property value, the object is considered not found.

Normally a low Score result means that the image doesn't contain any patterns which closely match the Model. However, it should be noted that a low Score result can also be obtained if the Accept property and Confusion Properties are not set high enough. If these properties are set low, the first pattern found that meets the Accept and Confusion property thresholds will be returned as found. This could mean that other patterns in the image which may have been better matches would not be found.

Don't expect that your Score results will always be close to 1000. Just because a Score result is returned which is relatively low (as compared to a perfect score of 1000) doesn't mean that the application cannot be done or isn't reliable. There are many different application types and each has its own circumstances which affect the Score results. Some applications will return Score results of less than 500 while others may always return Score results over 900. Proper settings of lighting, part presentation, overall vision application setup, and proper vision tool usage will all affect the Score results.

For ColorMatch objects, a score is calculated based on the distance in a color space. When the result is within the permitted range centering on the model color, a score of 1000 is returned. If the result is away from this permitted range, a score value is reduced according to its distance.

### Statistics

For the Score Result, the following statistics are available. ScoreMax, ScoreMean, ScoreMin, ScoreStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

**See Also**

Accept Property, CodeReader Object, Confusion Property, Correlation Object, Edge Object, Found Result, Geometric Object, Object Tab, OCR Object, Polar Object

## ScoreWeightContrast Property

### Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

### Description

Sets the percentage of the score which is affected by the contrast result.

### Usage

**VGet** *Sequence.Object.ScoreWeightContrast*, *var*

**VSet** *Sequence.Object.ScoreWeightContrast*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value from 0 - 100%

Default: 50

### Remarks

The ScoreWeightContrast is a percentage value that tells the Edge object how much to weigh the contrast result in the final score. ScoreWeightContrast works with ScoreWeighStrength. Both of these property values must add up to 100%. When you set one property, the system will automatically set the other property to the correct value.

### See Also

ArcFinder Object, ArcInspector Object, Edge Object, Contrast Result, LineFinder Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object, ScoreWeightStrength Property

# ScoreWeightStrength Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

## Description

Sets the percentage of the score which is affected by the strength result.

## Usage

**VGet** *Sequence.Object.ScoreWeightStrength, var*

**VSet** *Sequence.Object.ScoreWeightStrength, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value from 0 - 100%

Default: 50

## Remarks

The ScoreWeightStrength is a percentage value that tells the Edge object how much to weigh the edge strength result in the final score. ScoreWeightStrength works with ScoreWeighContrast. Both of these property values must add up to 100%. When you set one property, the system will automatically set the other property to the correct value.

## See Also

Contour Object, Edge Object, Contrast Result, ScoreWeightContrast Property

## SearchLength Property

### Applies To

Vision Objects: BoxFinder, CornerFinder  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Defines the length of the edge search range.

### Usage

**VGet** *Sequence.Object.SearchLength, var*

**VSet** *Sequence.Object. SearchLength, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

### Values

Positive real value from 10 to  $\text{SearchWinWidth}/2 - 10$  or  $\text{SearchWinHeight}/2 - 10$

Use the smaller value of the two: SearchWinWidth or SearchWinHeight.

### Remarks

Specify the length of the edge search range extending out from each edge of the search window.

While you can use SearchLength1 to 4 to set the edge search range for each edge individually, you can also use SearchLength to set the length for each edge together.

### See Also

BoxFinder Object, CornerFinder Object, SearchLength1 Property, SearchLength2 Property, SearchLength3 Property, SearchLength4 Property

# SearchLength1 Property

## Applies To

Vision Objects: BoxFinder, CornerFinder  
CV2 firmware Ver. 3.1.0.0 or late

## Description

Defines the length of the edge search range.

## Usage

**VGet** *Sequence.Object.SearchLength1, var*

**VSet** *Sequence.Object. SearchLength1, value*

*Sequence* Name of a sequence or string variable.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

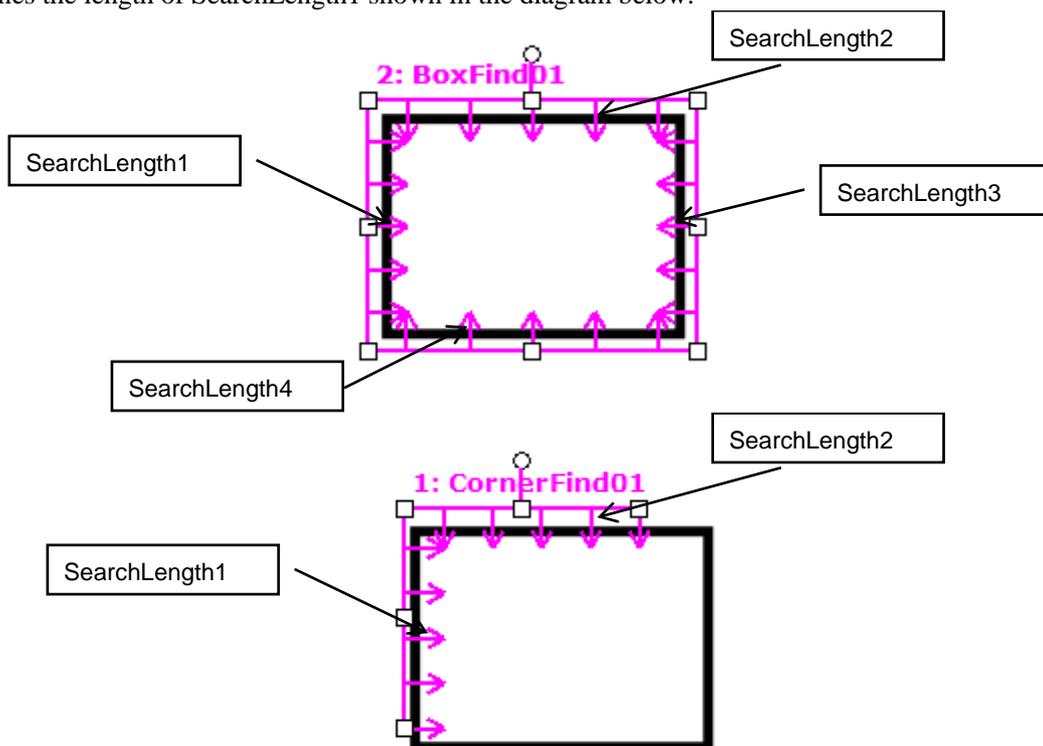
*value* Real value or expression for the new value of the property.

## Values

Positive real value from 10 to  $\text{SearchWinWidth}/2 - 10$

## Remarks

Defines the length of SearchLength1 shown in the diagram below.



## See Also

BoxFinder Object, CornerFinder Object, SearchLength Property, SearchLength2 Property, SearchLength3 Property, SearchLength4 Property

# SearchLength2 Property

## Applies To

Vision Objects: BoxFinder, CornerFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Defines the length of the edge search range.

## Usage

**VGet** *Sequence.Object.SearchLength2*, *var*

**VSet** *Sequence.Object.SearchLength2*, *value*

*Sequence* Name of a sequence or string variable.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

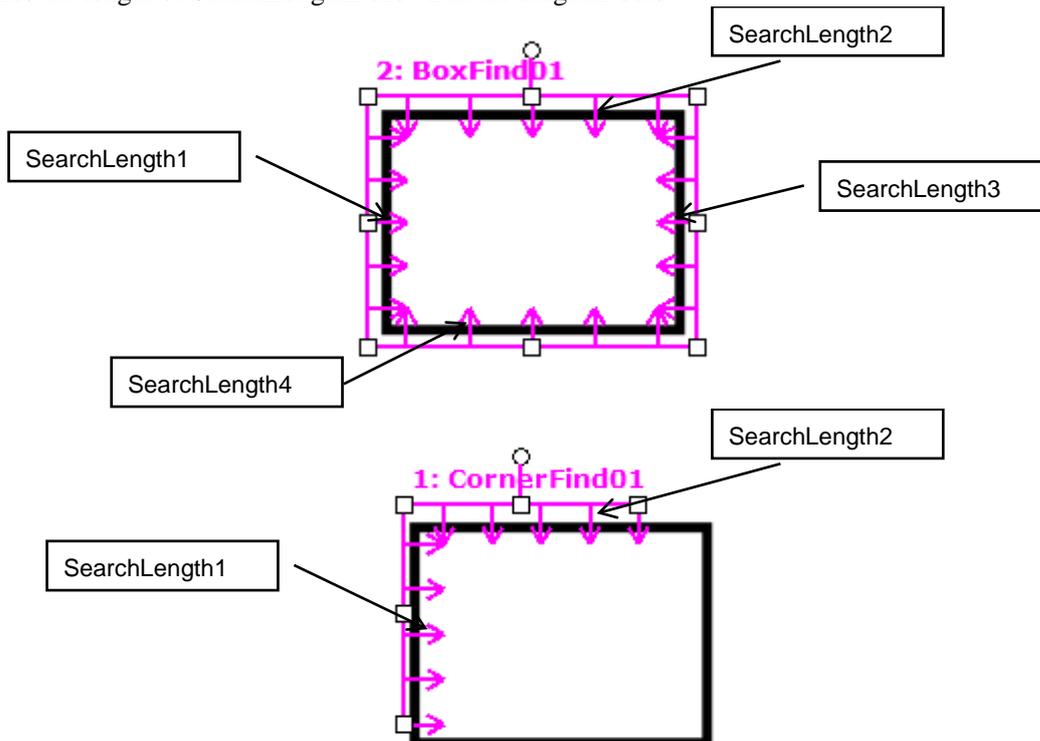
*value* Real value or expression for the new value of the property.

## Values

Positive real value from 10 to  $\text{SearchWinHeight}/2 - 10$

## Remarks

Defines the length of SearchLength2 shown in the diagram below.



## See Also

BoxFinder Object, CornerFinder Object, SearchLength Property, SearchLength1 Property, SearchLength3 Property, SearchLength4 Property

# SearchLength3 Property

## Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Defines the length of the edge search range.

## Usage

**VGet** *Sequence.Object.SearchLength3*, *var*

**VSet** *Sequence.Object.SearchLength3*, *value*

*Sequence* Name of a sequence or string variable.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

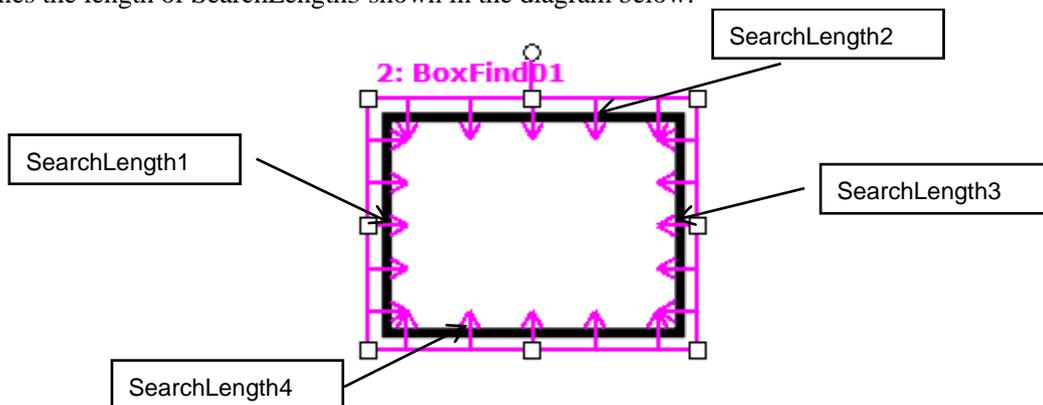
*value* Real value or expression for the new value of the property.

## Values

Positive real value from 10 to  $\text{SearchWinWidth}/2 - 10$

## Remarks

Defines the length of SearchLength3 shown in the diagram below.



## See Also

BoxFinder Object, CornerFinder Object, SearchLength Property, SearchLength1 Property, SearchLength2 Property, SearchLength4 Property

## SearchLength4 Property

### Applies To

Vision Object: BoxFinder  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Defines the length of the edge search range.

### Usage

**VGet** *Sequence.Object.SearchLength4*, *var*

**VSet** *Sequence.Object*, **SearchLength4**, *value*

*Sequence* Name of a sequence or string variable.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

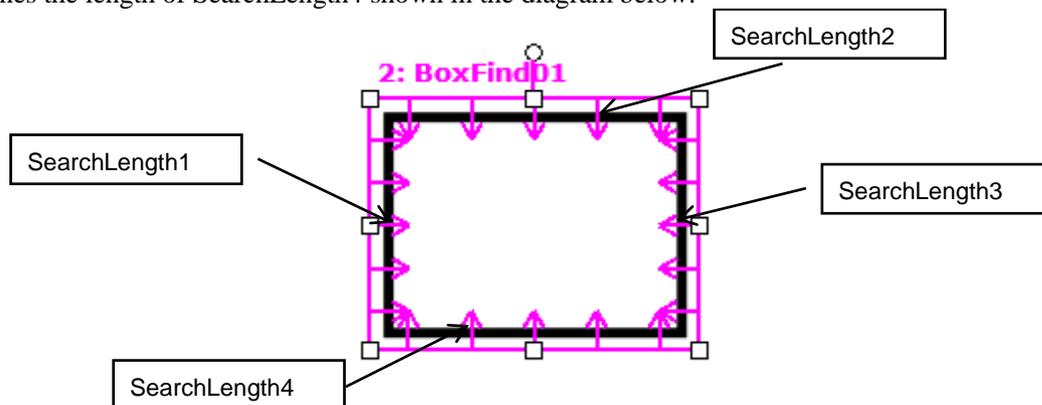
*value* Real value or expression for the new value of the property.

### Values

Positive real value from 10 to  $\text{SearchWinHeight}/2 - 10$

### Remarks

Defines the length of SearchLength4 shown in the diagram below.



### See Also

BoxFinder Object, CornerFinder Object, SearchLength Property, SearchLength1 Property, SearchLength2 Property, SearchLength3 Property

# SearchPolarity Property

## Applies To

Vision Object: Geometric

## Description

Sets and returns which polarity search mode to use.

## Usage

**VGet** *Sequence.Object.SearchPolarity*, *var*

**VSet** *Sequence.Object.SearchPolarity*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

## Values

0 - Find objects only with same polarity as model

1 - Find objects with same polarity or reverse polarity

2 - Find objects only with reverse polarity

Default: 0

## Remarks

Use SearchPolarity to configure the Geometric tool to search for objects with the same polarity as the model, same and reverse polarity, or reverse polarity.

### Note:

SearchPolarity replaces SearchReversed, which was used in versions for EPSON RC+ 7.0 prior to v7.1.0, and CV1 firmware prior to v2.2.0.0.

## See Also

Geometric Object, Reversed Result

## SearchReducedImage Property

### Applies To

Vision Object: Geometric

### Description

Sets / returns whether to use a reduced size image during the search.

### Usage

**VGet** *Sequence.Object.SearchReducedImage, var*

**VSet** *Sequence.Object.SearchReducedImage, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean value or expression for the new value of the property.

### Values

True: Use a reduced size image

False: Do not use a reduced size image

Default: False

### Remarks

By setting this property to True, a reduced size input image is used. Reduction ratio is set automatically within the system.

This property may reduce time to detect the object when input image has many feature points. Since the effect of this property varies depending on the input image and the model, check the operation with the actual system and decide on the best value.

The position and angle results will not be as accurate when SearchReducedImage is True, but speed could be increased.

### See Also

Geometric Object, ScaleTargetPriority Property, Confusion Property, Accept Property

# SearchType Property

## Applies To

Vision Object: Edge

## Description

Sets / returns the search type for an Edge object..

## Usage

**VGet** *Sequence.Object.SearchType*, *var*

**VSet** *Sequence.Object.SearchType*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

## Values

Following integer values representing the search window type

1 - Line (VISION\_SEARCHTYPE\_LINE)

2 - Arc (VISION\_SEARCHTYPE\_ARC)

## Remarks

Use SearchType to change the type of search to use for an Edge object. Available values are as follows:

1 - Line Search for edges along a line

2 - Arc Search for edges along an arc.

## See Also

Edge Object

# SearchWidth Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

## Description

The SearchWidth property specifies the width of the search for edge detection.

## Usage

**VGet** *Sequence.Object.SearchWidth*, *var*

**VSet** *Sequence.Object.SearchWidth*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer number from 3-99 pixels.

Default: 3

## Remarks

Normally the Edge object will work fine with the default 3 pixel search width. However, for some applications you may want to increase the width of the edge search to help find an edge with some imperfections. By increasing the SearchWidth, the Edge object can gather more information to determine where the edge is. During processing, the 2-dimensional search window is transformed to a 1-dimension row of grayscale values. Edge filters are applied to this row of values to determine where the edge is. Using a wider search window helps ignore imperfections in the edge.

The figures below show an edge object with SearchWidth set to 3 on the left and SearchWidth set to 30 on the right. The edge object on the left finds the bump, whereas the edge object on the right finds the correct edge because the wider search width will cause the projected search line to favor the true edge.



## See Also

Edge Object, Score Result, Object Tab

# SearchWin Property

Runtime only

## Applies To

Vision Objects: Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Geometric, ImageOp, LineFinder, OCR, Text

## Description

Defines the position and size of a search window.

## Usage

**VGet** *Sequence.Object.SearchWin, LeftVar, TopVar, WidthVar, HeightVar*

**VSet** *Sequence.Object.SearchWin, Left, Top, Width, Height*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*LeftVar* Integer variable representing the left position of the upperleft corner of the search window.

*TopVar* Integer variable representing the top position of the upperleft corner of the search window.

*WidthVar* Integer variable representing the width of the search window.

*HeightVar* Integer variable representing the height of the search window.

*Left* Integer expression representing the left position of the upperleft corner of the search window.

*Top* Integer expression representing the top position of the upperleft corner of the search window.

*Width* Integer expression representing the width of the search window.

*Height* Integer expression representing the height of the search window.

## Values

All Values are in pixels. See the SearchWinLeft, SearchWinTop, SearchWinWidth, and SearchWinHeight Properties for exact value data.

## Remarks

The SearchWin property was added to provide easy access to the SearchWinTop, SearchWinLeft, SearchWinWidth and SearchWinHeight Properties from the SPEL<sup>+</sup> Language. The SearchWin property allows the setting of all four properties with one function call. There are cases where the user may want to define the position and size of the search window dynamically and for that reason the SearchWin property was created.

The Left and Top values specify the location of the upperleft corner of the window, even when the window is rotated.

Avoid setting SearchWin too large. If the value is too large, the detection time gets longer and may also result in false detection.

## See Also

Blob Object, CodeReader Object, Correlation Object, Geometric Object, ImageOp Object, LineFinder Object, Object Tab, OCR Object, SearchWinHeight Property, SearchWinLeft Property, SearchWinTop Property, SearchWinWidth Property, BoxFinder Object, Contour Object, CornerFinder Object, Text Object

## SearchWinAngle Property

### Applies To

Vision Object: Blob, BoxFinder, Contour, CornerFinder, Correlation, Geometric, ColorMatch, ImageOp, LineFinder

### Description

Sets and returns the search window angle.

### Usage

**VGet** *Sequence.Object.SearchWinAngle*, *var*

**VSet** *Sequence.Object.SearchWinAngle*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

### Values

Real value within a range of  $\pm 180$  degrees

### Remarks

SearchWinAngle is only valid when SearchWinType is set to RotatedRectangle.

### See Also

SearchWinCenterX Property, SearchWinCenterY Property, SearchWinType Property

## SearchWinCenterX Property

### Applies To

Vision Object: Blob, BoxFinder, ColorMatch, Contour, CornerFinder, Correlation, Geometric, ImageOp, LineFinder

### Description

Sets and returns the X coordinate value of the center of the search window.

### Usage

**VGet** *Sequence.Object.SearchWinCenterX, var*

**VSet** *Sequence.Object.SearchWinCenterX, value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

### Values

Integer ranging from 0 to "image width - 1" in pixels

### Remarks

Values are automatically updated when the search window is moved or resized from the GUI.

### See Also

SearchWinCenterX, SearchWinCenterY, SearchWinType, SearchWinHeight, SearchWinWidth

## SearchWinCenterY Property

### Applies To

Vision Object: Blob, BoxFinder, ColorMatch, Contour, CornerFinder, Correlation, Geometric, LineFinder, ImageOp

### Description

Sets and returns the Y coordinate value of the center of the search window.

### Usage

**VGet** *Sequence.Object.SearchWinCenterY*, *var*

**VSet** *Sequence.Object.SearchWinCenterY*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

### Values

Integer ranging from 0 to “search window height - 1” in pixels

### Remarks

Values are automatically updated when the search window is moved or resized from the GUI.

### See Also

SearchWinCenterX Property, SearchWinCenterY Property, SearchWinType Property, SearchWinHeight Property, SearchWinWidth Property, Contour Object

# SearchWinHeight Property

## Applies To

Vision Objects: Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Geometric, ImageOp, LineFinder, OCR, Text

## Description

Defines the height of an object's search window.

## Usage

**VGet** *Sequence.Object.SearchWinHeight*, *var*

**VSet** *Sequence.Object.SearchWinHeight*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer number in pixels from 10 to video height - SearchWinTop

Default: 100

## Remarks

The SearchWinHeight property is available for the Blob, Correlation, Geometric, and ImageOp objects. Each of these object types have similar rectangular Search Windows used to define the area to search within. The SearchWinHeight property is set automatically when the user drags the upper or lower horizontal window handles for the Search Window of each object type.

There are cases where the user may want to expand or position the Search Window dynamically and for that reason the SearchWinHeight property can also be set from the SPEL<sup>+</sup> Language.

Do not the set the SearchWinHeight value too large. If the value is too large, the detection time gets longer and may result in false detection.

## See Also

Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Geometric Object, ImageOp Object, Object Tab, OCR Obejct, BoxFinder Obejct, Contour Object, CornerFinder Obejct, Text Object, SearchWinLeft Property, SearchWinTop Property, SearchWinWidth Property, Window Property

## SearchWinLeft Property

### Applies To

Vision Objects: Blob, BoxFinder, CodeReader, ColorMatch, Contour, Correlation, DefectFinder, Geometric, ImageOp, LineFinder, OCR, CornerFinder, Text

### Description

Defines the X coordinate of the upperleft corner of an object's search window.

### Usage

**VGet** *Sequence.Object.SearchWinLeft*, *var*

**VSet** *Sequence.Object.SearchWinLeft*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer number in pixels from 0 to video width - SearchWinWidth

### Remarks

The SearchWinLeft property is available for objects with rectangular search windows used to define the area to search within. The SearchWinLeft property is set automatically when the user drags the search window in the GUI.

There are cases where the user may want to position the search window dynamically and for that reason the SearchWinLeft property can also be set from the SPEL<sup>+</sup> Language.

### See Also

Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Geometric Object, ImageOp Object, LineFinder Object, Object Tab, OCR Obejct, BoxFinder Obejct, Contour Object, CornerFinder Obejct, Text Object, SearchWinHeight Property, SearchWinTop Property, SearchWinWidth Property, Window Property

# SearchWinTop Property

## Applies To

Vision Objects: Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Geometric, ImageOp, LineFinder, OCR, Text

## Description

Defines the Y coordinate of the upperleft corner of an object's search window.

## Usage

**VGet** *Sequence.Object.SearchWinTop*, *var*

**VSet** *Sequence.Object.SearchWinTop*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer value in pixels from 0 to video height - SearchWinHeight

## Remarks

The SearchWinTop property is available for objects with rectangular search windows used to define the area to search within. The SearchWinTop property is set automatically when the user drags the search window in the GUI.

There are cases where the user may want to position the Search Window dynamically and for that reason the SearchWinTop property can also be set from the SPEL<sup>+</sup> Language.

## See Also

Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Geometric Object, ImageOp Object, LineFinder Object, Object Tab, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object, SearchWinHeight Property, SearchWinLeft Property, SearchWinWidth Property, Window Property

## SearchWinType Property

### Applies To

Vision Objects: Blob, ColorMatch, Contour, Correlation, DefectFinder, Geometric, LineFinder, ImageOp

### Description

Sets / returns the search window type.

### Usage

**VGet** *Sequence.Object.SearchWinType*, *var*

**VSet** *Sequence.Object.SearchWinType*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer value or expression for the new value of the property.

### Values

The following integer values are used to specify the search window type:

1 - Rectangle (VISION\_WINTYPE\_RECTANGLE)

2 - RotatedRectangle (VISION\_WINTYPE\_ROTATEDRECT)

3 - Circle (VISION\_WINTYPE\_CIRCLE)

### Remarks

Sets and returns the search window type. Available values are as follows:

1 - Rectangle                Rectangular search window that cannot be rotated.

2 - RotatedRectangle      Rectangular search window that can be rotated.

3 - Circle                  Circular search window

### See Also

SearchWinCenterX Property, SearchWinCenterY Property, SearchWinHeight Property, SearchWinWidth Property

# SearchWinWidth Property

## Applies To

Vision Objects: Blob, BoxFinder, CodeReader, ColorMatch, Correlation, Contour, CornerFinder, DefectFinder, Geometric, ImageOp, LineFinder, OCR, Text

## Description

Defines the width of the an object's search window.

## Usage

**VGet** *Sequence.Object.SearchWinWidth*, *var*

**VSet** *Sequence.Object.SearchWinWidth*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Integer number in pixels from 10 to video width - SearchWinLeft

Default: 100

## Remarks

The SearchWinWidth property is available for objects with rectangular search windows used to define the area to search within. The SearchWinWidth property is set automatically when the user resizes the search window width in the GUI.

There are cases where the user may want to expand or position the Search Window dynamically and for that reason the SearchWinWidth property can also be set from the SPEL<sup>+</sup> Language.

Do not the set the SearchWinWidth value too large. If the value is too large, the detection time gets longer and may result in false detection.

## See Also

Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Geometric Object, ImageOp Object, LineFinder Object, Object Tab, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object, Text Object, SearchWinHeight Property, SearchWinLeft Property, SearchWinTop Property, Window Property

## SeparationAngle Property

### Applies To

Vision Objects: Geometric

### Description

Sets / returns the minimum angle allowed between found objects.

### Usage

**VGet** *Sequence.Object.SeparationAngle*, *var*

**VSet** *Sequence.Object.SeparationAngle*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real value from 0 - 180 degrees

0 = Disabled

Default: 10

### Remarks

Use SeparationAngle to specify the minimum angle required between found objects.

SeparationAngle works with SeparationMinX, SeparationMinY, SeparationScale. Note that only one separation condition needs to be satisfied for objects to be considered found.

### See Also

Geometric Object, Object Tab, SeparationMinX Property, SeparationMinY Property, SeparationScale Property

# SeparationMinX Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns the minimum distance along the X axis allowed between found objects.

## Usage

**VGet** *Sequence.Object.SeparationMinX*, *var*

**VSet** *Sequence.Object.SeparationMinX*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real value from 0 - 100% of model width

0 = Disabled

Default: 10

## Remarks

Use SeparationMinX to specify the minimum distance along the X axis required between found objects. SeparationMinX is a percentage of model width.

SeparationMinX works with SeparationAngle, SeparationMinY, SeparationScale. Note that only one separation condition needs to be satisfied for objects to be considered found.

## See Also

Geometric Object, Object Tab, SeparationAngle Property, SeparationMinY Property, SeparationScale Property

## SeparationMinY Property

### Applies To

Vision Objects: Geometric

### Description

Sets / returns the minimum distance along the Y axis allowed between found objects.

### Usage

**VGet** *Sequence.Object.SeparationMinY*, *var*

**VSet** *Sequence.Object.SeparationMinY*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

### Values

Real value from 0 - 100% of model width

0 = Disabled

Default: 10

### Remarks

Use SeparationMinY to specify the minimum distance along the Y axis required between found objects. SeparationMinY is a percentage of model height.

SeparationMinY works with SeparationAngle, SeparationMinX, SeparationScale. Note that only one separation condition needs to be satisfied for objects to be considered found.

### See Also

Geometric Object, Object Tab, SeparationAngle Property, SeparationMinX Property, SeparationScale Property

# SeparationScale Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns the minimum scale difference allowed between found objects.

## Usage

**VGet** *Sequence.Object.SeparationScale*, *var*

**VSet** *Sequence.Object.SeparationScale*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real expression for the new value of the property.

## Values

Real value from 1.0 - 4.0

Default: 1.1

## Remarks

Use SeparationScale to specify the minimum scale difference required between found objects.

SeparationScale works with SeparationMinX, SeparationMinY, SeparationScale. Note that only one separation condition needs to be satisfied for objects to be considered found.

## See Also

Geometric Object, Object Tab, SeparationAngle Property, SeparationMinX Property, SeparationMinY Property

# Sequences Property

Runtime only

## Applies To

Vision sequences  
CV2 firmware Ver. 3.1.0.0 or later

## Description

This is a sequence array. This is used to access Sequences properties and results with an index.

## Usage

**VGet** Sequences(*index*).*Property*, *var*

**VGet** Sequences(*index*).*Result*, *var*

*index*            An integer expression representing a sequence index

*Property*        The name of the Sequences property being accessed

*Result*          The name of the Sequences results being accessed

*var*              A variable showing the property or results value  
The data type will vary depending on whether property or results are specified.

## Remarks

The Sequences property can be used to use an index instead of a name to access sequences.

## See Also

Count Property

# SharedEdges Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns whether to allow edges to be shared between found objects.

## Usage

**VGet** *Sequence.Object.SeparationScale*, *var*

**VSet** *Sequence.Object.SeparationScale*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

0 – False Shared edges are not allowed.

1 – True Shared edges are allowed.

Default: 0 – False

## Remarks

You can choose to allow found objects to share edges by setting SharedEdges to 1–True. Otherwise, edges that can be part of more than one found object are considered part of the found object with the greatest score.

## See Also

Geometric Object, Object Tab

## ShowAllResults Result

Design time only

### Applies To

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, CodeReader, ColorMatch, Contour, CornerFinder, Correlation, DefectFinder, Edge, Geometric, LineFinder, LineInspector, OCR

### Description

A button is placed in the ShowAllResults result value field which when clicked opens a dialog which shows all the results for this vision object.

### Remarks

The ShowAllResults result is a special type of results which allows the user to see all the results for a specific vision object. It is the most useful when there are multiple results for a specific vision object because you can see all the results at one time.

The ShowAllResults result was designed to make it easier to see multiple results all together in one place. Therefore, it is only available for those vision objects which support multiple results (ArcFinder, ArcInspector, Blob, ColorMatch, Correlation, DefectFinder, Edge, Geometric, LineFinder, CodeReader, and LineInspector OCR, BoxFinder, CornerFinder objects.)

### See Also

ArcFinder Object, ArcInspector Object, Blob Object, Correlation Object, DefectFinder Object, Edge Object, Geometric Object, LineFinder Object, LineInspector Object, CodeReader Object, OCR Object, BoxFinder Object, Contour Object, CornerFinder Object, Object Tab

# ShowConfirmation Property

## Applies To

Vision Calibration

## Description

Sets / returns whether to display a confirmation dialog at calibration.

## Usage

**VGet** *Calibration.ShowConfirmation, var*

**VSet** *Calibration.ShowConfirmation, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* String variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

True : Display

False : Not display

## Remarks

Sets / returns whether to display a confirmation dialog at runtime for VCal when the calibration cycle has completed.

Set this parameter to False if you do not want operators to verify the calibration results.

## See Also

VCal Statement, VCalPoints Statement, Vision Calibration

## ShowExtensions Property

### Applies To

Vision Objects: Line, LineFinder, ArcFinder

### Description

For Line objects, this property displays a line from a starting reference (defined by the StartPointObject property) to an ending reference (defined by the EndPointObject property). For LineFinder and ArcFinder objects, only found segments are displayed. The ShowExtensions property causes the graphics display of the line to be extended out (by using a dotted line to indicate the extensions) so you can see the complete projection of the line or arc.

### Usage

**VGet** *Sequence.Object.ShowExtensions*, *var*

**VSet** *Sequence.Object.ShowExtensions*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not show line extensions

1 – True Show line extensions

Default: 0 – False

### Remarks

When a Line object is created, the default graphic display of the Line object is simply a line with a starting and ending point. The StartPointObject and EndPointObject Properties can be used to modify the direction and length of the line but in some cases you may want to see where the line extends to. This is the purpose of the ShowExtensions property.

Extensions are useful when you need to see more than just a line between 2 points. For example, assume you create a Line object perpendicular to another line and the point at which the 2 lines meet is not actually on the physical line but at some location extended from the line. You can see this point of intersection by running your application with the ShowExtensions property set to 1–True.

### See Also

ArcFinder Object, EndPointObject Property, Line Object, LineFinder Object, StartPointObject Property, Frame Object, Object Tab

# ShowLabel Property

## Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Specifies whether to show a label for rendered character strings.

## Usage

**VGet** *Sequence.Object.ShowLabel*, *var*

**VSet** *Sequence.Object.ShowLabel*, *value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

## Values

False Do not show label.

True Show a label.

Default: True

## Remarks

You can switch the format of character strings rendered with the Text object. Set the ShowLabel property to True to add a label indicating the type of results as a prefix to character strings. Set this to True to distinguish between result types from character strings rendered in an image.

## See Also

Text Object, ResultObject Property, ResultText1-3 Property

# ShowModel Property

Design time only

## Applies To

Vision Objects: Contour, Correlation, DefectFinder, Geometric, Polar

## Description

The ShowModel property is used to check a previously taught model at various zoom settings. You can also change the model origin and don't care pixels for some models.

## Remarks

### Correlation and Geometric Objects:

You can zoom the model and also change the model origin with mouse or arrow keys and paint / erase "don't care" pixels.

The ShowModel property is available from the Vision Guide window Object tab.

The user clicks on the value field of the ShowModel property which causes a button to appear in the Value field. Click on the button and the model will be displayed in the Vision Guide window.

### Changing the model origin

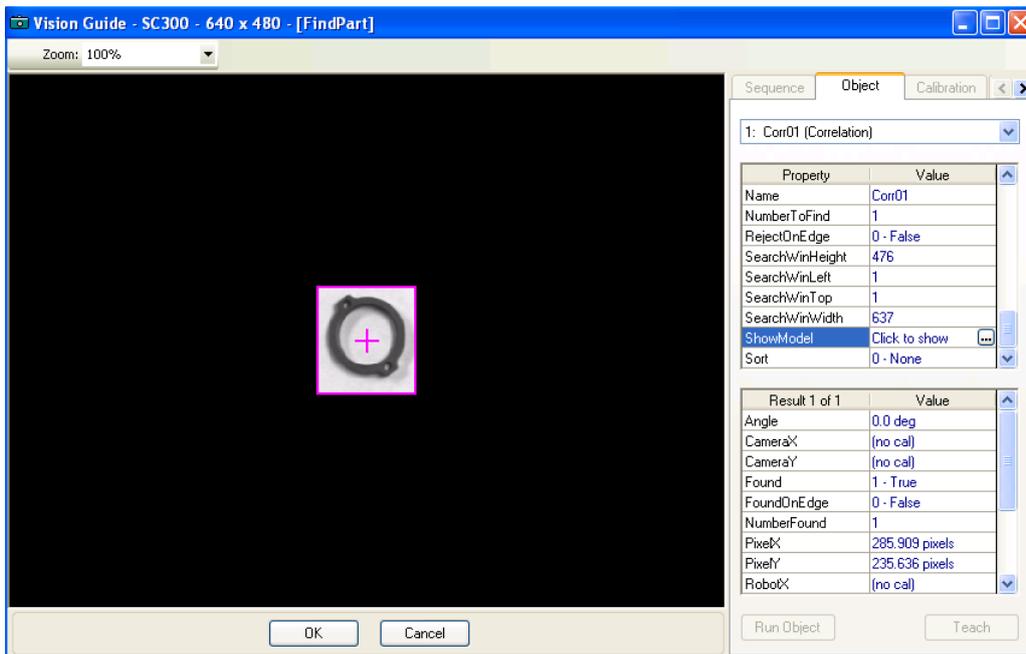
This allows you to change the model origin more accurately. A check box is provided so that ModelOrgAutoCenter can be toggled. When ModelOrgAutoCenter is set to on, the model origin is centered and cannot be changed.

To change the model origin with the mouse, make sure that the ModelOrgAutoCenter check box is off. Then point to the center of the cross, and click the mouse down. Now drag the cross to the desired position.

Click OK to save the new model origin settings.

### Polar and DefectFinder Objects:

The model can be zoomed. The ModelOrgAutoCenter check box and drawing toolbar are hidden, since they do not apply in this case. Also, there is a Close button instead of OK and Cancel.



## See Also

Contour Object, Correlation Object, Geometric Object, Object Tab, Polar Object, DefectFinder Object

# ShowProcessing Property

## Applies To

Vision Sequence

## Description

Determines whether image processing should be displayed when RunTimeFreeze is set to 1-True.

## Usage

**VGet** *Sequence.ShowProcessing*, *var*

**VSet** *Sequence.ShowProcessing*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – False Do not show processing.

1 – True Show processing

Default: 1 – True

## Remarks

Sometimes when using image processing objects such as ImageOp, it is desirable not to see the processing. For example, if you use an ImageOp to binarize an entire image before other objects execute, the display will show the binarized image if ShowProcessing is 1-True. By setting it to 0-False, only the object graphics are displayed without showing the image processing.

## See Also

RunTimeFreeze Property, Sequence Tab, Vision Sequences

## SizeToFind Property

### Applies To

Vision Objects: ArcInspector, Blob, Contour, DefectFinder, LineInspector

### Description

Selects which size of blobs to find.

### Usage

**VGet** *Sequence.Object.SizeToFind*, *var*

**VSet** *Sequence.Object.SizeToFind*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

- |              |  |
|--------------|--|
| 0 – Any      | Vision Constant: VISION_SIZETOFIND_ANY<br>Find blobs or defects of any size.       |
| 1 – Largest  | Vision Constant: VISION_SIZETOFIND_LARGEST<br>Find the largest blobs or defects.   |
| 2 – Smallest | Vision Constant: VISION_SIZETOFIND_SMALLEST<br>Find the smallest blobs or defects. |
| Default:     | 1 – Largest  |

### Remarks

Use the SizeToFind property to find the largest or smallest blobs or defects in a search area. When a blob object searches for blobs in an image, it finds several candidates; sometimes many more than the desired number. SizeToFind can filter the results so that you get the largest or smallest blobs.

### See Also

ArcInspector Object, Blob Object, Contour Object, DefectFinder Object, LineInspector Object, Sort Property

# SkewDirection Result

## Applies To

Vision Objects: Correlation, Geometric  
CV2 firmware Ver. 3.0.0.0 or later

## Description

Returns the skew direction of a detected object.

## Usage

**VGet** *Sequence.Object*.**SkewDirection**[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*result* Optional. Integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the CurrentResult.

## Values

-90 to 90 degrees

## Remarks

SkewDirection returns the skew direction when SkewFitEnable is set to True. If the property is set to False, it returns 0.0.

## See Also

Correlation Object, Geometric Object, SkewFitEnable Property, SkewRatio Result

## SkewFitEnable Property

### Applies To

Vision Objects: Correlation, Geometric  
CV2 firmware Ver. 3.0.0.0 or later

### Description

Sets whether or not to enable the skew corrected detection.

### Usage

**VGet** *Sequence.Object.SkewFitEnable*, *var*

**VSet** *Sequence.Object.SkewFitEnable*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable that will contain the value of the property.

*value* Boolean expression for the new value of the property.

### Values

0 – False Do not enable the skew corrected detection.

1 – True Enable the skew corrected detection.

Default: 0 - False

### Remarks

If it is set to True, the SkewFitEnable property detects an object while considering the skew direction and ratio. This is effective when the object may not be arranged at a position opposed to the camera.

### See Also

Correlation Object, Geometric Object, SkewDirection Result, SkewRatio Result

# SkewRatio Result

## Applies To

Vision Objects: Correlation, Geometric  
CV2 firmware Ver. 3.0.0.0 or later

## Description

Returns the skew ratio of the detected object.

## Usage

**VGet** *Sequence.Object.SkewRatio*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property.  
If omitted, the result number is the CurrentResult.

## Values

0 - 1

## Remarks

SkewRatio returns the skew ratio when SkewFitEnable is set to True. If the property is set to False, SkewRatio returns 1.0.

The skew ratio is a scale ratio between the skew direction returned by SkewDirection and the direction perpendicular to it.

If the object has no skew, the result is 1.0.

## See Also

Correlation Object, Geometric Object, SkewFitEnable Property, SkewDirection Result

# Smoothness Property

## Applies To

Vision Objects: Geometric

## Description

Sets / returns the smoothness level for the geometric edge extraction filter.

## Usage

**VGet** *Sequence.Object.Smoothness, var*

**VSet** *Sequence.Object.Smoothness, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - 100

Default: 50

## Remarks

The Smoothness property allows you to control the smoothing level of the edge extraction filter. The smoothing operation evens out rough edges and removes noise. The range of this control varies from 0 (no smooth) to 100 (a very strong smooth). The default setting is 50.

The DetailLevel property also affects how edges are extracted.

## See Also

DetailLevel Property, Geometric Object, Object Tab, Timeout Property

# Sort Property

## Applies To

Vision Objects: Blob, Contour, Correlation, DefectFinder, Geometric

## Description

Sets or returns the sort order used for the results of an object.

## Usage

**VGet** *Sequence.Object.Sort*, *var*

**VSet** *Sequence.Object.Sort*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – None	Vision Constant: VISION_SORT_NONE No sorting is done.
1 – PixelX	Vision Constant: VISION_SORT_PIXELX Results are ordered from left to right according to the PixelX result.
2 – PixelY	Vision Constant: VISION_SORT_PIXELY Results are ordered from top to bottom according to the PixelY result.
3 – PixelXY	Vision Constant: VISION_SORT_PIXELXY Results are ordered diagonally from upper left to lower right according to the PixelX and PixelY results.
4 – CameraX	Vision Constant: VISION_SORT_CAMERAX Results are ordered from left to right according to the CameraX result.
5 – CameraY	Vision Constant: VISION_SORT_CAMERAY Results are ordered from bottom to top according to the CameraY result.
6 – CameraXY	Vision Constant: VISION_SORT_CAMERAXY Results are ordered diagonally from lower left to upper right according to the CameraX and CameraY results.
7 – RobotX	Vision Constant: VISION_SORT_ROBOTX Results are ordered along the Robot's X axis according to the RobotX result.
8 – RobotY	Vision Constant: VISION_SORT_ROBOTY Results are ordered along the Robot's Y axis according to the RobotY result.
9 – RobotXY	Vision Constant: VISION_SORT_ROBOTXY Results are ordered diagonally according to the RobotX and RobotY results.
Default	0 - None

## Sort Property

---

### Remarks

The Sort property allows you to sort the results of an object so that you can retrieve the results in the desired order.

If you want to retrieve results in descending order, then reverse the order that you retrieve them. For example:

```
For i = numFound To 1 Step -1
    VGet seq1.blob01.RobotXYU(i), found(i), x(i), y(i), u(i)
Next i
```

### See Also

Blob Object, Contour Object, Correlation Object, Geometric Object, Object Tab

# StartPntObjResult Property

## Applies To

Vision Objects: Edge, Line, LineInspector, Contour

## Description

Specifies which result to use from the StartPointObject.

## Usage

**VGet** *Sequence.Object*. **StartPntObjResult**, *var*

**VSet** *Sequence.Object*. **StartPntObjResult**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

The value can be All or range from 1 to the NumberToFind value for the StartPointObject. If the StartPointObject is 'Screen', then the value is always 1.

## Remarks

StartPntObjResult enables you to attach several objects to the results of one StartPointObject. For example, you could create a blob object with NumberToFind set to 4. Then you could attach a line object to each one of the results by specifying the blob for the StartPointObject of each line and a different StartPntObjResult for each line. In addition, you can specify All. If both the StartPntObjResult and the EndPntObjResult properties are set to All, then the object will be executed for each result.

## See Also

Contour Object, Edge Object, EndPntObjResult Property, Line Object, LineInspector Object, Object Tab, StartPointObject Property

## StartPointObject Property

### Applies To

Vision Objects: Edge, Line, LineInspector, Contour

### Description

Specifies the vision object whose results will set the starting point of an object.

### Usage

**VGet** *Sequence.Object.StartPointObject*, *var*

**VSet** *Sequence.Object.StartPointObject*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property. Valid vision objects for the StartPointObject property are: Blob, Correlation, Edge, Geometric, Line, and Point objects.

### Values

Screen or any object that run prior to the Line object.

Default: Screen

### Remarks

When a Line object is first created, the StartPointObject property is set to Screen. However, Line objects are normally attached to other vision objects. This is the purpose of the StartPointObject and EndPointObject properties. Through these two properties the user can define a line between any two vision objects.

It is important to note that for each specific vision sequence, only those vision objects which are executed prior to the Line object in the vision sequence steps will be available to use as a StartPointObject.

### See Also

Contour Object, Edge Object, EndPointObject Property, Line Object, LineInspector Object, Object Tab, StartPointType Property

# StartPointType Property

## Applies To

Vision Objects: Contour, Edge, Line, LineInspector

## Description

Specifies the type of start point to use for an Edge, Line, or LineInspector object. In most cases the start point type will be a point (which usually means the PixelX and PixelY position of the StartPointObject). However, when the StartPointObject for the current line is a 2nd Line object, the StartPointType property is used to define an intersection point on the 2nd line such as the lines midpoint, endpoint, startpoint or perpendicular position.

## Usage

**VGet** *Sequence.Object.StartPointType*, *var*

**VSet** *Sequence.Object.StartPointType*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

StartPointObject = Line or LineFinder	StartPointObject = Screen, Blob, Correlation, Geometric, Edge, or Point object
See remarks. Default: 2 - MidPoint	0 - Point When used with objects other than the Line object, the StartPointType can only be of type 0 - Point. Default: 0 - Point

### Remarks

As you can see in the Values Table above, most of the StartPointObject's support only 1 StartPointType called 0-Point. This is because most StartPointObject's use the PixelX and PixelY position for a reference position for defining a start or end point for a line. So when the StartPointObject is defined as Screen, Blob, Correlation, Edge, or Point object the StartPointType will always be set to 0-Point.

The range of valid values for StartPointType depend on the StartPointObject.

However, when the StartPointObject is another Line object, or a LineFinder object, the user must decide where on the 2nd line to intersect with the 1st line. The choices are as follows:

- |                    |   |
|--------------------|---|
| 1 - EndPoint       | Use the end point of the other line as the endpoint for this line.  |
| 2 - MidPoint       | Cut the other line in half and use the center (or midpoint of the other line as the endpoint for this line.   |
| 3 - Perpendicular  | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion and use this position as the end point.   |
| 4 - StartPoint     | Use the starting point of the other line as the end point for this line.  |
| 5 - PerpToStartPnt | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the start point of the first line and use this position as the start point. |
| 6 - PerpToMidPnt   | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the mid point of the first line and use this position as the start point.   |
| 7 - PerpToEndPnt   | Calculate the position on the 2nd line where the 2 lines intersect in a perpendicular fashion through the end point of the first line and use this position as the start point.   |

If the StartPointObject is changed to a Line object or LineFinder object then the StartPointType is automatically changed to MidPoint.

If the StartPointObject is changed to Screen, Blob, Correlation, Edge, or Point object then the StartPointType is automatically changed to 0-Point.

### See Also

Contour Object, Edge Object, EndPointType Property, Line Object, LineFinder Object, LineInspector Object, Object Tab, StartPointObject Property

# Strength Result

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, CornerFinder, Edge, LineFinder, LineInspector

## Description

Returns the strength of the found edge.

## Usage

**VGet** *Sequence.Object.Strength*[(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the result.

*result* Optional integer result number from 1 to the NumberOfResults property. If omitted, the result number is the CurrentResult.

## Values

0 - 100%

## Remarks

Use the Strength result to help determine the value to use for the StrengthTarget property. For ArcFinder, ArcInspector, LineFinder, LineInspector, BoxFinder, CornerFinder objects, the Strength result is the average strength of all edges used in the search.

## See Also

ArcFinder Object, ArcInspector Object, Edge Object, LineFinder Object, LineInspector Object, BoxFinder Object, CornerFinder Object, StrengthTarget Property, StrengthVariation Property

# StrengthTarget Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

## Description

Sets the desired edge strength to search for.

## Usage

**VGet** *Sequence.Object.StrengthTarget*, *var*

**VSet** *Sequence.Object.StrengthTarget*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - 100%

Default: 0 (search for best strength)

## Remarks

An edge's strength is the minimum/maximum edge value along the width of the edge (depending on polarity). This value is a normalized percentage of the maximum pixel value.

Use StrengthTarget to find edges with lower strengths. First, find the edge you want to search for a record the Strength result value. Next, set the StrengthTarget property to this value. Then set the ScoreWeightStrength to a higher value than ScoreWeightContrast. This tells the Edge object to look for an edge with the desired strength and base the score on it.

## See Also

ArcFinder Object, ArcInspector Object, Edge Object, LineFinder Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder, Strength Result, StrengthVariation Property

# StrengthVariation Property

## Applies To

Vision Objects: ArcFinder, ArcInspector, BoxFinder, Contour, CornerFinder, Edge, LineFinder, LineInspector

## Description

StrengthVariation is the tolerance for the StrengthTarget property.

## Usage

**VGet** *Sequence.Object.StrengthVariation*, *var*

**VSet** *Sequence.Object.StrengthVariation*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - 100%

Default: 0

## Remarks

Use StrengthVariation to tighten the search for the edge with strength of StrengthTarget.

## See Also

ArcFinder Object, ArcInspector Object, Edge Object, LineFinder Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object, Strength Result, StrengthTarget Property

## StrobeBlackVideo Property

### Applies To

Vision Sequence

### Description

Determines whether video is cleared to black before the trigger is received.

### Usage

**VGet** *Sequence*.**StrobeBlackVideo**, *var*

**VSet** *Sequence*.**StrobeBlackVideo**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Boolean variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

0 – False Do not show black video before trigger.

1 – True Show black video before trigger.

Default: 1 – True

### Remarks

When RuntimeAcquire is set to Strobed, and VRun executes, by default the video display is cleared to black until a trigger is received. Then the grabbed image is displayed. For some applications, where VRun is executed rapidly in a loop, (such as conveyor tracking), it will be easier to see vision results if the video is not cleared to black after each VRun.

### See Also

RuntimeAcquire Property

# StrobeDelay Property

## Applies To

Vision Sequence

## Description

Sets / returns the delay time between receiving the hardware trigger signal and turning on the camera's strobe lamp output.

## Usage

**VGet** *Sequence.StrobeDelay*, *var*

**VSet** *Sequence.StrobeDelay*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

## Values

Long value in microseconds.

Default: 0 (microsecond)

## Remarks

Use StrobeDelay to set the time delay between the hardware trigger signal and turning on the strobe lamp output.

## See Also

RuntimeAcquire Property, ExposureTime Property, ExposureDelay Property, StrobeTime Property

## StrobeTime Property

### Applies To

Vision Sequence

### Description

Sets / returns the amount of time the camera's strobe lamp output is turned on during image acquisition.

### Usage

**VGet** *Sequence.StrobeTime*, *var*

**VSet** *Sequence.StrobeTime*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

### Values

Long value in microseconds.

Default: 0 (microsecond)

### Remarks

Use **StrobeTime** to set the amount of time that the camera's strobe lamp output is turned on during image acquisition.

### See Also

RuntimeAcquire Property, ExposureDelay Property, ExposureTime Property, StrobeDelay Property

# TargetSequence Property

## Applies To

Vision Calibration

## Description

Specifies the vision sequence that is used to find the calibration target(s) during calibration.

## Usage

**VGet** *Calibration.TargetSequence, var*

**VSet** *Calibration.TargetSequence, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

String value containing the name of the vision sequence

Default: None

## Remarks

The TargetSequence property must be specified for all calibrations. For more details, see the chapter *Calibration* in the Vision Guide manual.

## See Also

UpwardSequence Property, ReferenceType Property

## Text Result

### Applies To

Vision Objects: CodeReader, OCR

### Description

Returns the text found in a search operation.

### Usage

**VGet** *Sequence.Object.Text* [(*result*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the result.

### Values

String.

### Remarks

The Text result returns the text found by an OCR or CodeReader object. Invalid characters will be substituted with the character specified the the InvalidChar property.

### See Also

CodeReader Object, Found Result, InvalidChar Property, OCR Object, Score Result

# TextBackColor Property

## Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

## Description

Sets the background color for text.

## Usage

**VGet** *Sequence.Object.TextBackColor*, *var*

**VSet** *Sequence.Object.TextBackColor*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

Character string showing the background color of the label

Default: "Transparent"

## Remarks

Depending on the input image, color and shadows in the video image may reduce the legibility of character strings displayed. If this happens, use TextBackColor to make labels easier to read.

## See Also

PassColor Property, FailColor Property

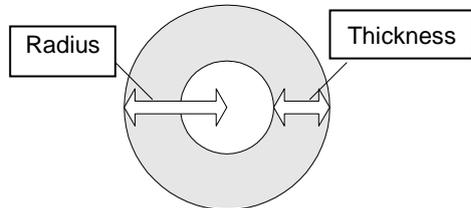
# Thickness Property

## Applies To

Vision Objects: Polar

## Description

Defines the thickness (in pixels) of the ring used for the Polar object.



## Usage

**VGet** *Sequence.Object.Thickness, var*

**VSet** *Sequence.Object.Thickness, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 - 200 pixels. The value cannot be greater than the Radius value.

Default: 5

## Remarks

It is important to remember that the Polar object is used to process images that are circular in nature. The Thickness property defines the thickness of the circular ring which is used to define the model and search window for the polar search.

In many cases the Thickness property does not need to be very large for a successful search. Since the Thickness property defines the Search Window size for the Polar object, keeping the Thickness small results in faster Polar search times.

## See Also

CenterPointObject Property, CenterX Property, CenterY Property, Object Tab, Polar Object, Radius Property

# ThresholdColor Property

## Applies To

Vision Objects: Blob, Contour, ImageOp

## Description

Sets or returns the color of the pixels whose gray values fall between the ThresholdHigh and ThresholdLow properties.

## Usage

**VGet** *Sequence.Object.ThresholdColor*, *var*

**VSet** *Sequence.Object.ThresholdColor*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 – Black (VISION\_THRESHCOLOR\_BLACK)

2 – White (VISION\_THRESHCOLOR\_WHITE)

Default: 1 – Black

## Remarks

The ThresholdColor property defines the color of the pixels whose gray values are between the ThresholdHigh and ThresholdLow properties during binarization. For example, when ThresholdColor = Black, ThresholdLow = 50 and ThresholdHigh = 100, then pixels whose gray values are between 50 and 100 will be set to black during binarization. All other pixels will be white.

## See Also

Blob Object, Contour Object, ImageOp Object, Object Tab, Polarity Property, ThresholdHigh, ThresholdLow Property

# ThresholdAuto Property

## Applies To

Vision Objects: Blob, Contour, ImageOp

## Description

Automatically sets or returns values of ThresholdHigh and ThresholdLow properties for Blob and ImageOp objects.

## Usage

**VGet** *Sequence.Object.ThresholdAuto*, *var*

**VSet** *Sequence.Object.ThresholdAuto*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – False Values of the ThresholdHigh and ThresholdLow properties are not set automatically.

1 – True Values of the ThresholdHigh and ThresholdLow properties are set automatically.

Default: 0 – False

## Remarks

The ThresholdAuto property automatically sets the values of the ThresholdHigh and ThresholdLow properties.

The ThresholdHigh property works with the ThresholdLow property to define the grey level regions which represent the feature (or object), the background and the edges of the image. The ThresholdHigh property defines the upper bound for grey level values that are considered to be a blob. Any part of the image which falls within grey level region between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is a blob.)

If the Polarity property is set to 1–DarkOnLight, then grey levels between ThesholdLow and ThresholdHigh will be changed to black pixels and all other pixels will be white.

If the Polarity property is set to 2–LightOnDark, then grey levels between ThesholdLow and ThresholdHigh will be changed to white pixels and all other pixels will be black.

One of the problems regarding the ThresholdLow and ThresholdHigh Properties is finding the correct values to use for each. This is where the Histogram feature of Vision Guide comes in. You can run a Histogram on an image to see the relationship between the pixel counts at various grey levels. From the Histogram dialog, you can adjust each of the threshold values and view the results.

### Note:

If ThreshouldAuto is set to “True”, the setting value declines to the threshold value which is capable to detect blobs (at least one blob can be detected) even when the image is homogeneous (all black or white), such as when the target work piece is unable to be captured.

## See Also

Blob Object, Contour Object, ImageOp Object, Polarity Property, ThresholdLow Property, Object Tab

# ThresholdHigh Property

## Applies To

Vision Objects: Blob, Contour, ImageOp, DefectFinder

## Description

Sets or returns the ThresholdHigh value for a Blob or ImageOp object.

## Usage

**VGet** *Sequence.Object.ThresholdHigh*, *var*

**VSet** *Sequence.Object.ThresholdHigh*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

1 - 255 This value must be greater than the ThresholdLow value or an error will occur.

Default: 128

## Remarks

The ThresholdHigh property works with the ThresholdLow property to define the grey level regions which represent the feature (or object), the background and the edges of the image. The ThresholdHigh property defines the upper bound for grey level values that are considered to be a blob. Any part of the image which falls within grey level region between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is a blob.)

If the Polarity property is set to 1–DarkOnLight, then grey levels between ThesholdLow and ThresholdHigh will be changed to black pixels and all other pixels will be white.

If the Polarity property is set to 2–LightOnDark, then grey levels between ThesholdLow and ThresholdHigh will be changed to white pixels and all other pixels will be black.

One of the problems regarding the ThresholdLow and ThresholdHigh Properties is finding the correct values to use for each. This is where the Histogram feature of Vision Guide comes in. You can run a Histogram on an image to see the relationship between the pixel counts at various grey levels. From the Histogram dialog, you can adjust each of the threshold values and view the results.

## See Also

Blob Object, Contour Object, DefectFinder Object, ImageOp Object, Object Tab, Polarity Property, ThresholdLow Property

# ThresholdLow Property

## Applies To

Vision Objects: Blob, Contour, ImageOp, DefectFinder

## Description

Sets or returns the ThresholdLow value for a Blob, DefectFinder, or ImageOp object.

## Usage

**VGet** *Sequence.Object.ThresholdLow*, *var*

**VSet** *Sequence.Object.ThresholdLow*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 - 254 This value must be less than the ThresholdHigh value or an error will occur.

Default: 0

## Remarks

The ThresholdLow property works with the ThresholdHigh property to define the grey level regions which represent the feature (or object), the background and the edges of the image. The ThresholdLow property defines the lower bound for grey level values that are considered to be a blob. Any part of the image which falls within the grey level region between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is a blob.)

If the Polarity property is set to DarkOnLight, then grey levels between ThesholdLow and ThresholdHigh will changed to black pixels and all other pixels will be white.

If the Polarity property is set to LightOnDark, then grey levels between ThesholdLow and ThresholdHigh will changed to white pixels and all other pixels will be black.

One of the problems regarding the ThresholdLow and ThresholdHigh Properties is finding the correct values to use for each. This is where the Histogram feature of Vision Guide comes in. You can run a Histogram on an image to see the relationship between the pixel counts at various grey levels. From the Histogram dialog, you can adjust each of the threshold values and view the results.

## See Also

Blob Object, Contour Object, DefectFinder Object, ImageOp Object, Object Tab, Polarity Property, ThresholdHigh Property

# Time Result

## Applies To

Vision Sequence

Vision Objects: ArcFinder, ArcInspector, Blob, BoxFinder, Contour, CodeReader, CornerFinder, Correlation, DefectFinder, Edge, Geometric, ImageOp, LineFinder, LineInspector, OCR, Polar

## Description

Returns the amount of time (in milliseconds) required to process the associated vision object or vision sequence.

## Usage

**VGet** *Sequence.Object.Time*, *var*

**VGet** *Sequence.Time*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long variable that will contain the value of the result.

## Values

0 - 2147483647 milliseconds (positive long integer)

## Remarks

The Time result is used to store how quickly a vision object or vision sequence is able to execute. (i.e. how quickly an object is found.)

The Time result is shown for each vision object (Blob, Correlation, Geometric, Edge, and Polar) as well as for an entire vision sequence.

For the sequence time result: if the RuntimeAcquire property is set to 1 - Stationary (default), then the total time includes the image acquisition time plus the total time for all steps in the sequence. The acquisition time can vary and depends on the time it takes for the vision system to synchronize with the camera.

For objects that return multiple results, the time returned is the total time to find all results.

## Statistics

For the Time result, the following statistics are available. TimeMax, TimeMean, TimeMin, TimeStdDev. Please see *Statistics* in the Vision Guide manual for details about using statistics.

## See Also

ArcFinder Object, ArcInspector Object, Blob Object, CodeReader Object, Correlation Object, DefectFinder Object, Edge Object, Geometric Object, ImageOp Object, LineFinder Object, LineInspector Object, BoxFinder Object, Contour Object, CornerFinder Object, Object Tab, OCR Object, Polar Object, Sequence Tab, Vision Sequences

## TimedOut Result

### Applies To

Vision Object: Correlation, Geometric

### Description

Returns whether the object search timed out or not.

### Usage

**VGet** *Sequence.Object.TimedOut*, *var*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Boolean variable representing the result value

### Values

True : Timeout occurred

False : No timeout occurred

### Remarks

Use the Timeout property to set the maximum search time. If the Timeout value is exceeded, then the TimedOut result will be True. If a timeout occurs, any accumulated results are still available.

### See Also

Correlation Object, Geometric Object, Timeout Property

# Timeout Property

## Applies To

Vision Objects: Correlation, Geometric

## Description

Sets / returns the maximum search time for a Correlation or Geometric object.

## Usage

**VGet** *Sequence.Object.Timeout*, *var*

**VSet** *Sequence.Object.Timeout*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Long integer variable that will contain the value of the property.

*value* Long integer expression for the new value of the property.

## Values

0 - 1000000

Default: 2000

## Remarks

Use the Timeout property to limit the amount of search time for a Correlation or Geometric object.

## See Also

Correlation Object, Geometric Object, Object Tab

## TotalArea Result

### Applies To

Vision Objects: ArcInspector, Blob, DefectFinder, LineInspector

### Description

Returns the sum of areas of all results.

### Usage

**VGet** *Sequence.Object.TotalArea*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the result.

### Values

Real number from 1- total pixels in image.

### Remarks

TotalArea can be used to determine all pixels in the image being searched whose blobs fall within the MinArea and MaxArea properties. By setting NumberToFind to 0 and MinArea to 1, the Blob object can be used as a pixel counter.

### See Also

ArcInspector Object, Area Result, Blob Object, DefectFinder Object, LineInspector Object, Object Tab, NumberToFind Property

# TriggerDebounce Property

## Applies To

Vision Sequence

## Description

Sets / returns hardware strobe trigger debounce value.

## Usage

**VGet** *Sequence*.**TriggerDebounce**, *var*

**VSet** *Sequence*. **TriggerDebounce**, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Long variable that will contain the value of the property.

*value* Long expression for the new value of the property.

## Values

Long value in microseconds.

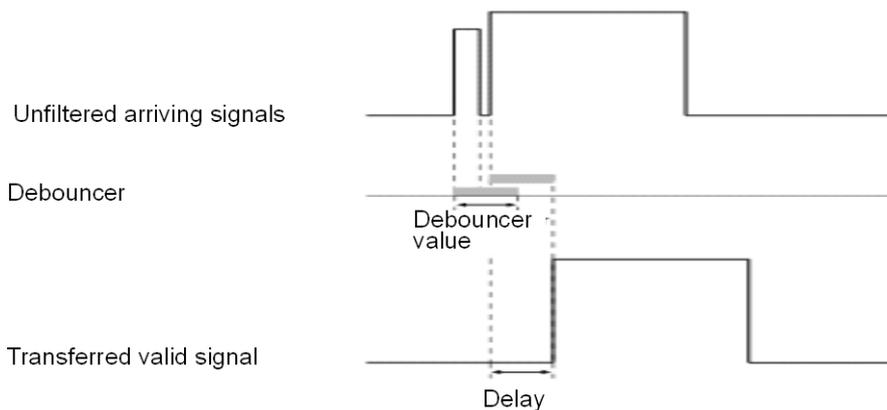
Range: 0 – 20,000 microseconds

Default: 0

## Remarks

This property is only available for the GigE cameras.

Use TriggerDebounce to filter noise on the strobe trigger signal. When the TriggerDebounce value is greater than zero, a new trigger signal is not accepted until the amount of debounce time has passed.



## Note

This property is only available for the GigE cameras and has no effect on the USB cameras.

## See Also

RuntimeAcquire Property, ExposureTime Property, ExposureDelay Property, StrobeTime Property, TriggerMode Property

## TriggerMode Property

### Applies To

Vision Sequence

### Description

Specifies the type of trigger signal transition used for electronic shutter image acquisition.

### Usage

**VGet** *Sequence.TriggerMode*, *var*

**VSet** *Sequence.TriggerMode*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value

1 – Leading Vision Constant: VISION\_TRIGGERMODE\_LEADINGEDGE  
The electronic shutter will open when the trigger signal transitions from low to high.

2 – Trailing Vision Constant: VISION\_TRIGGERMODE\_TRAILINGEDGE  
The electronic shutter will open when the trigger signal transitions from high to low.

Default: 1 – Leading

### Remarks

The TriggerMode property allows you to match the camera trigger signal transition according to the circuit you are using.

### See Also

RuntimeAcquire Property, Sequence Tab

# TwoRefPoints Property

## Applies To

Vision Calibration

## Description

Sets / returns whether a calibration should use two reference points instead of one.

## Usage

**VGet** *Calibration.TwoRefPoints, var*

**VSet** *Calibration.TwoRefPoints, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Boolean variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

0 – False Use one reference point

1 – True Use two reference points

Default: 0 – False

## Remarks

Set TwoRefPoints to 1–True to use two reference points for a calibration.

When CameraOrientation is FixedDownward or MobileJ2,J4,J5,J6, and ReferenceType is TaughtPoints, then you will be prompted to teach the reference point, rotate U 180 degrees, then teach the point again.

When CameraOrientation is FixedUpward, during the calibration cycle the system will search for the target, rotate 180 degrees, and search for the target again to calculate the midpoint of the two searches.

## See Also

CameraOrientation Property, ReferenceType Property

# Type Property

Runtime only

## Applies To

Vision Objects: All Objects

CV2 firmware Ver. 2.3.2.6 or later

## Description

Returns the type of a vision object.

## Usage

**VGet** *Sequence.Object.Type*, *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

## Values

The returned values are shown in the following table:

Object Type	Constant	Value
Correlation	VISION_OBJTYPE_CORRELATION	1
Blob	VISION_OBJTYPE_BLOB	2
Edge	VISION_OBJTYPE_EDGE	3
Polar	VISION_OBJTYPE_POLAR	4
Line	VISION_OBJTYPE_LINE	5
Point	VISION_OBJTYPE_POINT	6
Frame	VISION_OBJTYPE_FRAME	7
ImageOp	VISION_OBJTYPE_IMAGEOP	8
Ocr	VISION_OBJTYPE_OCR	9
CodeReader	VISION_OBJTYPE_CODEREADER	10
Geometric	VISION_OBJTYPE_GEOMETRIC	11
ColorMatch	VISION_OBJTYPE_COLORMATCH	14
LineFinder	VISION_OBJTYPE_LINEFINDER	15
LineInspector	VISION_OBJTYPE_ARCFINDER	16
ArcFinder	VISION_OBJTYPE_DEFECTFINDER	17
DefectFinder	VISION_OBJTYPE_LINEINSPECTOR	18
ArcInspector	VISION_OBJTYPE_ARCINSPECTOR	19
BoxFinder	VISION_OBJTYPE_BOXFINDER	20
CornerFinder	VISION_OBJTYPE_CORNERFINDER	21
Contour	VISION_OBJTYPE_CONTOUR	22
Text	VISION_OBJTYPE_TEXT	23

## Remarks

Use the Type property to determine the type of an object at runtime. This is useful for generic functions such as for data logging that need the object type in order to know which properties and / or results to access.

**Example**

```
Integer i, count, objType, score
Real area
VGet seq1.Objects.Count, count
For i = 1 To count
    VGet seq1.Objects(i).Type, objType
    Select objType
        Case VISION_OBJTYPE_CORRELATION
            VGet seq1.Objects(i).Score, score
            ' log some data here
        Case VISION_OBJTYPE_BLOB
            VGet seq1.Objects(i).Area, area
            ' log some data here
    End Select
Send
Next i
```

**See Also**

[Objects Property](#)

## UPCExpansionEnabled Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to support supplemental data in a UPC-E barcode.

### Remarks

Default: False

### See Also

Object Tab, UPCOutputChecksum Property

## UPCOutputChecksum Property

Design time only

### Applies To

Vision Object: CodeReader

### Description

Sets whether to include the checksum value in the UPC barcode Text result.

### Remarks

Default: True

### See Also

Object Tab, UPCExpansionEnabled Property

## UpwardLamp Property

### Applies To

Vision Calibration

### Description

Sets / returns the I/O output bit used for the calibration upward camera lamp.

### Usage

**VGet** *Calibration.UpwardLamp*, *var*

**VSet** *Calibration.UpwardLamp*, *value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Integer value of a valid output bit.

Default: None

### Remarks

Use the UpwardLamp property to automatically turn on a camera lamp during the calibration cycle for a calibration using an upward camera to find the reference point.

### See Also

Lamp Property, LampDelay Property

# UpwardSequence Property

## Applies To

Vision Calibration

## Description

UpwardSequence specifies the sequence used by an upward looking camera for a mobile calibration target reference.

## Usage

**VGet** *Calibration.UpwardSequence, var*

**VSet** *Calibration.UpwardSequence, value*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

## Values

String that contains the vision sequence name.

Default: None

## See Also

ReferenceType Property, TargetSequence Property

## UserText Property

### Applies To

Vision Object: Text  
CV2 firmware Ver. 3.1.0.0 or later

### Description

Renders user-defined character strings.

### Usage

**VGet** *Sequence.Object.UserText, var*

**VSet** *Sequence.Object. UserText, value*

*Sequence* Name of a sequence or string variable

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property.

### Values

Default: Empty

### Remarks

Set the character string to render in the Text object. UserText content will appear at the start of the search window, with ResultText1 to 3 content appearing after this.

### See Also

Text Object, ResultText1-3 Property

# VCal Statement

## Applies To

Vision Calibration

CV1 firmware Ver. 2.1.0 or later

## Description

VCal enables you to run a vision calibration from a SPEL<sup>+</sup> program.

## Usage

**VCal** *Calibration*

*Calibration* Name of the Calibration to be calibrated.

## Remarks

The calibration definition must be set up from the Vision Guide window or created using VCreateCalibration before a calibration will run. Also, the calibration points must already be taught, or you can use a point file in the controller with the same name as the calibration that has the point data required for calibration. If EPSON RC+ sees this point file, it will use the points in the file. See the following example.

After executing a calibration with VCal, you must call VSave to save the new calibration data.

### Note:

When the camera to be used for the calibration is connected to Compact Vision, EPSON RC+ needs to be connected to the robot controller. Otherwise, the command will result in an execution error.

## Example 1

In this example 1, disable the AutoReference property and enable the AutoCamPoints property to perform calibration of the mobile J6 camera.

```
Function CalMobileJ6
  String cal$

  cal$ = "mycal"
  VCreateCalibration 1, cal$
  VSet cal$.CameraOrientation, VISION_CAMORIENT_MOBILEJ6
  VSet cal$.TargetSequence, "calSeq"
  VSet cal$.AutoReference, False
  VSet cal$.AutoCamPoints, True
  VCalPoints cal$
  VCal cal$
  VSave
Fend
```

**Example 2**

In this example 2, use a recognition result of nine points using a calibrated mobile camera instead of implementing teaching to calibrate a fixed downward camera.

```
Function CalFixedCamera As Boolean
  Integer i
  Boolean found
  Real x, y, u
  String obj$
  ' "mobileCal" is a sequence with 9 blobs that uses a mobile calibration.
  ' First we search with the mobile camera
  Jump mobileCamView
  VRun mobileCal
  VGet mobileCal.AllFound, found
  If Not found Then
    MsgBox "Could not find all targets"
    Exit Function
  EndIf
  For i = 1 TO 9
    obj$ = "blob0" + Str$(i)
    VGet mobileCal.obj$.RobotXYU, found, x, y, u
    ' Save each target point in robot coordinates
    P(i) = XY(x, y, 0, 0)
  Next i
  ' Save the points for VCal to use
  ' Note that "fixed" is the name of the calibration
  SavePoints "fixed.pts"
  Jump clearFixed
  ' Calibrate the fixed camera calibration scheme
  VCal fixed
  CalFixedCamera = True
Fend
```

**See Also**

ShowConfirmation Property, VCalPoints, VCreateCalibration, Vision Sequences, VSave

# VCalPoints Statement

## Applies To

Vision Calibration

CV1 firmware Ver. 2.1.0 or later

## Description

VCalPoints enables you to teach points for a vision calibration from a SPEL<sup>+</sup> program.

## Usage

**VCalPoints** *Calibration*

*Calibration* Name of the Calibration to be calibrated.

## Remarks

When VCalPoints executes, a dialog is displayed that allows the user to teach the calibration points for the specified calibration definition.

After teaching calibration points with VCalPoints, you must call VSave to make the changes permanent.

### Note

When the camera to be used for the calibration is connected to Compact Vision, the EPSON RC+ needs to be connected to the robot controller. Otherwise, the command will result in an execution error.

## See Also

VCal, Vision Sequences, VSave

## VCLs Statement

Runtime only

### Applies To

The Graphics display

CV1 firmware Ver. 2.1.0 or later

### Description

VCLs clears the image display area of all graphics.

### Usage

**VCLs**

### Remarks

VCLs causes all graphics which were drawn during execution of a vision sequence to be removed from the image display area. This is most often used to remove screen clutter between vision sequences. For example, if your application uses vision in only one part of the application, you may want to clear the image display area while the robot is doing other parts of the application since the vision processing stage of the application is complete.

VCLs is available only from the SPEL<sup>+</sup> Language.

### See Also

VGet, VRun, VSet, Vision Sequences

# VCreateCalibration Statement

## Applies To

Vision Calibration

CV1 firmware Ver. 2.1.0 or later

## Description

VCreateCalibration creates a vision calibration at runtime.

## Usage

**VCreateCalibration** *CameraNumber, CalibrationName, [CopyCalibName]*

*CameraNumber* Numerical expression containing a camera number to be used for new calibration.

*CalibrationName* String expression containing new calibration name.

*CopyCalibName* Optional. String expression containing the name of copy source calibration

## Remarks

Here are the basic steps to create a runtime calibration:

1. Execute VCreateCalibration.
2. Use VSet to set CameraOrientation and TargetSequence properties. Set other properties as necessary.
3. If the camera is not Standalone, you must create a point file for the calibration using the same name, or call VCalPoints to teach calibration points.
4. Execute VCal to run the calibration.
5. Set the Calibration property for one or more sequences that will use the new calibration.
6. Execute VSave to save the changes.

To make changes to the vision configuration permanent, you must call VSave.

## Example

```
Function CreateCal
  String cal$

  cal$ = "mycal"
  VCreateCalibration 1, cal$
  VSet cal$.CameraOrientation, VISION_CAMORIENT_MOBILEJ2
  VSet cal$.TargetSequence, "calSeq"
  VCalPoints cal$
  VCal cal$
  VSave
Fend
```

## See Also

VCreateObject Statement, VCreateSequence Statement, VSave Statement

# VCreateObject Statement

## Applies To

Vision Sequence

CV1 firmware Ver. 2.1.0 or later

## Description

VCreateObject creates an object at runtime.

## Usage

**VCreateObject** *Sequence, ObjectName, ObjectType*

*Sequence* Name of a sequence or string expression containing a sequence name.

*ObjectName* String expression containing the name of an object to create in sequence *Sequence*.

*ObjectType* Integer expression for the vision object type.

Object Type	Constant	Value
Correlation	VISION_OBJTYPE_CORRELATION	1
Blob	VISION_OBJTYPE_BLOB	2
Edge	VISION_OBJTYPE_EDGE	3
Polar	VISION_OBJTYPE_POLAR	4
Line	VISION_OBJTYPE_LINE	5
Point	VISION_OBJTYPE_POINT	6
Frame	VISION_OBJTYPE_FRAME	7
ImageOp	VISION_OBJTYPE_IMAGEOP	8
Ocr	VISION_OBJTYPE_OCR	9
CodeReader	VISION_OBJTYPE_CODEREADER	10
Geometric	VISION_OBJTYPE_GEOMETRIC	11
ColorMatch	VISION_OBJTYPE_COLORMATCH	14
LineFinder	VISION_OBJTYPE_LINEFINDER	15
ArcFinder	VISION_OBJTYPE_ARCFINDER	16
DefectFinder	VISION_OBJTYPE_DEFECTFINDER	17
LineInspector	VISION_OBJTYPE_LINEINSPECTOR	18
ArcInspector	VISION_OBJTYPE_ARCINSPECTOR	19
BoxFinder	VISION_OBJTYPE_BOXFINDER	20
CornerFinder	VISION_OBJTYPE_CORNERFINDER	21
Contour	VISION_OBJTYPE_CONTOUR	22
Text	VISION_OBJTYPE_TEXT	23

## Remarks

Use VCreateObject to add an object to an existing vision sequence at runtime. Use VSave to save it after setting the properties.

## See Also

VCreateCalibration, VCreateSequence, VSave

# VCreateSequence Statement

## Applies To

Vision Sequence

CV1 firmware Ver. 2.1.0 or later

## Description

VCreateSequence creates a new vision sequence at runtime.

## Usage

**VCreateSequence** *CameraNumber, SequenceName, [CopySequenceName]*

*CameraNumber* Numerical expression containing a camera number to be used for a new sequence.

*SequenceName* String expression containing new sequence name.

*CopySequenceName* Optional. String expression containing the name of the copy source sequence.

## Remarks

Use VCreateSequence to create a new vision sequence at runtime. Use VCreateObject to add objects to the sequence. Use VSave to save it after setting the properties.

## See Also

VCreateCalibration, VCreateObject, VSave

# VDefArm Statement

## Applies To

Vision Calibration

## Description

VDefArm calculates an arm set value of the mobile J2 camera using a feature point detectable by the vision system.

## Usage

**VDefArm** *ArmNumber, ArmType, ArmsetMode, Sequence, [Rotation], [TargetTolerance], [RobotSpeed], [RobotAccel], [ShowWarning]*

<i>ArmNumber</i>	Integer variable that contains the arm number to perform arm set (1 to 15).
<i>ArmType</i>	Integer expression that contains the arm type. VISION_DEFARM_J2CAM: calculates a center of mobile J2 camera image.
<i>ArmsetMode</i>	Integer variable that contains the arm set mode. VISION_DEFARM_MODE_ROUGH: A mode to run a rough arm set. Although the robot motion is small, setting accuracy will be about 1 mm. VISION_DEFARM_MODE_FINE: A mode to run a fine arm set. By operating the robot largely while changing the hand orientation, it realizes arm setting with a higher accuracy.
<i>Sequence</i>	Name of a sequence or string expression containing a sequence name
<i>Rotation</i>	Real variable displaying the angle of rotation (degrees) for approximate arm setting Value range: 0 to 45 Default: 5 degrees When set to "0", or when left unspecified, this value will be set to "5".
<i>TargetTolerance</i>	Real variable displaying the pixel distance in which vision detection results are considered a match with the target position Value range: 0 to 3 pixels Default: 1 When set to "0", or when left unspecified, this value will be set to "1".
<i>RobotSpeed</i>	Integer variable displaying the speed of the robot (%) Value range: 0 to 100 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".
<i>RobotAccel</i>	Integer variable displaying the acceleration of the robot (%) Value range: 0 to 99 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".
<i>ShowWarning</i>	Integer variable that determines whether to display warning when <i>ArmSetMode</i> is Fine. 0 - Always display 1 - Display when <i>RobotSpeed</i> and <i>RobotAccel</i> are larger than the default value. -1 - Do not display If omitted, "1 - Display when <i>RobotSpeed</i> and <i>RobotAccel</i> are larger than the default value" is set.

**Remarks**

VDefArm calculates an arm set value of the mobile J2 camera using a feature point detectable by the vision system.

Set a horizontal distance from J2 to the center of orientation and offset angle of J2. Default value is set to other parameters.

**See Also**

VDefGetMotionRange Statement, VDefLocal Statement, VDefSetMotionRange Statement, VDefTool Statement VGoCenter Statement

## VDefGetMotionRange Statement

### Applies To

Vision Calibration

### Description

VDefGetMotionRange acquires values of the motion range limited by VDefTool, VDefArm, VDefLocal, and VGoCenter.

### Usage

**VDefGetMotionRange** *MaxMoveDist*, [*MaxPoseDiffAngle*],[ *LjmMode*]

<i>MaxMoveDist</i>	Real variable that contains the maximum distance of move. If 0 is specified, the range is not limited. ( 0 to 500. Default: 200) VDefTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
<i>MaxPoseDiffAngle</i>	Real variable that contains the maximum displacement angle (degrees) of tool orientation (UVW). If 0 is specified, the angle is not limited. It only affects VDefLocal. ( 0 to 180. Default: 45 degrees)
<i>LjmMode</i>	Integer variable that will contains the LJM mode.

### Remarks

VDefGetMotionRange limits the motion range by VDefTool, VDefArm, VDefLocal, and VGoCenter.

LJM mode controls the posture flag of the point data to prevent unintended rotation of the wrist. LJM mode vales are the same values as LJM function of SPEL+.

### See Also

VDefTool Statement, VDefArm Statement, VDefLocal Statement, VGoCenter Statement, VDefSetMotionRange Statement

# VDefLocal Statement

## Applies To

Vision Calibration

CV2 firmware Ver. 3.0.0.0 or later

## Description

VDefLocal detects a calibration plate placed on a work plane by a mobile camera, and defines local coordinates parallel to the work plane.

## Usage

**VDefLocal** *LocalNumber, LocalType, CalibPlateType, Sequence, [TargetTolerance], [CameraTooNo], [RefPointNo], [RobotSpeed], [RobotAccel]*

<i>LocalNumber</i>	Integer variable representing a tool number to set local coordinates (1-15)
<i>LocalType</i>	Integer variable representing a local type VISION_DEFLOCAL_J6CAM: Specifies local coordinates parallel to a calibration plate by using the mobile J6 camera.
<i>CalibPlateType</i>	Integer variable that will contains the type of calibration plate. VISION_CALIBPLATE_L: Large calibration plate VISION_CALIBPLATE_M: Medium calibration plate VISION_CALIBPLATE_S: Small calibration plate VISION_CALIBPLATE_XS: Extra small calibration plate
<i>Sequence</i>	String expression representing a vision sequence name of current project. When using the mobile camera, this is a vision sequence to detect a reference marker on the calibration plate. When using the fixed camera, this is a vision sequence to detect a feature point at tool end, such as user's workpiece.
<i>TargetTolerance</i>	Real variable containing a pixel distance to consider that the vision detection result matches the target position. Value range: 0 to 3 pixel Default: 1 If omitted or "0" is specified,, it is set to "1".
<i>CameraTooNo</i>	VISION_DEFLOCAL_J6CAM: If auto calibration has been executed, specify a tool number of mobile camera. To perform auto calibration, specify -1.
<i>RefPoint</i>	Specifies the point which a local plane parallel to a work plane passes. This point is used to specify the local plane height.
<i>RobotSpeed</i>	Integer variable displaying the speed of the robot (%) Value range: 0 to 100 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".
<i>RobotAccel</i>	Integer variable displaying the acceleration of the robot (%) Value range: 0 to 99 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".

## VDefLocal Statement

---

### Remarks

VDefLocal detects a calibration plate placed on a work plane by a mobile camera, and defines local coordinates parallel to the work plane.

### See Also

VDefArm Statement, VDefGetMotionRange Statement, VDefSetMotionRange Statement, VDefTool Statement, VGoCenter Statement

# VDefSetMotionRange Statement

## Applies To

Vision Calibration

## Description

VDefSetMotionRange limits a motion range by VDefTool, VDefArm, VDefLocal, and VGoCenter.

## Usage

**VDefSetMotionRange** *MaxMoveDist*, [*MaxPoseDiffAngle*],[*LjmMode*]

<i>MaxMoveDist</i>	Real value representing the maximum distance of move. If 0 is specified, the range is not limited. (0 to 500. Default: 200) VDefTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
<i>MaxPoseDiffAngle</i>	Real value representing the maximum displacement angle (degrees) of tool orientation (UVW). If 0 is specified, the angle is not limited. It only affects VDefLocal. ( 0 to 180. Default: 45 degrees)
<i>LjmMode</i>	Integer variable that will contains the LJM mode.

## Remarks

VDefSetMotionRange limits a motion range by VDefTool, VDefArm, VDefLocal, and VGoCenter.

LJM mode controls the posture flag of the point data to prevent unintended rotation of the wrist. LJM mode vales are the same values as LJM function of SPEL+.

## See Also

VDefTool Statement, VDefArm Statement, VDefLocal Statement, VGoCenter Statement, VDefGetMotionRange Statement

# VDefTool Statement

## Applies To

Vision Calibration

## Description

VDefTool uses vision detection to calculate a tool offset value for TCP and mobile camera position.

## Usage

**VDefTool** *ToolNumber, ToolDefType, Sequence,*  
*[FinalAngle],[InitialAngle],[TargetTolerance],[RobotSpeed],[RobotAccel]*

**VDefTool** *ToolNumber, VISION\_DEFTOOL\_FIXEDWITHCAL, Sequence.Object,*  
*[FinalAngle],[InitialAngle],[TargetTolerance],[RobotSpeed],[RobotAccel]*

<i>ToolNumber</i>	Integer variable representing a tool number to perform tool set. (1-15)
<i>ToolDefType</i>	Integer variable representing a tool type. VISION_DEFTOOL_FIXEDNOCAL: Tool set by using the fixed camera which is not calibrated. VISION_DEFTOOL_J4CAM: Calculates image center of the mobile J4 camera. VISION_DEFTOOL_J6CAM: Calculates image center of the mobile J6 camera. VISION_DEFTOOL_FIXEDWITHCAL: Sets tools using a calibrated upward camera.
<i>Sequence</i>	Name of a sequence or string expression containing a sequence name.
<i>FinalAngle</i>	Integer variable containing an angle (degrees) to rotate the tool or camera tool. Real variable displaying the degree of rotation of the tool and camera tool (degree) Value range: 5 to 180, -5 to -180[degrees] Default: 90 When specifying positive value, rotate to +U-axis direction of the tool coordinate system. When specifying negative value, rotate to -U-axis direction of the tool coordinate system. If omitted or "0" is specified, it is set to "90".
<i>InitialAngle</i>	Real variable containing an angle (degrees) to rotate the tool or camera tool in provisional tool setting. Value range: -10 to 10[degrees] Default: 5 When specifying positive value, rotate to +U-axis direction of the tool coordinate system. When specifying negative value, rotate to -U-axis direction of the tool coordinate system. If omitted or "0" is specified, it is set to "5".  Absolute value of this value must be smaller than that of <i>FinalAngle</i> .
<i>TargetTolerance</i>	Real variable containing a pixel distance to consider that the vision detection result matches the target position. Value range: 0 to 3 pixel Default: 1 If omitted or "0" is specified, it is set to "1".
<i>RobotSpeed</i>	Integer variable displaying the speed of the robot (%) Value range: 0 to 100 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".

<i>RobotAccel</i>	Integer variable displaying the acceleration of the robot (%) Value range: 0 to 99 Default: 5 When set to “0”, or when left unspecified, this value will be set to “5”.
<i>Object</i>	Name of an object or string expression containing the object name

#### Remarks

VDefTool uses vision detection to calculate a tool offset value for TCP and mobile camera position.

If the tool type is the fixed camera, X and Y are set as tool offset of TCP. At this time, 0 is set for Z, U, V, and W. If the tool type is mobile J4 camera and J6 camera, X, Y, and U are set as tool offset for installation position of the mobile camera. At this time, 0 is set for Z, V, and W.

#### See Also

VDefArm Statement, VDefGetMotionRange Statement, VDefSetMotionRange Statement, VDefTool Statement, VGoCenter Statement

## VDeleteCalibration Statement

### Applies To

Vision Calibration

CV1 firmware Ver. 2.1.0 or later

### Description

VDeleteCalibration deletes a vision calibration at runtime.

### Usage

**VDeleteCalibration** *CalibrationName*

*CalibrationName* Calibration name or string expression that contains the calibration name.

### Remarks

Use VDeleteCalibration to delete a vision calibration at runtime. If the calibration does not exist, no error occurs. Use VSave to save vision settings after deleting the calibration.

### See Also

VCreateCalibration, VDeleteObject, VDeleteSequence, VSave

### Example

```
VDeleteCalibration "mycal"
```

# VDeleteObject Statement

## Applies To

Vision Sequence

CV1 firmware Ver. 2.1.0 or later

## Description

VDeleteObject deletes a vision object at runtime.

## Usage

**VDeleteObject** *Sequence, ObjectName*

*Sequence*                      Name of a sequence or string expression containing a sequence name.

*ObjectName*                      String expression containing the name of an object to delete in sequence *Sequence*.

## Remarks

Use VDeleteObject to delete a vision object at runtime. If the object does not exist, no error occurs. Use VSave to save vision settings after deleting the object.

## See Also

VCreateObject, VDeleteCalibration, VDeleteSequence, VSave

## Example

```
VDeleteObject "myseq", "blob01"
```

## VDeleteSequence Statement

### Applies To

Vision Sequence

CV1 firmware Ver. 2.1.0 or later

### Description

VDeleteSequence deletes a vision sequence at runtime.

### Usage

**VDeleteSequence** *SequenceName*

*SequenceName*      Sequence name or string variable that contains the sequence name.

### Remarks

Use VDeleteSequence to delete a vision sequence at runtime. If the sequence does not exist, no error occurs. Use VSave to save vision settings after deleting the sequence .

### See Also

VCreateSequence, VDeleteCalibration, VDeleteObject, VSave

### Example

```
VDeleteSequence "myseq"
```

# VGet Statement

## Applies To

- Vision Sequence
- Vision Calibration
- Vision Objects: All

## Description

VGet is used to get the values of properties and results in SPEL+ and RC+ API.

## Usage

**VGet** *Sequence* .*Property*, *var*

**VGet** *Calibration* .*Property*, *var*

**VGet** *Sequence* .***Object*** .*Property*, *var*

**VGet** *Sequence* .***Object*** .*Result*[(*resultIndex*)], *var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Calibration* Name of a calibration or string variable containing a calibration name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence. Omit *Object* if you are retrieving a sequence or calibration property or result.

*Property* Name of the property to set or return the value of.

*Result* Name of the result to get the value of. You can optionally specify a *resultIndex* for objects that can return more than one set of result data, such as Blob and Correlation. This allows you to obtain a particular result without setting the CurrentResult property.

*var* Variable(s) that will contain the value(s) returned.

### Remarks

VGet is a powerful part of the Vision Guide structure. It provides the core mechanism to get the property and result values from the vision objects that are run from vision sequences.

VGet can be used to get property values before running a vision sequence so that you can check the value of a specific property or even check and set it by using the VGet statement and then the VSet statement. VGet can also be used to get property values after running a vision sequence

The most common use for VGet is to get the result values from vision objects after they have been run in a sequence. This allows you to use the results to make decisions, perform calculations, define point positions and a whole host of other things. In order to use VGet with results, you must first VRun the sequence which contains the vision object for which you want to get a result from. For example, assume you created a vision sequence which uses a Blob object to find how many holes are present in a specific part. This means you will want to VGet the value of the Holes result for the Blob object. The following SPEL<sup>+</sup> program shows how VGet would be used in this instance.

```
Function test
'It is assumed that a sequence called FindHoles has already been created
'prior to running this program. FindHoles contains a Blob object called Part
'which is configured to find how many holes are in the search window.
'In this example, we will run the sequence and then display the number
'of holes which were found.

Integer count

VRun FindHoles                'Run the vision sequence
VGet FindHoles.Part.Holes, count 'Get the # of holes found

Print count, "holes found"
Fend
```

### See Also

VRun, VSet, Vision Sequences

# VGoCenter Statement

## Applies To

Vision Calibration  
CV2 firmware Ver.3.0.0.0 or later

## Description

Using a feature point, VGoCenter moves the robot so that the feature point can be captured at the center of the camera image.

## Usage

**VGoCenter** *Sequence*, [*LocalNo*],[*TargetTolerance*],[*RobotSpeed*],[*RobotAccel*]

<i>Sequence</i>	Name of a sequence or string variable representing a sequence name.
<i>LocalNo</i>	Integer variable representing a local coordinate number for moving the robot (-1 to 15, Default: -1) Tool is moved in the XY plane of the specified local coordinate. If omitted, or set to -1, the robot will move on the XY plane of the tool orientation when a command is invoked.
<i>TargetTolerance</i>	Real variable containing a pixel distance to consider that the vision detection result matches the target position. Value range: 0 to 3 pixels. Default: 1 If omitted or "0" is specified, it is set to "1".
<i>RobotSpeed</i>	Integer variable displaying the speed of the robot (%) Value range: 0 to 100 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".
<i>RobotAccel</i>	Integer variable displaying the acceleration of the robot (%) Value range: 0 to 99 Default: 5 When set to "0", or when left unspecified, this value will be set to "5".

## Remarks

Using a feature point, VGoCenter moves the robot so that the feature point can be captured at the center of the camera image.

## See Also

VCal

## VLoad Statement

### Applies To

All Vision Properties

CV1 firmware Ver. 2.1.0 or later

### Description

Loads all vision properties for the current project from disk.

### Usage

**VLoad**

### Remarks

Use VLoad to restore all vision properties at runtime to design time values.

When VLoad executes, it loads the data from the .VIS file in the project directory.

#### Note:

When using a Compact Vision, EPSON RC+ needs to be connected to the robot controller. Otherwise, the command will result in an execution error.

### See Also

VSave

# VLoadModel Statement

## Applies To

Vision Objects: Contour, Correlation, Geometric, Polar, DefectFinder  
CV1 firmware Ver. 2.1.0 or later

## Description

Loads a model from a file on the controller that was created with VSaveModel.

## Usage

**VLoadModel** *Sequence.Object, fileName*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name.  
The object must exist in the specified sequence.

*fileName* Path and file name.

## Remarks

VLoadModel can be used in applications where you have several different part models but the same vision sequence can be used. The *fileName* parameter refers to a file that was previously saved with VSaveModel for the same type of vision object.

When you are using the Compact Vision, USB memory can be specified in *filename*. When using USB memory, specify *fileName* as CVUSB:\path\filename

VLoadModel can be used with the following vision objects:

- Correlation
- Geometric
- Polar
- DefectFinder

## Example

```
VLoadModel seq1.Corr01, "c:\models\corr01.mdl"
```

## See Also

VSaveModel

# VRun Statement

## Applies To

Vision Sequence

## Description

VRun is a SPEL<sup>+</sup> language statement used to initiate execution of vision sequences which were created in the Vision Guide development environment or at runtime using VCreateSequence.

## Usage

**VRun** *Sequence*

*Sequence* Name of a sequence or string expression containing a sequence name.

## Remarks

The VRun SPEL<sup>+</sup> Language statement initiates execution of vision sequences.

When VRun is initiated the vision sequence specified begins execution. First, an image is acquired (unless the user has set the RuntimeAcquire property to None) into the image buffer, and then vision objects are applied to that image as defined in the vision sequence.

It is important to note that when AsyncMode is True (default), VRun returns prior to completing execution of the vision sequence specified with VRun. Once the image is acquired, VRun returns control to the next SPEL<sup>+</sup> statement which follows VRun. This improves the throughput of the overall cycle time by allowing other SPEL<sup>+</sup> statements to execute while vision processing occurs. (For example, the robot can move during vision processing, or a calculation could be performed during this time.) When AsyncMode is False, VRun acquires the image (if necessary) and runs all objects before returning.

Once VRun is executed, VGet is normally used to get the results of the vision sequence such as part position data, good part bad part status, part count information or many other results. Shown below is a simple program which uses VRun and VGet to execute a vision sequence and then use results from that sequence to display useful information to the user.

Before running the program, create a sequence called "FindHoles" and a blob object called "Part".

```
Function test
'It is assumed that a sequence called FindHoles has already been created
'prior to running this program. FindHoles contains a Blob object called Part
'which is configured to find how many holes are in the search window.
'In this example, we will run the sequence and then display the number
'of holes which were found.

Integer count

VRun FindHoles                'Run the vision sequence
VGet FindHoles.Part.Holes, count 'Get the # of holes found

Print count, "holes found"
Fend
```

## See Also

VGet, VSet, Vision Sequences

## VSave Statement

### Applies To

All Vision Properties  
CV1 firmware Ver. 2.1.0 or later

### Description

Saves all vision properties for the current project to disk.

### Usage

**VSave**

### Remarks

Use **VSave** to make runtime changes to vision properties permanent.

When **VSave** executes, it updates the .VIS file in the project directory.

#### Note:

When using a Compact Vision, EPSON RC+ needs to be connected to the robot controller. Otherwise, the command will result in an execution error.

### See Also

VLoad, VSet

# VSaveImage Statement

## Applies To

Sequence

CV1 firmware Ver. 2.1.0 or later

## Description

Saves the current frame grabber image on a disk file.

## Usage

**VSaveImage** *Sequence*, *fileName*[, *saveGraphics*]

*Sequence* Name of a sequence or string variable containing a sequence name.

*fileName* Path and file name. The file extension must be BMP(Default format), TIF, or JPG.

*saveGraphics* Optional. Specify whether to save the image detection result with the graphics included.

## Remarks

VSaveImage can be used to save an image to a disk file at run time. This is particularly useful for analyzing images where parts were not found.

The image that is saved is the currently displayed image. The RuntimeFreeze property should be set to True for the sequence that you are saving the image from.

When you are using Compact Vision, USB memory plugged into the CV unit can be specified in *filename*. as CVUSB:\path\filename.

When *saveGraphics* is True, the image and sequence result graphics will be saved (Default: false).

## Example

```
VRun seq1
VGet seq1.AllFound, found
If found = False Then
    VSaveImage seq1, "c:\badimages\seq1.bmp"
EndIf
```

## See Also

ImageFile Property, SaveImage Property

# VSaveModel Statement

## Applies To

Vision Objects: Contour, Correlation, Geometric, Polar  
CV1 firmware Ver. 2.1.0 or later

## Description

Saves a model for a vision object on disk.

## Usage

**VSaveModel** *Sequence.Object, fileName*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*fileName* Path and file name excluding the extension.

## Remarks

VSaveModel can be used in applications when you have several different part models. Normally, a model for each object is stored in the Vision Guide data files. VSaveModel allows you to save the models into specified files so you can load them into other objects of the same type.

When you are using the Compact Vision, USB memory can be specified in *filename* as CVUSB:\path\filename.

VSaveModel can be used with the following vision objects:

- Correlation
- Geometric
- Polar
- DefectFinder

## Example

```
Integer status

VTeach seq1.corr01, status
If status = 1 Then
    VSaveModel seq1.corr01, "c:\models\corr01.mdl"
EndIf
```

## See Also

VLoadModel

# VSet Statement

## Applies To

- Vision Sequence
- Vision Calibration
- Vision Objects: All

## Description

VSet is used to set the values of properties from the SPEL<sup>+</sup> Language.

## Usage

**VSet** *Sequence.Property, value*

**VSet** *Calibration.Property, value*

**VSet** *Sequence.Object.Property, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Calibration* Name of the Calibration to set the property value for.

*Object* Name of the Object set the property value for. Omit if setting property for sequence or calibration.

*Property* Name of the property to set a new value for.

*value* Expression for the new value. The data type depends on the property type.

## Remarks

VSet is used to set property values for vision sequences, calibrations, and objects from the SPEL<sup>+</sup> language. Save the vision settings using VSave after setting the property.

For many vision sequences all the proper property settings will be set from within the Vision Guide development environment. However, there are also times when you will want to set property values in a SPEL<sup>+</sup> program prior to running a vision sequence. For example, you may want to set the NumberToFind property before running a sequence, or maybe you want to use the same vision sequence with 2 different cameras. Both of these scenarios can be handled in SPEL<sup>+</sup> using VSet.

Shown below is a Vision Guide program which runs the same vision sequence for 2 different cameras to calculate the number of holes found in a board.

It is assumed that a Sequence called "FindHoles" has already been created prior to running this program. FindHoles contains a "Part" Blob object which is configured to find the number of holes in the Search Window using Holes Result. In this example, we will run the sequence and then display the number of holes which were found.

When VSet is called from a program, changes are only made in memory and are not saved. You must call VSave to make the changes permanent. Otherwise, after program execution stops, the vision system is restored to the previously saved state.

```
Function test

Integer count
#define CAMERA1 1
#define CAMERA2 2

VSet FindHoles.Camera, CAMERA1      ' Find holes for part at camera 1
VSave
VRun FindHoles                       ' Run the Vision Sequence
VGet FindHoles.Part.Holes, count    ' Get the # of holes which were found
Print "Camera1 holes found =", count

VSet FindHoles.Camera, CAMERA2      ' Repeat for camera 2
VSave
VRun FindHoles
VGet FindHoles.Part.Holes, count    ' Get the # of holes which were found
Print "Camera2 holes found =", count

Fend
```

### See Also

VGet, VRun, VSave, VSet, Vision Sequences

## VShowModel Statement

### Applies To

Vision Objects: Contour, Correlation, Geometric, DefectFinder, Polar  
CV1 firmware Ver. 2.1.0 or later

### Description

The VShowModel command allows a previously taught model to be displayed in the model window at various zoom settings from a SPEL<sup>+</sup> program. For more details, see the ShowModel property.

### Usage

**VShowModel** *Sequence.Object*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

### Remarks

If changes are made to the model, such as don't care pixels, you must call VSave to make the changes permanent.

### Example

```
vshowModel seq1.corr01
```

### See Also

VSaveModel, VTeach, VTrain

# VStatsReset Statement

## Applies To

Vision Sequence

CV1 firmware Ver. 2.1.0 or later

## Description

The VStatsReset command resets all statistics in memory for one sequence in the current project. This includes all of the objects in the sequence. If you want to save the statistics to the file, execute VStatsSave.

## Usage

**VStatsReset** *Sequence*

*Sequence* Name of a sequence or string variable containing a sequence name.

## See Also

VGet, VRun, VStatsResetAll, VStatsSave, Vision Sequences

## VStatsResetAll Statement

### Applies To

All vision sequences in current project  
CV1 firmware Ver. 2.1.0 or later

### Description

The VStatsResetAll command resets all statistics in memory for all vision sequences in the current project. If you want to save the statistics to the file, execute VStatsSave

### Usage

**VStatsResetAll**

### See Also

VGet, VRun, VStatsResetAll, VStatsSave, Vision Sequences

# VStatsSave Statement

## Applies To

All vision sequences in current project  
CV1 firmware Ver. 2.1.0 or later

## Description

The VStatsSave command saves all vision statistics in the current project to a file in the current project's directory. The filename is the name of the project with an .STX extension.

## Usage

**VStatsSave**

## Remarks

Statistics are saved whenever the EPSON RC+ is closed. So typically VStatsSave is not needed. However, if you want to reset the saved statistics, execute VStatsSave after executing VStatsResetAll, or VStatsReset.

If no sequences have been run, no statistics file will be created.

If all statistics have been reset with the VStatsResetAll command, then the file will be deleted.

When you are using the Compact Vision, the statistics file will be saved in the Compact Vision.

## See Also

VGet, VRun, VStatsReset, VStatsResetAll

# VStatsShow Statement

**Applies To**

- Vision Sequence
- CV1 firmware Ver. 2.1.0 or later

**Description**

Displays statistics for a specified sequence.

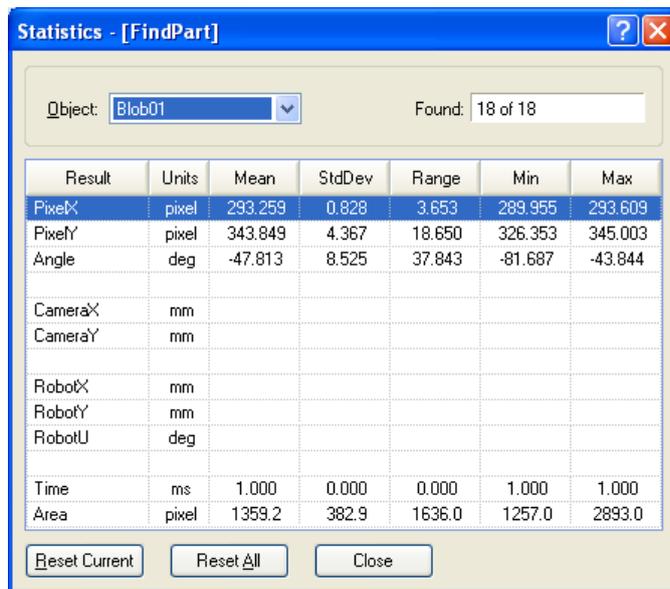
**Usage**

**VStatsShow** *Sequence*

*Sequence* Name of a sequence or string variable containing a sequence name.

**Remarks**

VStatsShow will display a dialog showing the statistics for all objects in the specified sequence.



**Example**

**VStatsShow** Seq1

**See Also**

VStatsReset, VStatsResetAll, VStatsSave

## VTeach Statement

### Applies To

Vision Objects: ColorMatch, Correlation, DefectFinder, Geometric, ImageOp, OCR, Polar  
CV1 firmware Ver. 2.1.0 or later

### Description

VTeach enables you to teach a vision model from a SPEL<sup>+</sup> program.

### Usage

**VTeach** *Sequence.Object, var*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the return status.

### Values

Returns status in *var*. If the teach operation was successful, *var* will contain 1, otherwise it will contain 0.

### Remarks

The object must exist before you can call VTeach. When VTeach executes, an image is acquired first. Next, all previous ImageOp objects are run. Then the model is taught using the current model window.

When using the ColorMatch or ImageOp objects, you must first set the CurrentModel property before executing VTeach.

After executing VTeach, you must call VSave to make the changes permanent.

### Example

```
Integer status

VTeach seq1.corr01, status
If status = 1 Then
    VSave
EndIf
```

### See Also

CurrentModel Property, VSaveModel, VTrain

# VTrain Statement

## Applies To

Vision Objects: Blob, Correlation, Edge, Geometric, Polar, ImageOp, Frame Line, Point  
CV1 firmware Ver. 2.1.0 or later

## Description

VTrain enables you to train the search window and model window for an object from a SPEL<sup>+</sup> program.

## Usage

**VTrain** *Sequence* [*Object*], *var* [, *flags* ]

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Optional. Name of an object or string variable containing an object name. The object must exist in the specified sequence. If omitted, the entire sequence can be trained.

*var* Integer variable that will contain the return status.

*flags* Optional. Configures VTrain operation.  
1 - Show teach button.  
2 - Don't show model windows.

## Values

Returns status in *var*. If the user clicks OK, *var* will contain 1, otherwise it will contain 0.

## Remarks

The sequence must exist before you call VTrain. If *Object* is specified then it must exist in the specified sequence before you can call VTrain. When VTrain executes, a dialog is opened showing live video with the specified sequence or object. The user can then move and size the search and model windows just as you would in the Vision Guide window.

If *flags* bit 1 is set, a teach button will be displayed. For objects with models, such as Correlation, Geometric, and Polar objects, the model will be taught if the teach button is clicked. You can retrieve the ModelOK property after running VTrain to check if a model was trained. For Blob objects and ImageOp objects with operation set to Binarize, the teach button will open the Histogram dialog and the operator can adjust both high and low thresholds and then view the effects of changes.

If *flags* bit 2 is set, model windows will not be displayed. The operator can only change search windows.

For objects with models, you can call VTech after calling VTrain to teach the model if you are not displaying the teach button.

After executing VTrain, you must call VSave to make the changes permanent.

## See Also

VTech, VSave

# X Property

## Applies To

Vision Objects: Point

## Description

Defines the X coordinate of a Point object.

## Usage

**VGet** *Sequence.Object.X*, *var*

**VSet** *Sequence.Object.X*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Min: 0

Max Video width: -1

## Remarks

The X property is used to specify the horizontal position of the Point object in the image coordinate system. This property is initially set to the X coordinate where the user drops a newly created Point object. However, if a Point object is associated with another object in the sequence (i.e. the PointType property is set to another vision object and not set to 0-Screen), then the X property for the Point object is automatically modified according to the associated object.

When the PointType property is set to 0-Screen there are two methods which can be used to move the Point object:

- Click on the Point object's label and drag the object to the position you want to place it.
- Change the X and Y properties for the Point object.

## See Also

Object Tab, Point Object, Y Property

# X1 Property

## Applies To

Vision Objects: Edge, Line, LineInspector

## Description

Defines the X1 coordinate of an object where the (X1, Y1) coordinate pair defines the position of the starting point of the object.

## Usage

**VGet** *Sequence.Object.X1*, *var*

**VSet** *Sequence.Object.X1*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Min: 0

Max Video width - 1

## Remarks

Line, LineInspector Objects:

The X1 property is used to locate horizontal position for the starting point of a Line object. This property is initially set to the starting point X position where the user drops a newly created Line object. However, if a Point object is associated with another object in the sequence (i.e. the StartPointType property is set to another vision object and not set to 0-Screen), then the X1 property for the Line object is automatically modified according to the associated property.

When the StartPointType property is set to 0-Screen there are 2 methods which can be used to move the Line object:

- Click on the Line object's label and drag the object to the position you want to place it.
- Change the X1, Y1, X2, or Y2 coordinates.

Edge Object:

The X1 property is used to locate horizontal position for the starting point of an Edge object.

## See Also

Edge Object, Line Object, LineInspector Object, Object Tab, StartPointObject Property, StartPointType Property, X2 Property, Y1 Property, Y2 Property

# X2 Property

## Applies To

Vision Objects: Edge, Line, LineInspector

## Description

Defines the X2 coordinate of an object where the (X2, Y2) coordinate pair defines the position of the starting point of the object.

## Usage

**VGet** *Sequence.Object.X2, var*

**VSet** *Sequence.Object.X2, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Min: 0

Max Video width: -1

## Remarks

Line, LineInspector Objects:

The X2 property is used to locate horizontal position for the endpoint of a Line object. This property is initially set to the ending point X position where the user drops a newly created Line object. However, if a Point object is associated with another object in the sequence (i.e. the EndPointerType property is set to another vision object and not set to 0-Screen), then the X2 property for the Line object is automatically modified according to the associated property.

When the EndPointerType property is set to 0-Screen there are 2 methods which can be used to move the Line object:

- Click on the Line object's label and drag the object to the position you want to place it.
- Change the X1, Y1, X2, or Y2 coordinates.

Edge Object:

The X2 property is used to locate horizontal position for the ending point of an Edge object.

## See Also

Edge Object, EndPointObject Property, EndPointerType Property, Line Object, LineInspector Object, Object Tab, X1 Property, Y1 Property, Y2 Property

## XAvgError Result

### Applies To

Vision Calibration

### Description

Returns the average calibration error along the X axis.

### Usage

**VGet** *Calibration.XAvgError*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

XAvgError is the average calibration error along the X axis detected during calibration.

### See Also

XMaxError, XmmPerPixel, YAvgError

## XMaxError Result

### Applies To

Vision Calibration

### Description

Returns the maximum calibration error along the X axis.

### Usage

**VGet** *Calibration.XMaxError*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

XMaxError is the maximum calibration error along the X axis detected during calibration.

### See Also

XAvgError, XmmPerPixel, YMaxError

## XmmPerPixel Result

### Applies To

Vision Calibration

### Description

Returns the X millimeters/pixel value of the specified calibration.

### Usage

**VGet** *Calibration.XmmPerPixel*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

XmmPerPixel is the number of millimeters per pixel along the camera X axis. The calibration must be completed before XmmPerPixel can be retrieved.

### See Also

FOVHeight Result, FOVWidth Result, XAvgError Result, XMaxError Result, YmmPerPixel Result

## XTilt Result

### Applies To

Vision Calibration

### Description

Returns the calibration X tilt result.

### Usage

**VGet** *Calibration.XTilt*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Remarks

XTilt is a relative value that indicates camera tilt along the camera X axis. The directions are as viewed from the camera in the image coordinate system (plus x is right).

A positive value indicates tilt to the right, negative is tilt to the left.

### See Also

YTilt Result

## Y Property

### Applies To

Vision Objects: Point

### Description

Defines the Y coordinate of a Point object.

### Usage

**VGet** *Sequence.Object.Y*, *var*

**VSet** *Sequence.Object.Y*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Min: 0

Max: Video height- 1

### Remarks

The Y property is used to locate the vertical position of the Point object in the image coordinate system. This property is initially set to the Y position where the user drops a newly created Point object. However, if a Point object is associated with another object in the sequence (i.e. the PointType property is set to another vision object and not set to 0-Screen), then the Y property for the Point object is automatically modified according to the associated object.

When the PointType property is set to 0-Screen, there are two methods which can be used to move the Point object:

- Click on the Point object's label and drag the object to the position you want to -place it.
- Change the X and Y properties for the Point object.

### See Also

Object Tab, Point Object, X Property

# Y1 Property

## Applies To

Vision Objects: Edge, Line, LineInspector

## Description

Defines the Y1 coordinate of an object where the (X1, Y1) coordinate pair defines the position of the starting point of the object.

## Usage

**VGet** *Sequence.Object.Y1, var*

**VSet** *Sequence.Object.Y1, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

Min: 0

Max :Video height- 1

## Remarks

Line, LineInspector Objects:

The Y1 property is used to locate vertical position for the starting point of a Line object. This property is initially set to the Y position where the user drops a newly created Line object. However, if a Point object is associated with another object in the sequence (i.e. the StartPointType property is set to another vision object and not set to 0-Screen), then the Y1 property for the Line object is automatically modified according to the associated property.

When the StartPointType property is set to 0-Screen there are 2 methods which can be used to move the Line object:

- Click on the Line object's label and drag the object to the position you want to place it.
- Change the X1, Y1, X2, or Y2 coordinates.

Edge Object:

The Y1 property is used to locate vertical position for the starting point of an Edge object.

## See Also

Edge Object, Line Object, LineInspector Object, Object Tab, StartPointObject property, StartPointType property, X1 property, X2 property, Y2 property

## Y2 Property

### Applies To

Vision Objects: Edge, Line, LineInspector

### Description

Defines the Y2 coordinate of an object where the (X2, Y2) coordinate pair defines the position of the starting point of the object.

### Usage

**VGet** *Sequence.Object.Y2*, *var*

**VSet** *Sequence.Object.Y2*, *value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

### Values

Min: 0

Max: Video height- 1

### Remarks

There are cases where the user may want to position a Point object dynamically and for that reason the Y2 property can also be set from the SPEL<sup>+</sup> Language.

Line, LineInspector Objects:

The Y2 property is used to locate vertical position for the ending point of a Line object. This property is initially set to the ending point Y position where the user drops a newly created Line object. However, if a Point object is associated with another object in the sequence (i.e. the EndPointerType property is set to another vision object and not set to 0-Screen), then the Y2 property for the Line object is automatically modified according to the associated property.

When the EndPointerType property is set to 0-Screen there are two methods which can be used to move the Line object:

- Click on the Line object's label and drag the object to the position you want to place it.
- Change the X1, Y1, X2, or Y2 coordinates.

Edge Object:

The Y1 property is used to locate vertical position for the ending point of an Edge object.

### See Also

Edge Object, EndPointObject Property, EndPointerType Property, Line Object, LineInspector Object, Object Tab, X1 Property, X2 Property, Y1 Property

# YAxisPntObjResult Property

## Applies To

Vision Objects: Frame

## Description

Specifies which result to use from the YAxisPointObject.

## Usage

**VGet** *Sequence.Object.YAxisPntObjResult, var*

**VSet** *Sequence.Object.YAxisPntObjResult, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Integer variable that will contain the value of the property.

*value* Integer expression for the new value of the property.

## Values

The value can range from 1 to the NumberToFind value for the YAxisPointObject.

If the YAxisPointObject is 'Screen', then the value is always 1.

## Remarks

Use the YAxisPntObjResult property to specify a result number other than one for a Frame Object's YAxisPoint.

## See Also

Frame Object, Object Tab, OriginPntObjResult Property, OriginPoint Property, YAxisPoint Property

# YAxisPoint Property

## Applies To

Vision Objects: Frame

## Description

Defines the vision object to be used as the Y axis point for a Frame object.

## Usage

**VGet** *Sequence.Object.YAxisPoint, var*

**VSet** *Sequence.Object.YAxisPoint, value*

*Sequence* Name of a sequence or string variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* String variable that will contain the value of the property.

*value* String expression for the new value of the property. Valid vision objects for the YAxisPoint property are: Blob, Correlation, Edge, Line, and Point objects. The YAxisPoint may also be based on the Screen position of the Frame.

## Values

Screen or any object that runs prior to the frame and returns PixelX and PixelY results.

Default: Screen

## Remarks

When a Frame object is first drag-and-dropped onto the Image display area of the Vision Guide Window, the default YAxisPoint property is set to Screen. Frame objects are normally attached to other Vision objects. This is the purpose of the OriginPoint and YAxisPoint. Through these 2 properties the user can define a frame of reference for other objects to have their position based upon. This capability is useful when specific features can be used to find reference points on a part and then other vision objects can be located on the image with respect to the frame position defined.

The OriginPoint and YAxisPoint properties are used together to define a vision frame which has an origin at the OriginPoint and a Y axis direction defined by the YAxisPoint property.

It is important to note that for each specific vision sequence, only those vision objects which are executed prior to the Frame object in the vision sequence steps will be available to use as an OriginPoint. (The order of the vision object execution can be adjusted from the Sequence Tab.)

When using the point and click interface click on the YAxisPoint property Value Field. Then click on the arrow and a drop down list will appear showing a list of available vision objects (along with the default value Screen) which can be used to define the Y Axis direction of the Frame. Click on one of the choices and the value field will be set accordingly.

When using the Object tab to set the YAxisPoint property it is important to note that only those objects which are defined prior to the Frame object are displayed in the drop down list. This helps reduce the chances of the user defining an OriginPoint which isn't defined prior to the Frame object.

Vision Guide automatically checks which vision objects can be used as the YAxisPoint and displays only those object names in the drop down list.

## See Also

Frame Object, Frame Property, Object Tab, OriginPoint Property

## YAvgError Result

### Applies To

Vision Calibration

### Description

Returns the average calibration error along the Y axis.

### Usage

**VGet** *Calibration.YAvgErr*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

YAvgError is the average calibration error along the Y axis detected during calibration.

### See Also

XAvgError Result, YMaxError Result, YmmPerPixel Result

## YMaxError Result

### Applies To

Vision Calibration

### Description

Returns the maximum calibration error along the Y axis.

### Usage

**VGet** *Calibration.YMaxErr*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Values

Real number in millimeters.

### Remarks

YMaxError is the maximum calibration error along the Y axis detected during calibration.

### See Also

XMaxError Result, YAvgError Result, YmmPerPixel Result

# YmmPerPixel Result

## Applies To

Vision Calibration

## Description

Returns the Y millimeters/pixel value of the specified calibration.

## Usage

**VGet** *Calibration.YmmPerPixel, var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

## Values

Real number in millimeters.

## Remarks

YmmPerPixel is the number of millimeters per pixel along the camera Y axis. The calibration must be completed before YmmPerPixel can be retrieved.

## See Also

FOVHeight Result, FOVWidth Result, XmmPerPixel Result, YAvgError Result, YMaxError Result

## YTilt Result

### Applies To

Vision Calibration

### Description

Returns the calibration Y tilt result.

### Usage

**VGet** *Calibration.YTilt*, *var*

*Calibration* Name of a calibration or string variable containing a calibration name.

*var* Real variable that will contain the value of the result.

### Remarks

YTilt is a relative value that indicates camera tilt along the camera Y axis. The directions are as viewed from the camera in the image coordinate system (plus y is down).

A positive value indicates tilt down, negative indicates tilt up.

### See Also

XTilt Result

# ZoomFactor Property

## Applies To

Vision Object: ImageOp

## Description

Magnifies or reduces an image area.

## Usage

**VGet** *Sequence.Object.ZoomFactor*, *var*

**VSet** *Sequence.Object.ZoomFactor*, *value*

*Sequence* String variable containing a sequence name.

*Object* Name of an object or string variable containing an object name. The object must exist in the specified sequence.

*var* Real variable that will contain the value of the property.

*value* Real value or expression for the new value of the property.

## Values

Positive real value between 0.1 and 10.0.

## Remarks

ZoomFactor resizes the image bounded by the ImageOp search window from the center of the window. When the image is magnified (ZoomFactor value is greater than 1), the enlarged image is clipped by the search window. When the image is reduced (ZoomFactor is less than 1), the image data outside of the search window is used. If not enough data is available, an error will occur.

## See Also

ImageOp Object, Operation Property

