



Intellect 1.1.x Commands List

1. Table of Contents

1. Table of Contents	2
2. Commands List Basics	5
Command Structure	6
Sample Terminal Command Session.....	7
Binary Packet Format	8
3. Determining a Sensor's Parameters	9
Sensor Query Steps	10
4. Common Sensor Parameters	12
General Parameters Tab	13
Sensor Name.....	13
Sensor Description	13
Perform Inspection	14
Position Reference	14
Image Reference	14
Local Marking Mode and Timeout.....	15
Scaling	16
Consumer	16
Producer	17
Shape	18
Position Reference, Transform and Common Shape Parameters.....	18
Shape Specific Parameters	20
Threshold.....	38
Inspection Outputs	41
5. Sensor Specific Parameters	43
Preprocessing	44
Preprocessing Filter.....	44
Positioning	50
Position Along Line/Arc	50
Position in Area	53
Circle/Line Fit.....	58
Counting	62
Count Along Line	62
Count in Area.....	64
Pixel Counting	70
Measurement.....	75
Measure Along Line.....	75
Measure in Area	76
Measure in Circle.....	79
Measure with Points and Lines	81
Datum	83
Readers	84
2D Reader	84
OCR.....	90
Flaw Detection	96
Defect Detection	96
6. Terminal Command Reference.....	100

Access Control Commands	101
Al - Log in user	101
AO - Log out user	102
AU - Get current user	103
Digital I/O Commands.....	104
Dq - Query digital I/O.....	104
DS - Set physical I/O bits.....	106
Image Management Commands.....	107
li - Query inspection statistics.....	107
li - Query specific inspection statistics.....	109
liz - Reset inspection statistics	111
liz - Reset specific inspection statistics	112
IP - Set system buffers	113
IQ - Query image info	114
Script Management Commands	116
ZL - List background scripts	116
ZN - Start/stop background script by name.....	118
ZR - Start/stop background script by ID	119
Product Management Commands	120
PA - Add product	120
PD - Delete product	121
PF - Save products.....	122
PI - Select inspection product.....	123
PN - Rename product.....	124
PQ - General product query	125
Pq - Specific product query	126
PR - Reset product results	127
PS - Product select by product ID	128
PSI - Product select by digital ID	129
PSN - Product select by name	130
Register Access Commands.....	131
Rq - Register query	131
Rs - Register set.....	133
Sensor Management Commands	134
SA - Add sensor	134
SD - Delete sensor	135
SI - Change sensor ID	136
SM - List sensor modules	137
SN - Rename sensor.....	139
Sm - Set reader match string	140
See Also	140
SP - Set sensor parameters	141
SQ - Sensor query	143
Sq - Sensor results query	145
SR - Reset sensor results.....	147
System Management Commands.....	148
Y - Enable/disable inspections	148
Yf - Query flash file system.....	149
YI - Trigger inspection	150
YN - Get/set image sensor name	151
YP - Set trigger source	152
Yp - Set trigger mode	153
YQ - Query system status	154
Yq - Query system type	156

YR3742 - Reset camera	158
YS1bdc - Reformat and save	159
Ys - Query format progress	160
YV - Query firmware	161
7. Intellect 1.2 Command line Options.....	162

2.Commands List Basics

This document defines the commands used to communicate between a user interface like Intellect and DVT cameras. These commands are thus the underlying language used to program DVT cameras. They can be used to create products and sensors, change sensor parameters, and query the camera for specific data. While these commands should ideally remain transparent to the user, some users have found need for functionality that is not programmable through the Intellect user interface. By request these commands have been released to help solve some tough applications that would not have been solvable otherwise.

Here is the informal disclaimer. These commands can be difficult to use and often change as product development continues. These commands are meant for R&D use, not customer use, so they are provided as is. While these commands are not “officially” supported by DVT, we recognize their usefulness in some circumstances and so provide this document as a guide for advanced users.

Command Structure

Intellect commands and responses can contain strings of ASCII characters and numerical information. Numbers can be in one of three formats - Decimal, Hexadecimal, or Binary. All commands and responses are prefixed with a character identifying the command format. These prefix characters are:

Format	Command	Reply	End Reply
Decimal	#	\$	%
Hexadecimal	*	(	)
Binary	0xAA	0xA8	0xA9

Table 2-1: Format Specific Characters

In Decimal mode, all numerical parameters are separated by a space character. All commands are terminated with a CR (just hit 'Enter' if sending the commands manually from a terminal). Each reply sequence is completed with an EndReply with a value representing success or error, respectively, and a then followed by a ? or ! for success or error.

Value	Description
0	Command successfully completed
1	General (unspecified) error
2	Access denied – user is not authorized to perform this operation.
3	Out of memory (system RAM).
4	Error accessing flash memory.
5	Network error.
6	Invalid parameter - one of the command parameters is invalid.
7	Locked – attempt to modify a parameter/module that is being used by another user or process.

Table 2-2: Command Reply Codes

Sample Terminal Command Session

To bring up an Intellect terminal, run Intellect from within Windows and connect to a DVT system running Intellect. From the menu select **View, Terminal Window...** or press **Control-Alt-T** to open a terminal window. Then use the commands shown below to configure your specific sensors. Commands entered by the user are in **bold** and responses are in *Italics*.

Example: System has two products defined. Product 'Clip' (id=1) is selected as the inspection product and the power on inspection product.

Command/Response	Description
#PQ	Query system for products
\$ -1 1 1	No selected Product, Product 1 set as power-on and inspection product. Product 0 is named 'Test Product'.
\$ 0 Test Product	
\$ 1 Clip	Product 1 is named 'Clip'.
% 0	Command completed without error.
?	Command completed without error.
#Pq 2	Query product 2 for pass/fail stats.
% 1	Error performing command.
!	Error performing command.
#Pq 1	Query product 1 for pass/fail stats.
\$ 100 2 3 -1	Product 1 has performed 100 Pass, 2 Warn, 3 Fail inspections, and the last inspection was Pass.
% 0	
?	Command completed without error.
#PS 1	Select Product 1
% 0	Command completed without error.
?	Command completed without error.
#SQ	Query selected product ('Clip') for defined sensors.
\$ 2 1 Sensor 2	'Sensor 2' has an ID of 2 and is a type 1 sensor.
\$ 5 2 Test Path	'Test Path' has an ID of 5 and is a type 2 sensor.
% 0	Command completed without error.
?	Command completed without error.
#Sq 5	Query Sensor 0 of selected product for pass/fail stats.
\$ 5 1 Test Path	Sensor 0 id is 0, is active, and is named 'Clip Fold'.
\$ 101 2 2 0	Sensor 0 has performed 101 Pass, 2 Warn, and 3 Fail inspections, and the last inspection result was Pass
%0	
?	Command completed without error.

Binary Packet Format

With decimal or hexadecimal packet formats, the end of the packet can be easily determined by parsing until a carriage return is received. When the system is sending data in binary format, additional information is sent in order to allow the system to conveniently separate individual packets in the stream. After the command type is sent (0xA8, 0xA9, 0xAA), a 16 bit unsigned packet length is sent and then the data for the packet. The current system limits a single packet to 2048 bytes of data.

In binary mode, the type of the data must be adhered to explicitly in order to correctly read and write to the system. The descriptions for the commands give the explicit type for each data element.

The data in the binary packets is stored in big endian order. With a computer system that operates in little endian mode, such as an Intel PC, each multibyte entity must have the byte order reversed.

In order to reliably decode the binary information, a set of functions should be written to parse the individual packets from the information stream. A second set of functions should be written in order to read data of the appropriate types (UINT8, UINT16, UINT32, INT8, INT16, INT32, FLOAT32, FLOAT64, String) from the packet.

3. Determining a Sensor's Parameters

This section details each sensor's response to the SQ (Sensor Query) command. The SQ response contains all the sensor instance data including inspection parameters, pass/fail parameters, position and shape, position reference, image reference, timeout, transform settings and scale data. This often is a lot of information to sift through, so this section uses tables to associate the Intellect sensor parameters with the data returned by the SQ command for a specific sensor.

Parameters common to all or many of the sensors are grouped together in the first section "Common Sensor Parameters". Parameters that are specific to a sensor's algorithm or that need specific detailed explanation are grouped in the section "Specific Sensor Parameters". To avoid bloating this document the common sensor parameters are not repeated in the individual sections for specific sensors, so refer back to "Common Sensor Parameters" for that information. Each section includes examples of sending data to set specific parameters.

Why would you want to look at this data? This is a good question since most parameters and the functionality this data represents are already accessible through the Intellect user interface. The problem is the word 'most'. Sometimes advanced users need to access parameters that aren't exposed in the Intellect UI. Sometimes they need more flexible control over when and how this data is set. Specifically, the most common use of this data is as a reference to what data to send with the SP (Set Parameters) command to programmatically set specific sensor parameters, especially shape and position, from a remote system like a custom user interface. This is often not trivial since each parameter exists not by itself, but rather grouped with other similar parameters which must be simultaneously set. Thus understanding this data's format helps separate the parameters of interest from the rest. Because usage of the SP command is common, examples of sending SP commands over a terminal (Intellect terminal, telnet session, socket connection, etc...) and examples of using the script command SetParams() to set some parameters are included in each section.

Sensor Query Steps

The following is a copy of a terminal session in which a Defect Detection sensor is queried for its sensor parameters. This is the easiest process for determining the syntax necessary to set sensor parameters (from somewhere other than the Intellect UI parameter windows). The steps involve three commands: PQ, PS, and SQ. You can get more information on each in Chapter 3, the Terminal Command Reference.

System Terminal Driver

```
?#PQ
$ -1 2 -1
$ 1 Product1
$ 2 Product2
% 0

?#PS 1
%0

?#SQ
$ 1 17 Sensor1
$ 2 20 Sensor2
% 0

?#SQ 2
$ 240 "Sensor1"
$ 241 1 0
$ 242 ""
$ 243 1
$ 245 0 1 -1 "" 1
$ 246 0 1 0 0 -1 "" 1
$ 251 -1
$ 253 2000 0
$ 255 -1
$ 0 7 1 -1 2544.092312 324 265 1 0 -1 0 0 1 1 1 1 1 1 50 0 0
5 27 202 621 202 621 328 27 328 27 202
$ 2 18
$ 3 -1
% 0
?
```

Step 1: Send the #PQ command. This queries the system for existing products. The response indicates the *ProductID* and name of each product.

Step 2: Send the #PS command. This command sets a specific product as the one to which we will be sending sensor commands. This command must include a *ProductID*, like #PS 1 or #PS 2, to indicate which product is to be selected. Here we've selected Product1.

Step 3: Send the #SQ command. This is basically the same as the #PQ command but the response tells us the *SensorID*, sensor type and name of each sensor in the selected product. Here Sensor1 has a *SensorID* = 1.

Step 4: Send the #SQ command with a specific *SensorID*. The *SensorID* indicates which sensor to query for parameter information. The response is a list of sensor parameters and their formats. Each grouping of common parameters is preceded by a '\$'. For instance '\$0' is always the

grouping for sensor shape, position and transform information. Notice that the shape and position data actually take up two lines. There is no '\$' on the second line, indicating that data is still part of the previous line's '\$0' data.

Step 5: Compare the reported data to what you see in Intellect to figure out exactly what each line represents. For instance, the numbers '27 202 621 202 621 328 27 328 27 202' are the (x,y) corner points of our Defect Detection sensor, which could be verified by looking at the mouse cursor coordinates as it moves over the sensor in the image. Alternatively, use the tables provided in this document to interpret the query response.

****These steps can be followed to query any sensor for its parameter information****

4.Common Sensor Parameters

This section describes the format for setting parameters that are common to many or all sensors. The following table describes the parameter IDs associated with each of the common parameters.

Parameter ID		Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data

General Parameters Tab

When you view the parameters window for a sensor in Intellect, it will probably look like this.

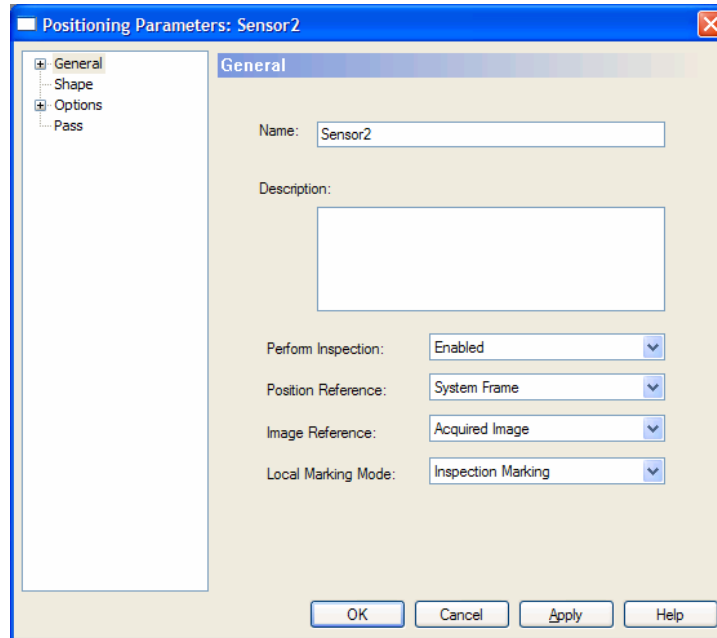


Figure 4-1: General parameters tab

No sensor will have more settings than you see here on the General tab, though some will not have a position or image reference. The following sections describe how to set each of these parameters.

Sensor Name

This parameter sets the name of any sensor.

\$240 - Set Sensor Name			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	240	Sensor Name	UINT8
Sensor Name	"..."	Character string	STRING

Table 4-1: Sensor Name

Terminal Example

```
? #SP 1 240 BlobSensor
```

Script Example

```
MySensorVar.SetParams("240 BlobSensor");
```

Sensor Description

This parameter sets the description associated with any sensor.

\$242 - Set Sensor Description			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	242	Sensor Description	UINT8
Sensor Description	"..."	Text in quotes	STRING

Table 4-2: Sensor Description

Terminal Example

```
? #SP 1 242 The crow flies at midnight
```

Script Example

```
MySensorVar.SetParams("242 The crow flies at midnight");
```

Perform Inspection

This parameter sets the inspection flag for any sensor. If disabled (0) the sensor will not be used during the inspection.

\$243 - Set Inspection Flag			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	243	Inspection Flag	UINT8
Perform Inspection Flag	0	Disable	UINT8
	1	Enable	

Table 4-3: Perform Inspection Flag

Terminal Example

```
? #SP 1 243 1
```

Script Example

```
MySensorVar.SetParams("243 1");
```

Position Reference

These parameters determine whether a sensor uses position referencing, to which sensor it is referenced, and the offset and rotation from the referenced sensor. Position reference parameters are included with the shape and position parameters (\$0) later in this section.

Image Reference

This parameter sets the image reference for any sensor. An image reference is actually a reference to a sensor that produces an output image that can be inspected by other sensors. The default image reference is the unchanged acquired image. This reference may be set to any filter or defect detection sensor.

\$255 - Set Sensor Image Reference			
SP <sensor ID> <parameter ID> <Data>			

Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	255	Image Reference	UINT8
Image Reference	-1	No image reference	INT16
	0-32,767	Sensor ID	

Table 4-4: Image Reference

Terminal Example

```
? #SP 1 255 3 //Set the image reference to sensor ID 3
```

Script Example

```
MySensorVar.SetParams("255 3");
```

Local Marking Mode and Timeout

This parameter sets the sensor timeout value in milliseconds and the local marking mode. A sensor automatically fails if it has not finished inspecting before the timeout (the timeout value is normally set in the properties window, not in the sensor's pop up parameters window). Sensor marking can be controlled at the product level and at the sensor level. By default the product allows all sensors to control how they mark their inspection results and all sensors are set to display inspection result markings. If the product marking mode is set to defer to "Local Sensor Settings", then the sensor's local marking mode takes over. For instance, if the local marking mode is set to '3' then the sensor will only mark the passing results. Other sensors will continue to mark based on their local marking modes.

\$253 - Set Sensor Timeout and Local Marking Mode			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	253	Timeout and Local Marking Mode	UINT8
Timeout value	0-60,000	Timeout in ms	UINT32
Local Marking Mode	0	Do not mark	UINT8
	1	Inspection Marking	
	2	Pass/Fail Result	
	3	Pass Result	
	4	Fail Result	

Table 4-5: Timeout and Local Marking Mode

Terminal Example

```
? #SP 1 253 150 3
```

Script Example

```
MySensorVar.SetParams("253 150 3");
```

Scaling

Scaling parameters are used with any sensor that provides point to point dimensional output like distance or radius. Below is a screenshot of the Scaling tab.

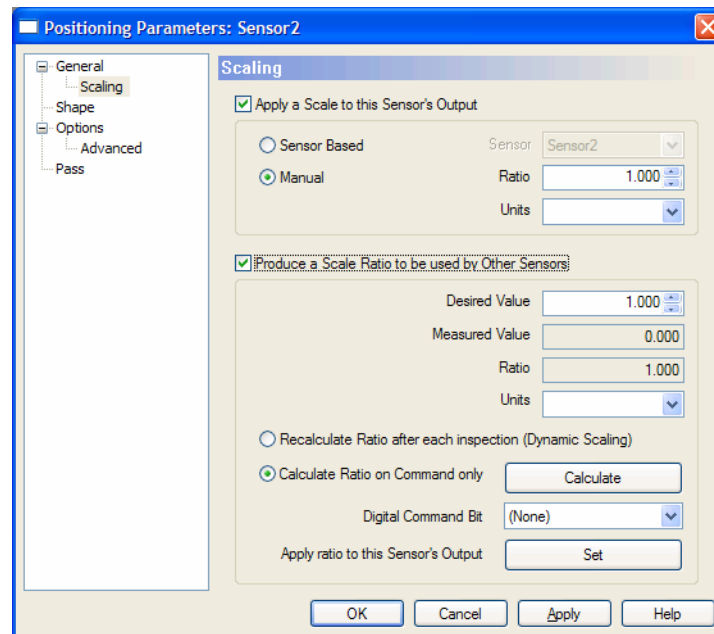


Figure 4-2: Sensor Scaling tab

Sensors are divided in two scaling categories: Producers and Consumers. Producers output results that are used as a scale reference by other sensors. Consumers are those 'other' sensors. A sensor can be a producer of a scale but not actually consume a scale. Likewise, a sensor can both produce and consume its own scale, or produce a scale for other sensors, but consume the scale produced by another. The following tables describe the formats for setting scaling parameters.

Consumer

This parameter sets whether a sensor is a scale consumer, the scale source, and the scale units.

\$245 - Set Sensor Scale Consumer Data			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	245	Scale Consumer Data	UINT8
Apply a Scale	0	Disabled	UINT8
	1	Enabled	
Manual/Sensor Based	1	Manual	UINT8
	2	Sensor based	
Sensor Scale Reference	-1	None	INT16
	0-32,767	Sensor ID	

Manual Units	""	Units as char string	STRING
Manual Ratio	0-999.99	Scale ratio as number	FLOAT

Table 4-6: Scale Consumer Data

Terminal Example

```
? #SP 1 245 1 2 3 "mm" 1.0 //Enable sensor 1 as scale consumer, reference to sensor ID 3
```

Script Example

```
MySensorVar.SetParams("245 1 2 3 "mm" 1.0");
```

Producer

This parameter sets whether the sensor is a scale producer, what the scale ratio is, and when to recalculate the scale ratio.

\$246 - Set Sensor Scale Producer Data			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	246	Scale Producer Data	UINT8
Produce a scale ratio	0	Disable	BIT
	1	Enable	
Desired value	0-255	Real world value	UINT8
Dynamic scaling	0	Disable	BIT
	1	Enable	
Recalculate Flag	0	Don't Recalculate	UINT8
	1	Recalculate	
Digital Relearn Bits	-1	None	INT16
	0-32,767	Virtual I/O bit	
Units	""	Units as char string	STRING
Computed ratio	0-999.99	Scale ratio as number	FLOAT

Table 4-7: Scale Producer Data

Terminal Example

```
? #SP 2 246 1 3.0 1 -1 "cm" 1 //Enable sensor 2 as scale producer, scale value = 3cm
```

Script Example

```
MySensorVar.SetParams("246 1 3.0 1 -1 "cm" 1");
```

The 'Calculate' and 'Set' buttons under the producer section of the scaling tab are not controlled by specific commands. Clicking these buttons causes the UI to send the appropriate commands to set the producer and consumer data. Thus the 'Calculate' and 'Set' commands can be mimicked by gathering the necessary data and sending the SP command to set producer and consumer data.

Shape

In this section we will discuss the terminal commands necessary to set individual SoftSensors' shape, location and position reference using the SP command. The SP command will be discussed extensively, not only in this section but in others to come. While there are many functions of this command, this section will focus on setting a sensor's shape, location, and position reference. What is discussed here will apply to many SoftSensors. Not all sensors support all shapes, but those which share a specific shape will have similar location, shape and position reference parameters.

To effectively execute the SoftSensor Parameters (#SP) command it will be necessary to specify the product where the SoftSensor resides and the SoftSensor internal identification number. Refer to the Chapter "Determining a Sensor's Parameters". The ID number of the desired SoftSensor will be used as part of the SoftSensor Parameter (#SP) command to change its settings. Let us begin by discussing the "#SP" commands and its syntax for setting location, shape and position reference of individual SoftSensors.

Position Reference, Transform and Common Shape Parameters

\$0 - All Shapes			
Parameters	Data	Description	Type
Sensor ID	0...32768	Sensor Specific	UINT16
Parameter ID	0	Shape	UINT8
Shape ID ¹	2	Point	UINT16
	3	Line Segment	
	4	Elliptical Arc	
	5	Annular Arc	
	6	Polyline	
	7	Rectangle	
	9	Parallelogram	
Loading from File? ²	0	No	UINT8
	1	Yes	
Transform ID ²	-1	None	INT16
	0...	Transform ID	
TransformChecksum ²	0...	Error check	FLOAT
Transformed Position X ²	0+/- max	Transformed Frame. Usually center of shape	FLOAT
Transformed Position Y ²	0 +/- max	Transformed Frame. Usually center of shape	FLOAT
Transform Angle cos ²	-1...1	Cos(θ)	FLOAT
Transform Angle sin ²	-1...1	Sin(θ)	FLOAT
Position Reference ³	0...32,768	Sensor ID	FLOAT
Reference Position X ³	0 +/- max	X offset	FLOAT
Reference Position Y ³	0 +/- max	Y offset	FLOAT

Reference Angle $\cos^3$	-1...1	Cos(θ)	FLOAT
Reference Angle $\sin^3$	-1...1	Sin(θ)	FLOAT
Defined in Transform Plane	0	No. Default	FLOAT
	1	Yes	
Rotatable	0	Shape is not rotatable. Sensor Specific	UINT8
	1	Shape is rotatable. Sensor Specific	
Aligned with Image?	0	Shape does not need to be aligned with image pixel grid. Sensor Specific	UINT8
	1	Shape needs to be aligned with image pixel grid. Sensor Specific	
Shape specific Data	...	See shape specific data table that goes with the above shape ID	Shape specific

Table 4-8: Common Shape Parameters

¹ Each one of the shape types has an identification number. They are also specified in each shape-specific parameter table below. For a list of shapes supported by each SoftSensor see the table at the end of this chapter.

² The "Loading from File" flag should always be 0. When this is the case the other values that depend on this flag will not need to be sent. Thus, "Transform ID", "Transform Checksum", "Transformed Position X", "Transformed Position Y", "Transform Angle Cos" and "Transform Angle Sin" will not be included in the command when "Loading from File" is set to 0.

³ If there is not a position reference for this SoftSensor the "Position Reference" value should be set to -1. In this case, the "Reference Position X", "Reference Position Y", "Reference Angle Cos" and "Reference Angle Sin" are not included in the command. If it is desired to set position reference for a particular SoftSensor, one just needs to specify the ID for the reference SoftSensor and enter 0, 0, 1, 0 for "Reference Position X", "Reference Position Y", "Reference Angle Cos", and "Reference Angle Sin" respectively.

SoftSensor Category	SoftSensor	Supported Shapes
Preprocessing	Filters	Rectangle
Positioning	Position Along Line	Line Segment, Elliptical Arc
	Area Positioning	Rectangle
	Line Fit	Rectangle
	Circle Fit	Annular Arc
Counting	Count Along Line	Line Segment, Elliptical Arc, Polyline
	Count in Area	Rectangle
	Pixel Counting	Rectangle, Polyline, Elliptical Arc
Measurement	Measure Along Line	Line Segment
	Measure in Area	Rectangle
	Measure in Circle	Annular Arc
	Measure with Points and Lines	N/A

	Datum	Point, Line Segment, Elliptical Arc
	Statistics	N/A
Readers	1D Codes	Rectangle, Line Segment, Elliptical Arc
	2D Reader	Rectangle
	OCR Reader	Rectangle, Annular Arc, Parallelogram
Flaw Detection	Defect Detection	Rectangle, Polyline, Elliptical Arc
	Color Monitoring	Rectangle, Polyline, Elliptical Arc
Script	Script	N/A
Spectrograph	Spectrograph	Rectangle
Application Specific	Bead Tracking	Polyline

Table 4-9: Supported Softsensor Shapes

Shape Specific Parameters

This section describes the *shape specific data* that is sent to the camera at the end of the *common shape data* string. Each sensor has specific shapes that it can take and each shape has a unique shape ID that must be used for the *Shape ID* parameter in the *common shape data* string. To set the shape of a sensor, choose the shape, match it to the shape ID in Table 2-9, and then concatenate the *common shape data* string (using the shape ID) with the appropriate *shape specific data* string from the tables below.

Each table below is specific to a shape type and shape ID and it may contain extra notes on usage and optional data. The examples show the format for sending the command from both DVT script and from a terminal.

Point

The Datum Point sensor is the only sensor that uses the point shape. Below is a screen shot of the point Shape tab.

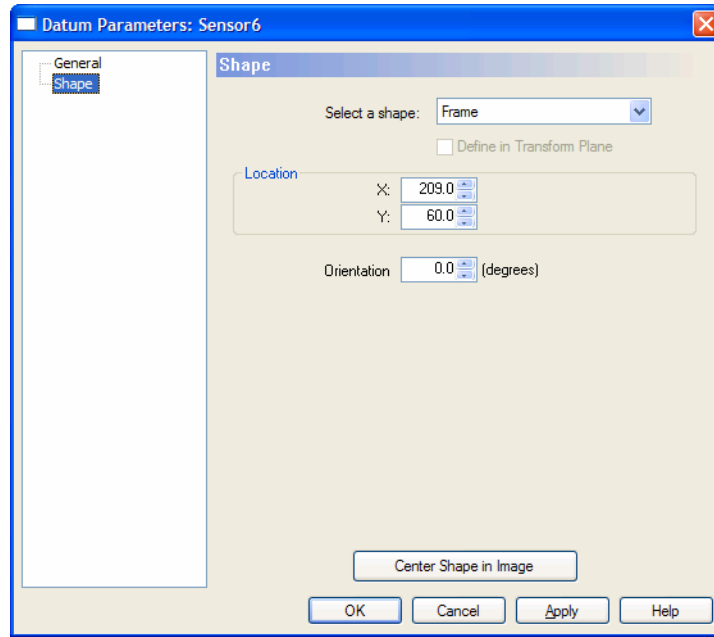


Figure 4-3: Datum Point Shape tab

Point (Shape ID = 2)			
Parameters	Data	Description	Type
Point's X Coordinate	0 +/- max	X Coordinate	FLOAT
Point's Y Coordinate	0 +/- max	Y Coordinate	FLOAT
Angle Cos	-1...1	Cos(θ)	FLOAT
Angle Sin	-1...1	Cos(θ)	FLOAT

Table 4-10: Point

Terminal Example - This sets a Datum Frame sensor to location (100, 200) with angle 0°.

```
? #SP 1 0 2 0 -1 0 1 0 100 200 1 0
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 2> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Point's X Coordinate: 100> <Point's Y Coordinate: 200>
<Angle cosine: 1> <Angle sine: 0> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 2 0 -1 0 1 0 100 200 1 0");
```

Line Segment

The Line Segment shape is used by positioning, counting, measurement, and reader sensors. A screen shot of the Line Shape tab is shown below.

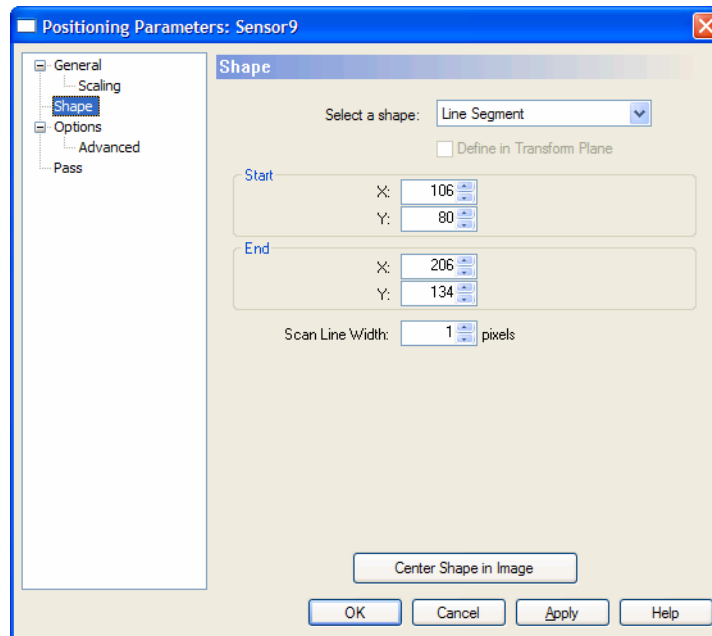


Figure 4-4: Line Segment Shape tab

Line Segment (Shape ID = 3)			
Parameters	Data	Description	Type
Line Width	0 +/- max	Expands the SoftSensor perpendicularly to the direction of the line.	FLOAT
Point 1. Position X	0 +/- max	X Coordinate	FLOAT
Point 1. Position Y	0 +/- max	Y Coordinate	FLOAT
Point 2. Position X	0 +/- max	X Coordinate	FLOAT
Point 2. Position Y	0 +/- max	Y Coordinate	FLOAT
Learn Start Perc. ¹	0...1	Start of learn region as fraction of full segment. Default is 0.1	FLOAT
Learn End Perc. ¹	0...1	End of learn region as fraction of full segment. Default is 0.9	FLOAT

Table 4-11: Line Segment

¹: These parameters are not included for the Datum Line SoftSensor or the 1D-code line reader SoftSensor. For other SoftSensors the parameters are included but do not apply and do not affect functionality. These SoftSensors are: Position Along Line (Edge Find task), Counting Along Line (Edge Count and Feature Count tasks), and Measure Along Line.

Terminal Example - This sets a Positioning Line sensor with the Pattern Find task.

```
? #SP 1 0 3 0 -1 0 1 0 1 200 50 200 150 0.20 0.35
```

Terminal Format

```
# SP<SoftSensor ID:1> <Parameter ID:0> <Shape ID: 3> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Line Width: 1> <Point 1. X Coordinate: 200> <Point 1. Y
Coordinate: 50> <Point 2. X Coordinate: 200> <Point 2. Y Coordinate: 150> <Learn Start
%: 0.20> <Learn End %: 0.35> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 3 0 -1 0 1 0 1 200 50 200 150 0.20 0.35");
```

Elliptical Arc

The Elliptical Arc shape is used by positioning, counting, readers, and defect detection sensors. A screen shot of the Elliptical Arc Shape tab is shown below.

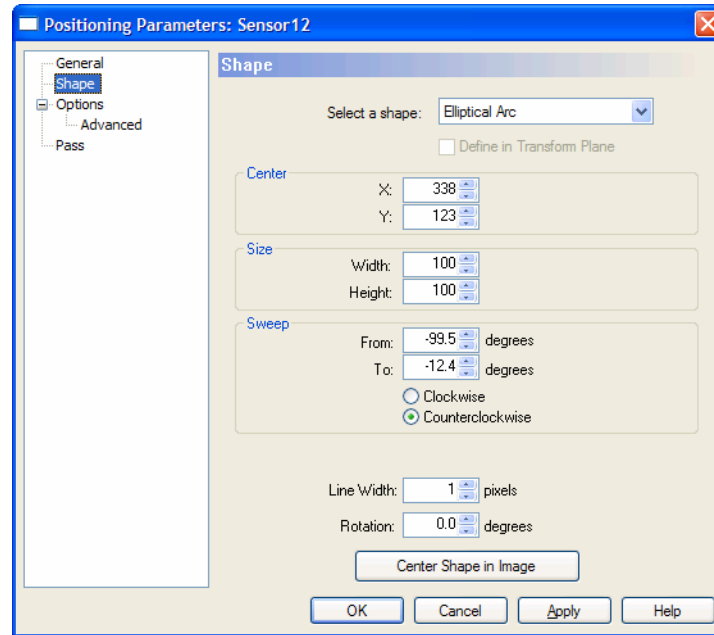


Figure 4-5: Elliptical Arc

Elliptical Arc (Shape ID = 4)			
Parameters	Data	Description	Type
Circle?	0	Not a circle	UINT8
	1	Shape is a circle. Default	
Closed Shape?	0	Not a closed shape	UINT8
	1	The shape is closed. This is independent of the shape segmentation	
Area/Contour Shape Flag ¹	0	Contour-based shape	UINT8
	1	Area-based shape	
Arc Width	0 +/- max	Expands the SoftSensor perpendicularly to the direction of the arc.	FLOAT
Center. Position X	0 +/- max	("Center" in the figure below). X Coordinate	FLOAT
Center. Position Y	0 +/- max	("Center" in the figure below). Y Coordinate	FLOAT
Start Point. Position X	0 +/- max	("AP 0" in the figure below). X Coordinate	FLOAT
Start Point. Position Y	0 +/- max	("AP 0" in the figure below). Y Coordinate	FLOAT
End Point. Position X	0 +/- max	("AP 1" in the figure below). X Coordinate	FLOAT
End Point. Position Y	0 +/- max	("AP 1" in the figure below). Y Coordinate	FLOAT
Axis Crossing 0. Position X	0 +/- max	("AX 0" in the figure below). X Coordinate	FLOAT

Axis Crossing 0. Position Y	0 +/- max	("AX 0" in the figure below). Y Coordinate	FLOAT
Axis Crossing 1. Position X	0 +/- max	("AX 1" in the figure below). X Coordinate	FLOAT
Axis Crossing 1. Position Y	0 +/- max	("AX 1" in the figure below). Y Coordinate	FLOAT
Direction	0	Counterclockwise	UINT8
	1	Clockwise	
Learn Start Perc. Or Learn region to Search region X offset. ²	0...1 Or 0+/- max	Start of learn region as percentage of full arc segment for contour tools. X offset of Search region with respect to learn region for area-based SoftSensors.	FLOAT
Learn Start Perc. Or Learn region to Search region Y offset. ²	0...1 Or 0+/- max	End of learn region as percentage of full arc segment for contour tools. Y offset of Search region with respect to learn region for area-based. SoftSensors.	FLOAT

Table 4-12: Elliptical Arc

¹: Some SoftSensors' shapes are contour-based. This means they do not inspect inside the elliptical area. This is the case for the Position Along Arc, Counting Along Arc, Datum Circle and 1-D Reader Arc SoftSensors. Other SoftSensors will have this flag set to 1 since they inspect inside the elliptical area. This is the case for the Pixel Count Ellipse, Defect Detection Ellipse and Color Monitoring Ellipse SoftSensors.

²: For SoftSensors that allow the learning of a pattern along the arc, these parameters correspond to the start and end of the learn region as a fraction of the full arc (defaults are 0.1 and 0.9 for start and end respectively). This is the case for the Position Along Arc and Count Along Arc SoftSensors. For the Pixel Counting Ellipse SoftSensor which contains learn and search areas, these parameters correspond to the X and Y distance between the limits of the learn region and the search regions to each side (up, down, right, left). For a third group of SoftSensors, these parameters are not included at all. This applies to the Datum Circle, the 1-D Reader Arc, the Defect Detection Ellipse and the Color Monitoring Ellipse SoftSensors.

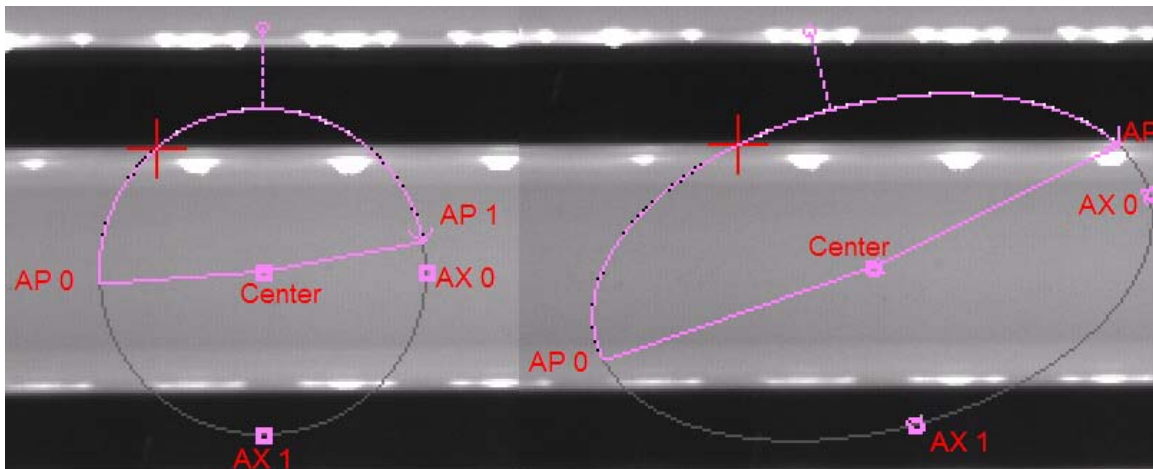


Image 4-1: Elliptical Arc

Terminal Example – This sets the Count Along Arc Line sensor

```
? #SP 1 0 4 0 -1 0 1 0 0 0 0 1 350 280 225 280 458 217.5 475 280 350 405 1 0.1 0.9
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 4> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Circle?: 0> <Closed Shape?: 0> <Area or Contour?: 0>
<Arc Width: 1> <Center. X Coordinate: 350> <Center. Y Coordinate: 280> <Start Point
(AP 0). X Coordinate: 225> <Start Point (AP 0). Y Coordinate: 280> <End Point (AP 1). X
Coordinate: 458> <End Point (AP 1). Y Coordinate: 217.5> <Axis Crossing 0 (AX 0). X
Coordinate: 475> <Axis Crossing 0 (AX 0). Y Coordinate: 280> <Axis Crossing 1 (AX 1).
X Coordinate: 350> <Axis Crossing 1 (AX 1). Y Coordinate: 405> <Direction: 1> <Learn
Start %: 0.1> <Learn End %: 0.9> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 4 0 -1 0 1 0 0 0 1 350 280 225 280 458 217.5 475 280 350
405 1 0.1 0.9");
```

Annulus Arc

The annulus arc is used by positioning, measurement and OCR sensors. A screen shot of the annulus arc shape tab is shown below.

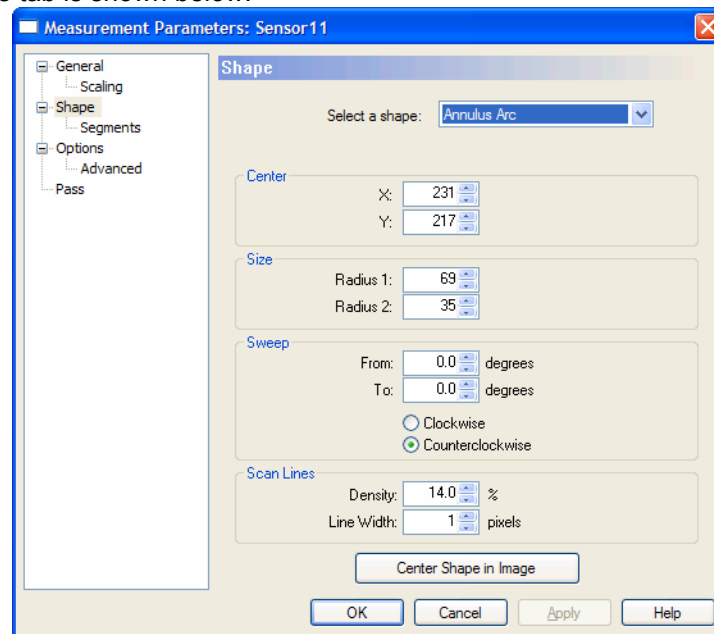


Figure 4-6: Annulus Arc

Annulus Arc (Shape ID = 5)			
Parameters	Data	Description	Type
Circle?	1	Constant. All SoftSensors using this shape have circular arcs.	UINT8
Closed Shape?	0	Constant. All SoftSensors using this shape can be segmented or are not full circles.	UINT8
Area/Contour Shape Flag ¹	0	Contour-based shape	UINT8
	1	Area-based shape (e.g. OCR Arc)	
Arc Width	0 +/- max	Expands the SoftSensor perpendicularly to the direction of the arc.	FLOAT
Arc Width (Again)	0 +/- max	Expands the SoftSensor perpendicularly	FLOAT

		to the direction of the arc. This parameter is repeated.	
Scanning Density	0 +/- max	Percentage of area being scanned. Default is 14.	FLOAT
Number of Segments ₂	0 +/- max	Some SoftSensors can be segmented; namely, the Positioning Find Circle and the Measure in Circle SoftSensors. A value of 0 here indicates no segmentation. Default is 0.	INT32
Start Percentage of First Segment ²	0 +/- max	Starting point of the first segment as a percentage of a full circle.	FLOAT
End Percentage of First Segment ²	0 +/- max	End point of the first segment as a percentage of a full circle.	FLOAT
...	...	...	...
Start Percentage of Segment N ²	0 +/- max	Starting point of the N th segment as a percentage of a full circle.	FLOAT
End Percentage of Segment N ²	0 +/- max	End point of the N th segment as a percentage of a full circle.	FLOAT
Center. Position X	0 +/- max	("Center" in the figure below). X Coordinate	FLOAT
Center. Position Y	0 +/- max	("Center" in the figure below). Y Coordinate	FLOAT
Start Point. Position X	0 +/- max	("AP 0" in the figure below). X Coordinate	FLOAT
Start Point. Position Y	0 +/- max	("AP 0" in the figure below). Y Coordinate	FLOAT
End Point. Position X	0 +/- max	("AP 1" in the figure below). X Coordinate	FLOAT
End Point. Position Y	0 +/- max	("AP 1" in the figure below). Y Coordinate	FLOAT
Axis Crossing 0. Position X	0 +/- max	("AX 0" in the figure below). X Coordinate	FLOAT
Axis Crossing 0. Position Y	0 +/- max	("AX 0" in the figure below). Y Coordinate	FLOAT
Axis Crossing 1. Position X	0 +/- max	("AX 1" in the figure below). X Coordinate	FLOAT
Axis Crossing 1. Position Y	0 +/- max	("AX 1" in the figure below). Y Coordinate	FLOAT
Direction	0	Counterclockwise	UINT8
	1	Clockwise	
Second Axis Crossing 0. Position X	0 +/- max	("SX 0" in the figure below). X Coordinate	FLOAT
Second Axis Crossing 0. Position Y	0 +/- max	("SX 0" in the figure below). Y Coordinate	FLOAT
Second Axis Crossing 1. Position X	0 +/- max	("SX 1" in the figure below). X Coordinate	FLOAT
Second Axis Crossing 1. Position Y	0 +/- max	("SX 1" in the figure below). Y Coordinate	FLOAT

¹: Some SoftSensors' shapes are contour-based. This means they do not inspect inside the annular area. This is the case for the Position Circle Fit and the Measure in Circle SoftSensors.

Other SoftSensors will have this flag set to 1 since they inspect inside the annular area. This is the case for the OCR Arc Reader SoftSensor.

²: In most cases, the Number of Segments is 0. This means the shape is not segmented. In these cases, the start and end percentages for segments are not included. If the shape is in fact segmented there must be a pair of start and end percentages as a fraction of the full circle for each segment.

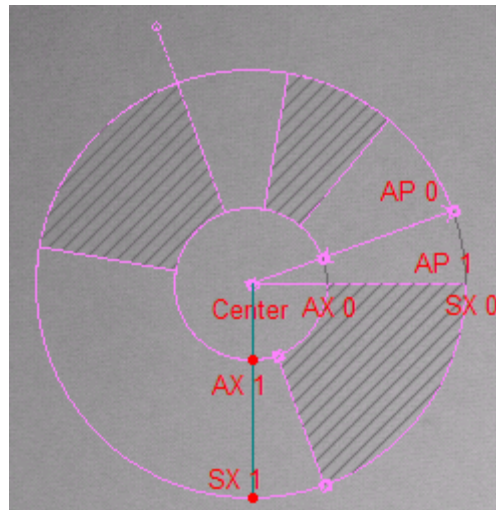


Image 4-2: Annulus Arc

Terminal Example – This set a Measurement Circle Fit sensor

```
? #SP 1 0 5 0 -1 0 1 0 1 0 0 1 1 28 0 350 250 450 250 450 250 380 250 350 280 0 450
250 350 350
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 5> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Circle?: 1> <Closed Shape?: 0> <Area or Contour?: 0>
<Scanning Line Width: 1> <Scanning Line Width (Again: 1> <Scanning Density: 28>
<Number of Segments: 0> <Center. X Coordinate: 350> <Center. Y Coordinate: 250>
<Start Point (AP 0). X Coordinate: 450> <Start Point (AP 0). Y Coordinate: 250> <End
Point (AP 1). X Coordinate: 450> <End Point (AP 1). Y Coordinate: 250> <Axis Crossing
0 (AX 0). X Coordinate: 380> <Axis Crossing 0 (AX 0). Y Coordinate: 250> <Axis
Crossing 1 (AX 1). X Coordinate: 350> <Axis Crossing 1 (AX 1). Y Coordinate: 280>
<Direction: 0> < Second Axis Crossing 0 (SX 0). X Coordinate: 450> <Axis Crossing 0
(AX 0). Y Coordinate: 250> <Second Axis Crossing 1 (SX 1). X Coordinate: 350>
<Second Axis Crossing 1 (SX 1). Y Coordinate: 350> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 5 0 -1 0 1 0 1 0 0 1 1 28 0 350 250 450 250 450 250 380
250 350 280 0 450 250 350 350");
```

Polyline

The Polyline shape is used by counting, defect detection, and application specific sensors. A screenshot of the Polyline Shape tab is shown below.

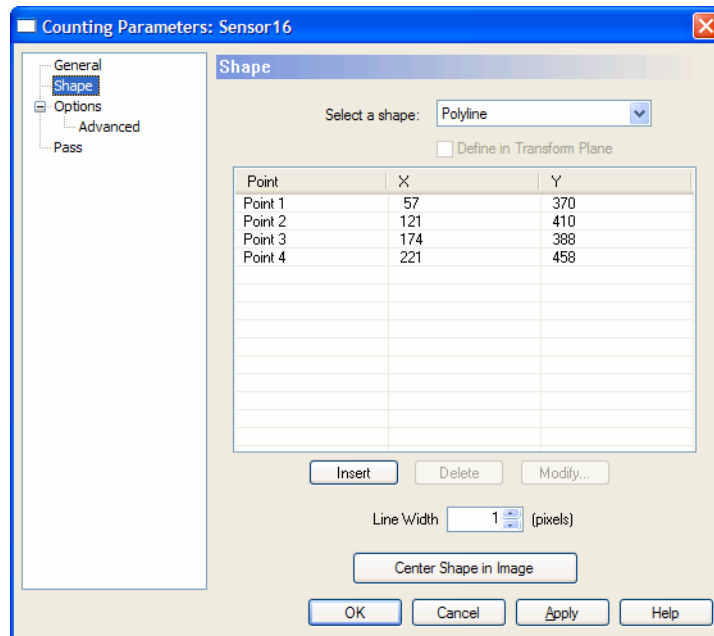


Figure 4-7: Polyline Shape tab

Polyline (Shape ID = 6)			
Parameters	Data	Description	Type
Polygon?	0	Shape is not a polygon. That is, the shape is not closed.	UINT8
	1	Shape is a polygon: that is, the shape is closed. This means that the first point is the same as the last one.	
Area/Contour Shape Flag ¹	0	Contour-based shape	UINT8
	1	Area-based shape	
Rectangle?	0	Shape is not a rectangle.	UINT8
	1	Shape is a rectangle.	
Line Width	0 +/- max	Width of line segments.	FLOAT
Number of Points for Line Segments	0 +/- max	Number of points for the whole polyline. Each point starts a new line segment. For polygons, that is, closed shapes, the last segment is the same as the first one.	INT32
First Point. X Coordinate	0 +/- max	X Coordinate of the starting point of the first segment.	FLOAT
First Point. Y Coordinate	0 +/- max	Y Coordinate of the starting point of the first segment.	FLOAT
...	...	...	...
Last Point. X Coordinate	0 +/- max	X Coordinate of the last point. For a closed shape (polygon) this will be the same as the X Coordinate for the first point.	FLOAT
Last Point. Y Coordinate	0 +/- max	Y Coordinate of the last point. For a closed shape (polygon) this will be the same as the Y Coordinate for the first	FLOAT

		point.	
Search region X offset. ²	0+/- max	X offset of Search region with respect to Learn region for area-based SoftSensors. For those SoftSensors that make use of this parameter, the default is 100.	FLOAT
Search region Y offset. ²	0+/- max	Y offset of Search region with respect to Learn region for area-based SoftSensors. For those SoftSensors that make use of this parameter, the default is 100.	FLOAT

Table 4-13: Polyline

¹: Some SoftSensors' shapes are contour-based. These are all the shapes that can be open, that is, for which the Polygon? flag is set to 0. This is the case for the Count Along Line and the Bead Tracking SoftSensors. Other SoftSensors are polygons (closed shape) and will have this flag set to 1 since they inspect inside the polygonal area. This is the case for the Pixel Count, the Defect Detection and the Color Monitoring SoftSensors.

²: For the Count Along Line SoftSensor these two parameters are always set to 0 and they need to be included on the command string. For the Pixel Count in Polygon SoftSensor, which contains learn and search areas, these parameters correspond to the X and Y distance between the limits of the learn region and the search regions to each side (up, down, right, left). For a third group of SoftSensors, these parameters are not included at all. This applies to the Defect Detection Polygon and the Color Monitoring Polygon SoftSensors.

Terminal Example – This sets a Defect Detection Polygon sensor.

```
? #SP 1 0 6 0 -1 0 1 0 1 1 0 1 5 240 65 340 45 335 90 260 90 240 65
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 6> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Polygon?: 1> <Area/Contour?: 1> <Rectangle?: 0> <Width
(Not really applicable here but needs to be included): 1> <Number of Points: 5> <Point 1.
X Coordinate: 240> <Point 1. Y Coordinate: 65> <Point 2. X Coordinate: 340> <Point 2.
Y Coordinate: 45> <Point 3.X Coordinate: 335> <Point 3. Y Coordinate: 90> <Point 4. X
Coordinate: 260> <Point 4.Y Coordinate 90> <Point 5 (same as Point 1 to close the
shape). X Coordinate: 240> <Point 5 (same as Point 1 to close the shape). Y Coordinate:
65> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 6 0 -1 0 1 0 1 1 0 1 5 240 65 340 45 335 90 260 90 240
65");
```

Rectangle

The rectangle shape is used by all sensors. Some take more or less parameters to define the shape depending on whether they are simple or double rectangles, whether they use scan lines, whether they are segmentable and whether they can rotate. Below is a screen shot of a Rectangle Measure in Area Shape tab.

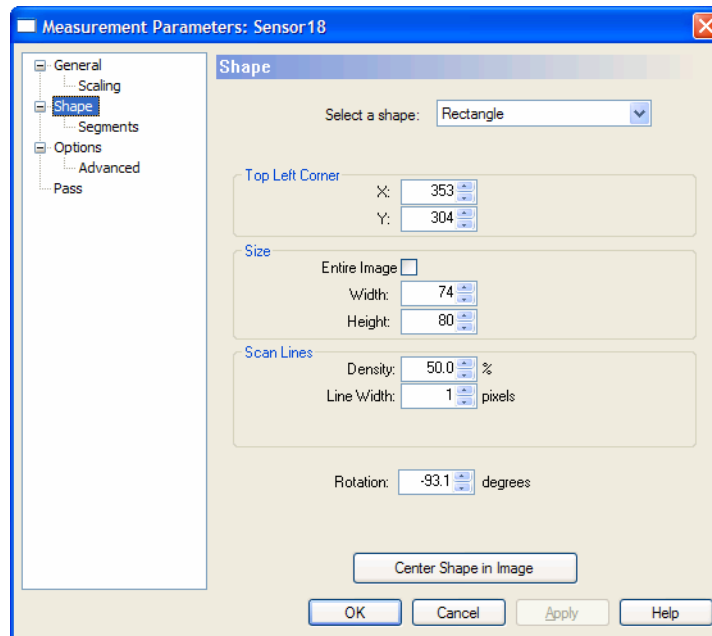


Figure 4-8: Measurement Rectangle Shape Tab

Rectangle (Shape ID = 7)			
Parameters	Data	Description	Type
Polygon Flag	1	Constant. All rectangular shapes are polygons.	UINT8
Area/Contour Shape Flag ¹	0	Contour-based	UINT8
	1	Area-based	
Rectangle Flag	1	Shape is always a rectangle	UINT8
Scan Line Width	1	Constant	FLOAT
Scan Line Width (again)	1	Constant	FLOAT
Scanning Density (percentage)	0 ... 100	Scanning density. Default is 50	FLOAT
Number of Segments ₂	0 +/- max	Some SoftSensors can be segmented; namely, the Positioning Line Fit and the Measure Across Area. A value of 0 here indicates no segmentation. Default is 0.	INT32
Start Percentage of First Segment ²	0 +/- max	Starting point of the first segment as a percentage of a full length.	FLOAT
End Percentage of First Segment ²	0 +/- max	End point of the first segment as a percentage of a full length.	FLOAT
...	...	...	...
Start Percentage of Segment N ²	0 +/- max	Starting point of the N th segment as a percentage of a full length.	FLOAT
End Percentage of Segment N ²	0 +/- max	End point of the N th segment as a percentage of a full length.	FLOAT
Full Screen Flag	0	Partial screen	UINT8
	1	Full screen	

Number of Points	5	Constant. Number of points is always 5. The last point is the same as the first.	INT32
First Point. X Coordinate	0 +/- max	X Coordinate of the starting point of the first segment.	FLOAT
First Point. Y Coordinate	0 +/- max	Y Coordinate of the starting point of the first segment	FLOAT
...	...	...	...
Fifth Point. X Coordinate	0 +/- max	X Coordinate of the last point. For a closed shape (polygon). This will be the same as the X Coordinate for the first point.	FLOAT
Fifth Point. Y Coordinate	0 +/- max	Y Coordinate of the last point. For a closed shape (polygon). This will be the same as the Y Coordinate for the first point.	FLOAT
Learn region to Search region X offset or Calibration Box Point 1. X Coordinate ³	0+/- max	For Fixed-Font Reader Only. Upper-left Corner of the character in the Calibration Box. X Coordinate.	FLOAT
Learn region to Search region Y offset or Calibration Box Point 1. Y Coordinate ³	0+/- max	For Fixed-Font Reader Only. Upper-left Corner of the character in the Calibration Box. Y Coordinate.	FLOAT
...	...	...	...
Calibration Box Point 4. X Coordinate ³	0+/- max	For Fixed-Font Reader Only. Lower-left Corner of the character in the Calibration Box. X Coordinate.	FLOAT
Calibration Box Point 4. Y Coordinate ³	0+/- max	For Fixed-Font Reader Only. Lower-left Corner of the character in the Calibration Box. Y Coordinate.	FLOAT

Table 4-14: Rectangle

¹: Most of these SoftSensors are area-based. That is, they inspect inside their region of interest. This includes the Filter, Area Positioning and Counting, Pixel Counting Rectangle, 1-D Rectangular Code Reader, 2-D Rectangular Code Reader, OCR, Defect Detection and Color Monitoring SoftSensors. Other SoftSensors are contour-based, such as, the Positioning Line Fit and Measure Across Area SoftSensors.

²: In most cases, the Number of Segments is 0. This means the shape is not segmented. In these cases, the start and end percentages for segments are not included. If the shape is in fact segmented there must be a pair of start and end percentages as a fraction of the full length for each segment.

³: For SoftSensors that allow the learning of a pattern, object, color, etc within a defined learn region, the first two of the similarly shaded values correspond to the X and Y distance between the limits of the learn region and the search regions to each side. This is the case for Area Positioning, Area Counting and Pixel Counting Rectangle SoftSensors. No other parameters are included for these SoftSensors after the mentioned offsets are included. For the OCR Rectangular Reader SoftSensor these two values and 6 more that follow them represent the X and Y coordinates of four points that form the calibration box used for the learning phase by the Fixed Font Reader task. Even though the Trainable Font Reader task does not need this calibration box, the parameters need to be included. For a third group of SoftSensors, none of

these similarly shaded parameters (neither the Learn region to Search region X and Y offsets nor the calibration box points' X and Y coordinates) are included at all. This applies to the Filter, Positioning Line Fit, Measure Across Area, 1-D Code Reader, 2-D Code Reader, Defect Detection and Color Monitoring SoftSensors. Refer to the examples below for a more detailed description.

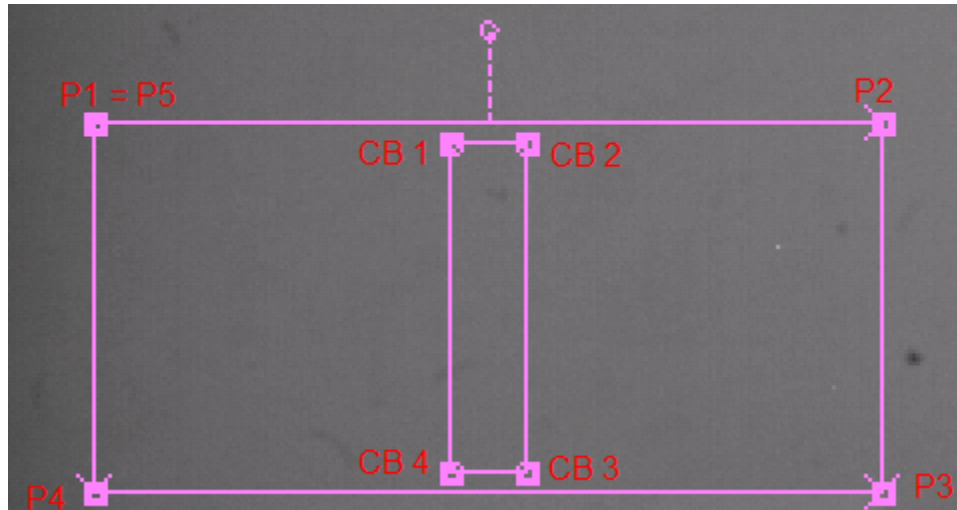


Image 4-3: Fixed Font OCR Rectangle

Terminal Example - This sets a Fixed Font OCR Reader sensor (see Image 2-3)

```
? #SP 1 0 7 0 -1 0 1 0 1 1 1 1 1 50 0 0 5 145 160 365 160 365 280 145 280 145 160 245
165 270 165 270 275 245 275
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 7> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Polygon?: 1> <Area/Contour?: 1> <Rectangle?: 1> <Width
(Not really applicable here but needs to be included): 1> <Width (Again): 1> <Scanning
Density: 50> <Number of Segments: 0> <Full frame flag: 0> <Number of Points: 5>
<Point 1. X Coordinate: 145> <Point 1. Y Coordinate: 160> <Point 2. X Coordinate: 365>
<Point 2. Y Coordinate: 160> <Point 3. X Coordinate: 365> <Point 3. Y Coordinate: 280>
<Point 4. X Coordinate: 145> <Point 4. Y Coordinate: 280> <Point 5 (same as Point 1 to
close the shape). X Coordinate: 145> <Point 5 (same as Point 1 to close the shape). Y
Coordinate: 160> <Calibration Box Point 1. X Coordinate: 245> <Calibration Box Point 1.
Y Coordinate: 165> <Calibration Box Point 2. X Coordinate: 270> <Calibration Box Point
2. Y Coordinate: 165> <Calibration Box Point 3. X Coordinate: 270> <Calibration Box
Point 3. Y Coordinate: 165> <Calibration Box Point 4. X Coordinate: 245> <Calibration
Box Point 4. Y Coordinate: 275> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 7 0 -1 0 1 0 1 1 1 1 1 50 0 0 5 145 160 365 160 365 280 145
280 145 160 245 165 270 165 270 275 245 275");
```

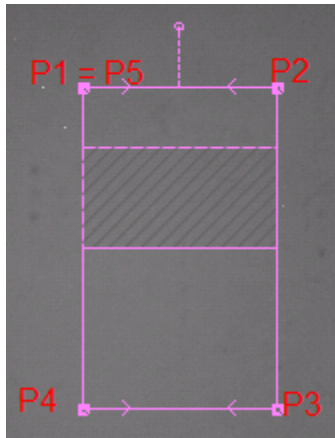


Image 4-4: Measure in Area Rectangle

Terminal Example - This sets a Measure in Area sensor (see Image 2-4)

```
? #SP 1 0 7 0 -1 0 1 0 1 0 1 1 1 50 2 0 0.1875 0.5 1 0 5 340 140 435 140 435 300 340
300 340 140
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 7> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Polygon?: 1> <Area/Contour?: 1> <Rectangle?: 1> <Width
(Not really applicable here but needs to be included): 1> <Width (Again): 1> <Scanning
Density: 50> <Number of Segments: 2> <Segment 1 Start Perc: 0> <Segment 2 End
Perc: 0.1875> <Segment 2 Start Perc: 0.5> <Segment 2 End Perc: 1> <Full Screen
Flag: 0> <Number of Points: 5> <Point 1. X Coordinate: 340> <Point 1. Y Coordinate:
140> <Point 2. X Coordinate: 435> <Point 2. Y Coordinate: 140> <Point 3. X Coordinate:
435> <Point 3. Y Coordinate: 300> <Point 4. X Coordinate: 340> <Point 4. Y Coordinate
300> <Point 5 (same as Point 1 to close the shape). X Coordinate: 340> <Point 5 (same
as Point 1 to close the shape). Y Coordinate: 140> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 7 0 -1 0 1 0 1 0 1 1 1 50 2 0 0.1875 0.5 1 0 5 340 140 435
140 435 300 340 300 340 140");
```

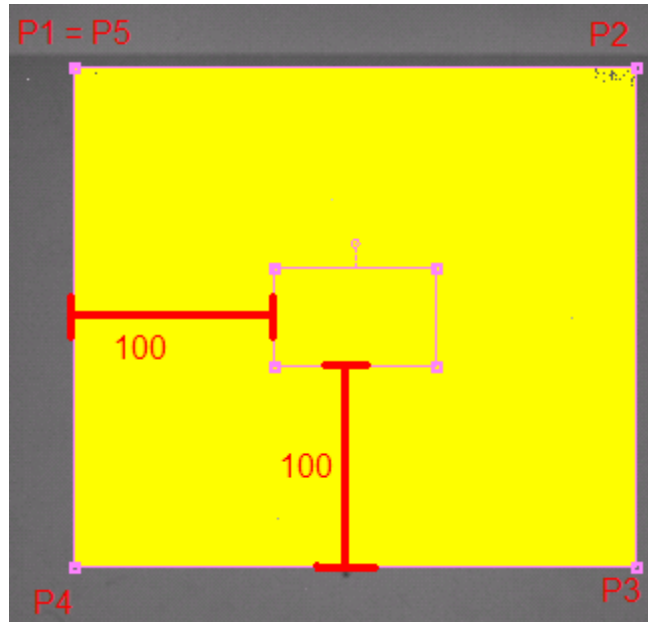


Image 4-5: Pixel Counting Rectangle

Terminal Example - This sets a Pixel Counting sensor (see Image 2-5)

```
? #SP 1 0 7 0 -1 0 1 0 1 1 1 1 1 50 0 0 5 100 150 400 150 400 350 100 350 100 150 100 100
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 7> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Polygon?: 1> <Area/Contour?: 1> <Rectangle?: 1> <Width
(Not really applicable here but needs to be included): 1> <Width (Again): 1> <Scanning
Density: 50> <Number of Segments: 0> <Full frame flag: 0><Number of Points: 5>
<Point 1. X Coordinate: 100> <Point 1. Y Coordinate: 150> <Point 2. X Coordinate: 400>
<Point 2. Y Coordinate: 150> <Point 3. X Coordinate: 400> <Point 3. Y Coordinate: 350>
<Point 4. X Coordinate: 100> <Point 4. Y Coordinate 350> <Point 5 (same as Point 1 to
close the shape). X Coordinate: 100> <Point 5 (same as Point 1 to close the shape). Y
Coordinate: 150> <X Offset for Search Region: 100> <Y Offset for Search Region: 100>
<CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 7 0 -1 0 1 0 1 1 1 1 1 50 0 0 5 100 150 400 150 400 350 100
350 100 150 100 100");
```

Parallelogram

The OCR reader is the only sensor that uses the parallelogram shape. A screen shot of the OCR Shape tab is shown below.

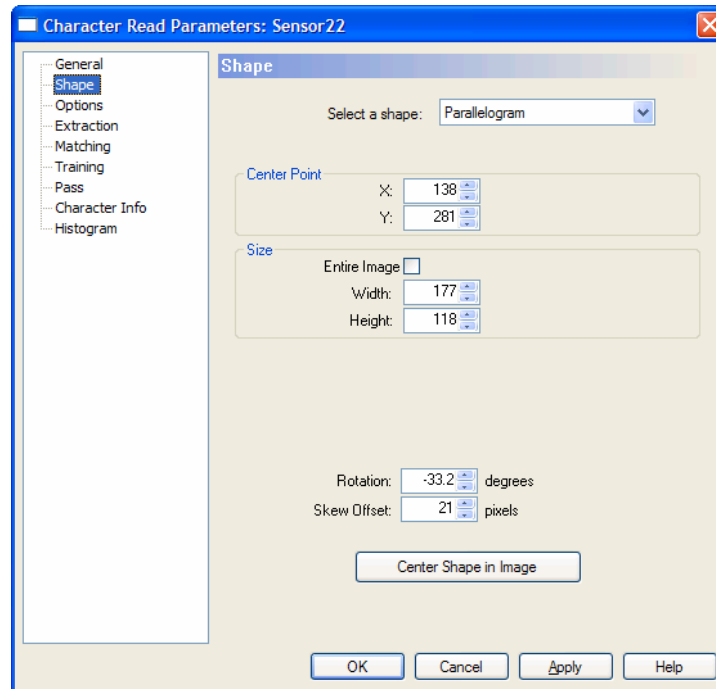


Figure 4-9: Parallelogram Shape Tab

Parallelogram (Shape ID = 9)			
Parameters	Data	Description	Type
Polygon?	1	Only the OCR Reader has this shape. It is always a polygon: that is, the shape is closed. This means that the first point is the same as the last one.	UINT8
Area/Contour Shape Flag	1	Only one SoftSensor has this shape. It is the OCR Parallelogram. This SoftSensor is area-based.	UINT8
Rectangle?	0	Shape is not a rectangle.	UINT8
	1	Shape is a rectangle.	
Line Width	1	Constant.	FLOAT
Number of Points ¹	5	Constant. Number of points for parallelogram shape is always 5.	INT32
First Point. X Coordinate ¹	0 +/- max	X Coordinate of the starting point of the parallelogram.	FLOAT
First Point. Y Coordinate ¹	0 +/- max	Y Coordinate of the starting point of the parallelogram.	FLOAT
...	...	...	...
Fifth Point. X Coordinate ¹	0 +/- max	X Coordinate of the last point. This will be the same as the X Coordinate for the first point.	FLOAT
Fifth Point. Y Coordinate ¹	0 +/- max	Y Coordinate of the last point. This will be the same as the Y Coordinate for the first point.	FLOAT
Calibration Box Point 1. X Coordinate ²	0 +/- max	For Fixed-Font Reader Only. Upper-left Corner of the character in the Calibration	FLOAT

		Box. X Coordinate.	
Calibration Box Point 1. Y Coordinate ²	0+/- max	For Fixed-Font Reader Only. Upper-left Corner of the character in the Calibration Box. Y Coordinate.	FLOAT
...	...	...	...
Calibration Box Point 4. X Coordinate ²	0+/- max	For Fixed-Font Reader Only. Lower-left Corner of the character in the Calibration Box. X Coordinate.	FLOAT
Calibration Box Point 4. Y Coordinate ²	0+/- max	For Fixed-Font Reader Only. Lower-left Corner of the character in the Calibration Box. Y Coordinate.	FLOAT

Table 4-15: Parallelogram

¹: The number of points is always 5 for this shape. The first and fifth points are always the same. This shape only exists for the OCR Reader SoftSensor. The points are defined according to the direction of the reading operation. In this case, the second point of the parallelogram is located in the direction of the reading with respect to the first point. That is, if the reading is from left to right (most common case) on the image, the second point will be to the right of the first point of the parallelogram.

²: These 8 values represent the X and Y coordinates of the calibration box. The calibration box's sides need to be parallel to the corresponding side of the search region. If the user enters points that violate this principle, the firmware will generate a calibration box that best accommodates the user's input but still holds the principle. The calibration box is taken in the direction of reading, starting at the upper-left corner of the character in the box. Even though these parameters are not used by the OCR SoftSensor when the "Trainable OCR" task is used they need to be included. The four points are illustrated below.

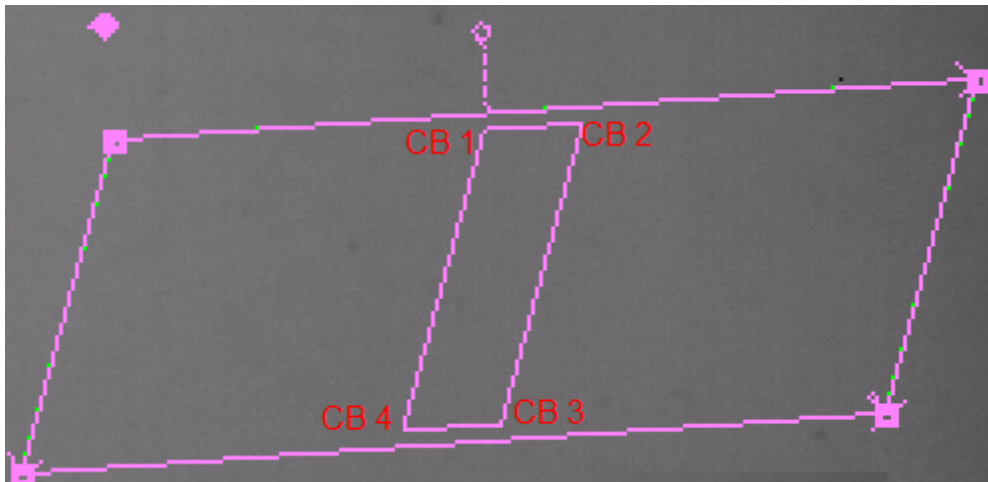


Image 4-6: Fixed Font OCR Parallelogram

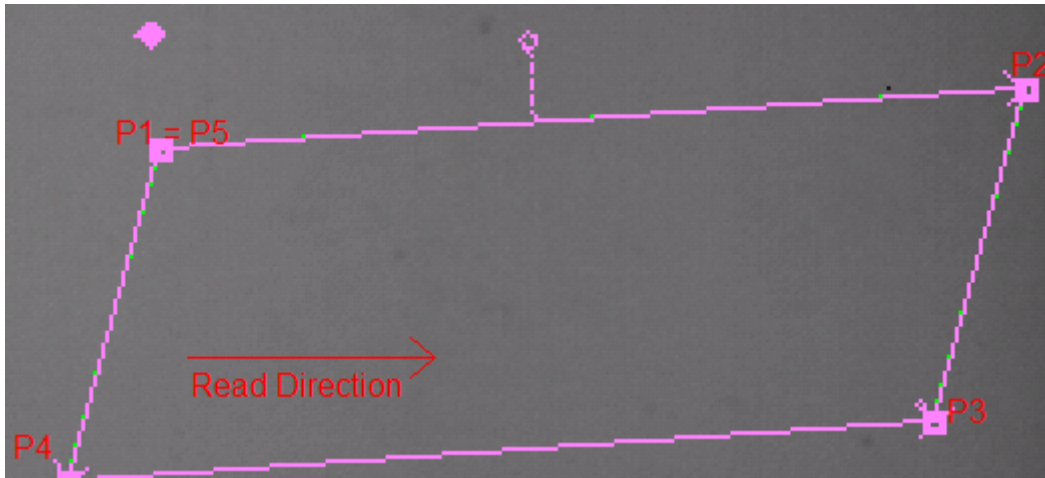


Image 4-7: Trainable OCR Parallelogram

Terminal Example - This sets a Trainable OCR Sensor

```
? #SP 1 0 9 0 -1 0 1 0 1 1 0 1 5 375 255 595 240 570 320 355 335 375 255 460 250 480
250 485 325 465 325
```

Terminal Format

```
#SP<SoftSensor ID:1> <Parameter ID: 0> <Shape ID: 9> <Loading from file?: 0>
<Reference SoftSensor ID: -1> <Defined in Transformed Plane?: 0> <Rotatable shape?:
1> <Aligned with Image?: 0> <Polygon?: 1> <Area/Contour?: 1> <Rectangle?: 0> <Width
(Not really applicable here but needs to be included): 1> <Number of Points: 5> <Point 1.
X Coordinate: 375> <Point 1. Y Coordinate: 255> <Point 2. X Coordinate: 595> <Point 2.
Y Coordinate: 240> <Point 3. X Coordinate: 570> <Point 3. Y Coordinate: 320> <Point 4.
X Coordinate: 355> <Point 4. Y Coordinate: 335> <Point 5 (same as Point 1 to close the
shape). X Coordinate: 375> <Point 5 (same as Point 1 to close the shape). Y Coordinate:
255> <Calibration Box Point 1 (Not used here but needs to be included). X Coordinate:
460> <Calibration Box Point 1 (Not used here but needs to be included). Y Coordinate:
250> <Calibration Box Point 2 (Not used here but needs to be included). X Coordinate:
480> <Calibration Box Point 2 (Not used here but needs to be included). Y Coordinate:
250> <Calibration Box Point 3 (Not used here but needs to be included). X Coordinate:
485> <Calibration Box Point 3 (Not used here but needs to be included). Y Coordinate:
325> <Calibration Box Point 4 (Not used here but needs to be included). X Coordinate:
465> <Calibration Box Point 4 (Not used here but needs to be included). Y Coordinate:
325> <CR><LF>
```

Script Example

```
MySensorVar.SetParams("0 9 0 -1 0 1 0 1 1 0 1 5 375 255 595 240 570 320 355 335 375
255 460 250 480 250 485 325 465 325");
```

Threshold

This section contains information on all the available thresholds in the system. Not all sensors use thresholds. Those that do may not use all the available thresholds, so threshold information specific to each sensor is also included in later in this manual. The screenshot below shows a sample Threshold Options tab.

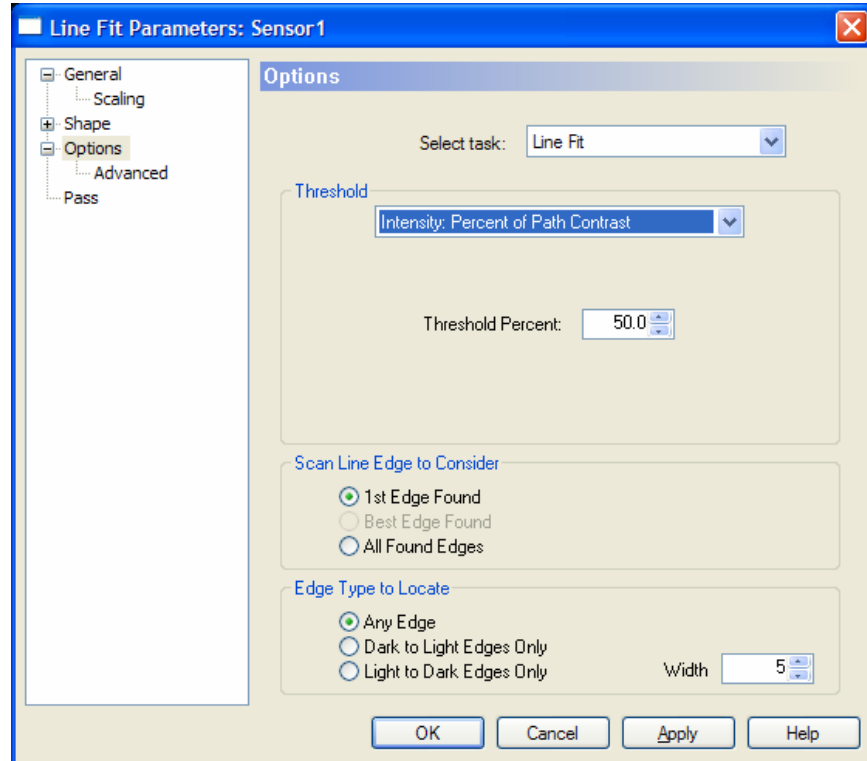


Figure 4-10: Threshold Options Tab

Table 2 – 17 below shows the parameters for all available threshold options. Each threshold type is selected using a SensorID, ParameterID (251), ThresholdID and then any parameters associated with that threshold type.

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16

1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
10 Local Difference			
Threshold ID	10	Local Difference	INT8
% Contrast Threshold	0...10000	Threshold level*100	UINT16
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16
4 Adaptive Intensity			
Threshold ID	4	Adaptive Threshold	INT8
Threshold contrast	0...10000	Threshold value*100	UINT16
Window Size	0...100	Window in pixels	INT16
Min Window Contrast	1...10000	Contrast value*100	UINT16
64 OCR Computed			
Threshold ID	4	OCR Computed	INT8
Threshold Value	5000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Checkbox	UINT8
Block Size	4	Contrast value*100	UINT8
65 OCR Linear			
Threshold ID	65	OCR Linear	INT8
Threshold Value	0...5000...10000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Don't reduce	UINT8
	1	Reduce	
Block Size	4	Contrast value*100	UINT8
66 OCR Nonlinear			
Threshold ID	66	OCR Nonlinear	INT8
Threshold Value	0...5000...10000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Don't reduce	UINT8

	1	Reduce	
Block Size	0... 4 ...50	Number of areas	UINT8

Table 4-16: Thresholds

Terminal Example - This sets the Counting Line Sensor's threshold to Fixed Intensity, 25%.

```
? #SP 1 251 0 2500
```

Script Example

```
MySensorVar.SetParams("251 0 2500");
```

Inspection Outputs

The Inspection Outputs table in Intellect is opened by double clicking on the row for 'Inspection Results' in the Result Table. Here you can use Boolean logic to combine individual sensor inspection results to create a final product inspection result or User output result. For each sensor (rows), the columns for Pass, User1, User2...User16 can be set to Pass, Fail or nothing (ignore). How each sensor is used to determine an output's result can be set using the SP command by sending two bit patterns whose bits represent the Pass/Fail/Ignore states in each row of this table. The first bit pattern has a 1 at each bit location that should be set to PASS. The second bit pattern has a 1 at each bit location that should be set to FAIL. If a bit location is not set to either PASS or FAIL, then it is left empty which means the sensor is ignored when determining the Boolean result for that output. The two Pass and Fail bit patterns are mutually exclusive. If a 1 is set for the same bit in each, the Pass value will be used.

Figure 4-11: Inspection Outputs

\$241 - Set User Output Bits			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	INT16
Parameter ID	241	User Output Bits	UINT8
Pass Bits	0...131071	On bits representing Pass pattern	UINT32
Fail Bits	0...131071	On bits representing Fail pattern	UINT32

Table 4-17: Inspection Outputs

Terminal Example - This sets the pattern for Sensor 2 in the above screen shot (Ignore, Pass, Pass, Fail).

```
? #SP 2 241 6 8 // PASS pattern 01100000... (Binary 6, bit 0 on left)
// FAIL pattern 00010000... (Binary 8, bit 0 on left)
```

Script Example

```
MySensorVar.SetParams("241 6 8");
```

5. Sensor Specific Parameters

This section describes the parameters that are specific to each sensor. This includes parameters used to tune each sensor to your specific application like model learning and searching options, edge counting and positioning parameters, filter choices, code types, pass/fail settings, etc... Each section contains a table indicating all the parameter IDs returned in the SQ response for each sensor and a description of each. Information on the common parameters is detailed in the previous section. Threshold information specific to each sensor is detailed in this chapter.

Preprocessing

Preprocessing Filter

A preprocessing filter is used as an image reference by other sensors. Each filter sensor may contain up to 10 filters under the umbrella of an aggregate filter. This section describes parameter IDs 2 and 3, filter options and the filter to display.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data
\$	2	Filter options
\$	3	Filter to display

Parameter ID 2, filter options, contains the majority of the filter information. This parameter contains information on the selected filter(s) and the associated parameters. Below is a screen shot of the filter options tab.

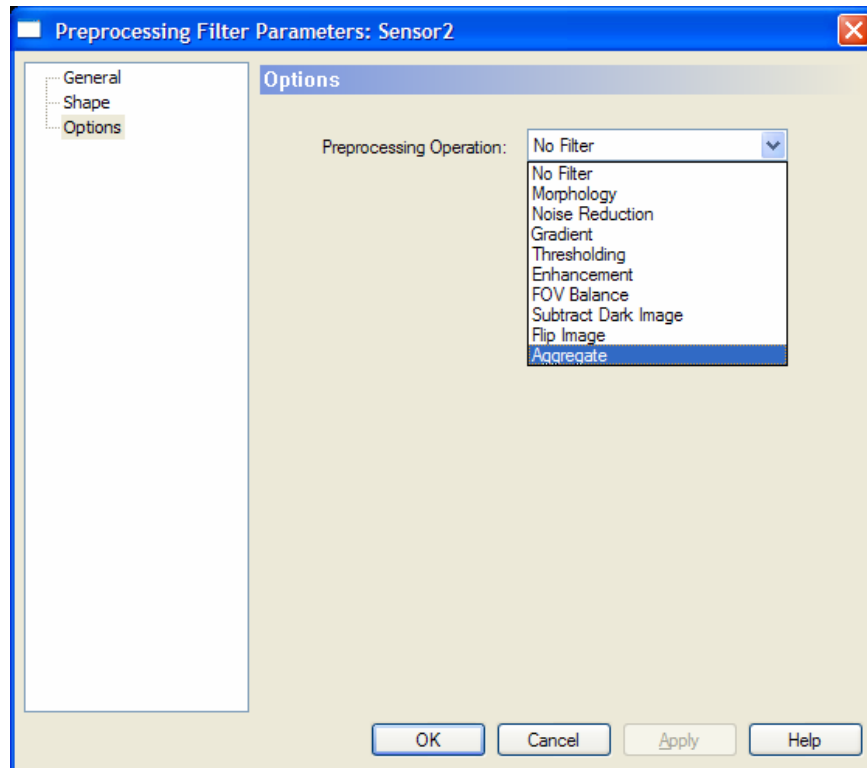


Figure 5-1: Filter Options Tab

Select a filter and set its parameters using the following table. The filter to be displayed

\$2 – Filter Options			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Sensor Options	UINT8
Filter Type	1-26	See Filter Tables	UINT8
Filter Parameters		See Filter Tables	

Table 5-1: Set Sensor Options

\$3 - Filter to Display			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Filter to Display	UINT8
Filter # to Display	0...9	Filter index	UINT8

Table 5-2: Filter to Display

Filter Tables

Each filter is grouped in the following tables according to its placement in the filter options tab.

Morphology Filters			
11 – Dilate			
Filter Type	11	Dilate	UINT8
Mask Area Size – Columns	0-15	Index of mask size	UINT8
Mask Area Size – Rows	0-15	Index of mask size	UINT8
10 – Erode			
Filter Type	10	Erode	UINT8
Mask Area Size – Columns	0-15	Index of mask size	UINT8
Mask Area Size – Rows	0-15	Index of mask size	UINT8
12 – Open			
Filter Type	12	Open	UINT8
Mask Area Size – Columns	0-15	Index of mask size	UINT8
Mask Area Size – Rows	0-15	Index of mask size	UINT8
13 – Close			
Filter Type	13	Close	UINT8
Mask Area Size – Columns	0-15	Index of mask size	UINT8
Mask Area Size – Rows	0-15	Index of mask size	UINT8
14 – Fill Dark Holes			
Filter Type	14	Fill Dark Holes	UINT8
Filter Options		None	
15 – Fill Light Holes			
Filter Type	15	Fill Light Holes	UINT8
Filter Options		None	

Table 5-3: Morphology Filters

Noise Reduction Filters			
4 – Low Pass			
Filter Type	4	Low Pass	UINT8
Filter Options		None	
7 – Median			
Filter Type	7	Median	UINT8
Filter Options		None	

Table 5-4: Noise Reduction Filters

Gradient Filters			
1 – Full			
Filter Type	1	Full	UINT8
Filter Options		None	
3 – Vertical			
Filter Type	3	Vertical	UINT8

Filter Options		None	
2 – Horizontal			
Filter Type	2	Horizontal	UINT8
Filter Options		None	
5 - High Pass			
Filter Type	5	High Pass	UINT8
Filter Options		None	
8 – Kirsh			
Filter Type	8	Kirsh	UINT8
Filter Options		None	

Table 5-5: Gradient Filters

Thesholding Filters			
20 - Color Filter			
Filter Type	20	Color Filter	UINT8
Pixel Counting Sensor	0...32,768	Sensor ID	INT32
Color Name	0...	Index in color list	UINT8
Inverted	0	Not Inverted	UINT8
	1	Inverted	UINT8
22 - Edge Filter			
Filter Type	22	Edge Filter	UINT8
Edge Fuzziness	0 - 10	# Pixels	UINT8
Gradient Threshold	0 - 100	% Threshold	UINT8
Inverted	0	Not Inverted	UINT8
	1	Inverted	UINT8
23 - Blob Filter			
Filter Type	23	Blob Filter	UINT8
Threshold	0 - 100	Threshold value	UINT8
Unused	-1	Default	
Pixel Connectivity	0	8 way connectivity	UINT8
	1	4 way connectivity	UINT8
Enable Boundary Blobs	0	Enabled	UINT8
	1	Disabled	UINT8
Minimum Size	-1	Disabled	INT32
	0...	Size value	INT32
Maximum Size	-1	Disabled	INT32
	0...	Size value	INT32
Blob Threshold Type	0	Dark Blobs	UINT8
	1	Light Blobs	UINT8
Morphology Filter 1 Type	-1	None	INT8
	0	Dilate	INT8
	1	Erode	INT8
	2	Open	INT8
	3	Close	INT8

Width	0 - 10	Filter column size	UINT8
Height	0 - 10	Filter row size	UINT8
Morphology Filter 2 Type	-1...3	see Filter 1	INT8
Width	0 - 10	Filter column size	UINT8
Height	0 - 10	Filter row size	UINT8
Morphology Filter 3 Type	-1...3	see Filter 1	INT8
Width	0 - 10	Filter column size	UINT8
Height	0 - 10	Filter row size	UINT8
Morphology Filter 4 Type	-1...3	see Filter 1	INT8
Width	0 - 10	Filter column size	UINT8
Height	0 - 10	Filter row size	UINT8
Morphology Filter 5 Type	-1...3	see Filter 1	INT8
Width	0 - 10	Filter column size	UINT8
Height	0 - 10	Filter row size	UINT8
Max # blobs	2000	Not exposed to filter	UINT16

Table 5-6: Thresholding Filters

Enhancement			
6 – Sharpen			
Filter Type	6	Sharpen	UINT8
Filter Options		None	
9 – Convolution			
Filter Type	9	Convolution	UINT8
Matrix Width	1...5	Column size	UINT8
Matrix Height	1...5	Row size	UINT8
Scaling	1...127	Matrix divisor	UINT8
Matrix Values	-128...127	25 matrix cell values	INT8
17 – Image Domain			
Filter Type	17	Image Domain	UINT8
Color Plane	1	Red	UINT8
	2	Green	
	3	Blue	
	4	Intensity	
	5	Custom ¹	
	6	Gray Scale ²	
Custom Color Plane Values ¹	-128..127	Red	INT8
	-128..127	Green	
	-128..127	Blue	
21 - Optical Density			
Filter Type	21	Optical Density	UINT8
Filter Options		None	

24 - Histogram Equalization			
Filter Type	24	Histogram Equalization	UINT8
Filter Options		None	

Table 5-7: Enhancement Filters

¹Three values are only exposed when the Color Plane is set to Custom (5, only available on color cameras). If the Color Plane is 1, 2, 3, 4 or 6, only one value will be exposed which represents the gain in that plane.

18 - FOV Balance			
Filter Type	18	FOV Balance	UINT8
Filter Options		None	

Table 5-8: FOV Balance Filter

19 - Dark Image Subtraction			
Filter Type	19	Dark Image Sub.	UINT8
Filter Options		None	

Table 5-9: Dark Image Subtraction Filter

26 - Flip Image			
Filter Type	26	Flip Image	UINT8
Direction	1	Vertical	UINT8
	2	Horizontal	UINT8
	3	Both	UINT8

Table 5-10: Flip Image Filter

16 - Aggregate Filter			
Filter Type	16	Aggregate	UINT8
Count	1...10	# of Filters	UINT8
Sub-Filter Type	0...26	From above tables	UINT8
Sub-Filter Options	...	From above tables	

Table 5-11: Aggregate Filter

Terminal Example – Set filter as an aggregate of a horizontal gradient followed by a 3x3 erode.

```
#SP 1 2 16 2 2 10 1 1
```

Script Example - Set filter as an aggregate of a horizontal gradient followed by a 3x3 erode.

```
MySensorVar.SetParams("2 16 2 2 10 1 1");
```

Positioning

This section contains descriptions of the parameters for the Position along Line/Arc, Area Positioning, Line Fit and Circle Fit sensors. Several of these sensors have the option of using different algorithms which have unique parameters. Use only the parameters that match the chosen algorithm.

Position Along Line/Arc

The Position Along Line/Arc sensor has 2 algorithm options: Edge Find and Pattern Find. In this case their parameters are similar and thus share the same parameter ID, so the parameter table is split into two sections. Use only the section for the algorithm you are using.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data
\$	2	Option/ Advanced option parameters
\$	3	Pass/Fail criteria
\$	10	Sensor pattern
\$	45	Relearn pattern

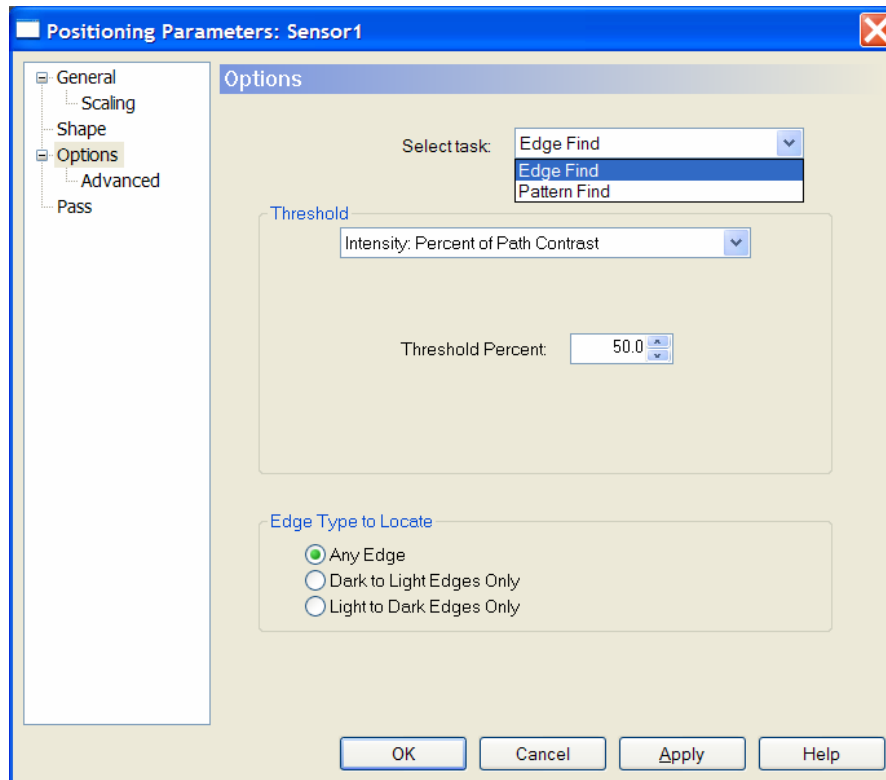


Figure 5-2: Positioning Line Options Tab

The Edge Find and Pattern Find options use different parameters so use only the parameters that apply to each algorithm. The pattern find option chooses it's own threshold so no threshold is exposed in the Intellect UI. The below thresholds apply only to the Edge Find option.

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8

Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16

Table 5-12: Sensor Threshold

\$10 - Sensor Pattern			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	10	Sensor Pattern	UINT8
Number of Pixels	1...30000	Pixels in Pattern	INT8
Pixel Intensity 1	0...255	Pixel Value	UINT8
...	0...255	Pixel Value	UINT8
Pixel Intensity X	0...255	Pixel Value	UINT8

Table 5-13: Sensor Pattern

\$2 – Set Option and Advanced Parameters			
SP <sensor ID> <parameter ID> <Data>			
Edge Finding Option Parameters			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Options and Advanced Parameters	UINT8
Digital Relearn Input Bit	-1	Virtual input bit	INT8
Algorithm Type	1	Edge Finding	UINT16
Edge Type to Find	0	Any	UINT8
	1	Dark to Light	
	2	Light to Dark	
Edge Index to Find	0...1000	Index	UINT8
Edge Width	0...1000	Width in pixels	INT8
Pattern Finding Option Parameters			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Options and Advanced Parameters	UINT8
Digital Relearn Input Bit	-1	Virtual input bit	INT8
Algorithm Type	2	Pattern Finding	UINT8
Correlation Type	0	Not Used	UINT8
Minimum Score	0...100	Percent Matching	UINT8
Match to Used	0	Best	INT8
	1	First	

Table 5-14: Option and Advanced Parameters

\$3 - Set Sensor Pass Conditions			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Pass Conditions	UINT8
Minimum Contrast	0...10,000	Milliseconds	INT16
Minimum Offset	-1...2000	Offset from start	FLOAT
Maximum Offset	-1...2000	Offset from start	FLOAT

Table 5-15: Pass Conditions

\$45 - Relearn Pattern			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Relearns pattern on next inspection.	UINT8

Table 5-16: Relearn Pattern

Position in Area

For instance the Area Positioning sensor has 3 choices: Blob, Object Locate, and Pattern Match. In this case the parameters for each algorithm have a unique parameter ID. Use only the parameters for the chosen algorithm.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data
\$	2	Digital Relearn Bit and Algorithm Type
\$	3	Experimental (ignore)
\$	4	Pattern Match matching parameters
\$	5	Blob Generator Parameters
\$	*7	Model: Binary Information about Object Model
\$	8	Object Locate Positioning Parameters
\$	9	Blob Positioning Parameters
\$	*10	Model: Learned Blob Model values for Selection Criteria
\$	*11	Command: Add to Object Locate

\$	*12	Command: Remove from Object Locate Model
\$	14	Not Used (ignore)
\$	*20	Model: Pattern Match Binary Pattern information
\$	*45	Command: Relearn Pattern Match on Next Inspection
\$	*46	Command: Relearn Blob

*These parameters may not be included in the sensor query response depending on which algorithm is currently selected. \$11, \$12, \$45, \$46 are commands and never appear in the response.

\$2 - Digital Relearn and Algorithm Type			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Digital Relearn and Algorithm Type	UINT8
Digital Relearn Input	-1...63	Input bit	INT8
Select Task	0	Object Locate	UINT8
	1	Pattern Match	
	2	Blob	

Table 5-17: Digital Relearn and Algorithm Type

\$3 – Experimental algorithm (ignore)			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Experimental Algorithm	UINT8
n/a	0.9	Default	n/a
n/a	3	Default	n/a
n/a	-15	Default	n/a
n/a	15	Default	n/a
n/a	10	Default	n/a
n/a	100	Default	n/a
n/a	100	Default	n/a

Figure 5-3: Experimental (ignore)

\$4 – Pattern Matching Options			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	4	Pattern Matching Options	UINT8
Minimum Matching %	0...100	Percent	INT16
Not Used	0	Default	UINT16
Not Used	2	Default	INT32

Not Used	10	Default	INT32
----------	----	---------	-------

Figure 5-4: Pattern Matching Options

\$5 - Blob Generation Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	5	Blob Generator Parameters	UINT8
Minimum Contrast	0...100	Contrast%	INT16
Pixel Connectivity	0	8-way	FLOAT
	1	4-way	
Enable Boundary Blobs	0	Boundary blobs disabled	FLOAT
	1	Boundary blobs enabled	
Minimum Blob Size	-1..400000	Pixel area	INT16
Maximum Blob Size	-1..400000	Pixel area	INT16
Blob Color	0	Dark	FLOAT
	1	Light	
Preprocessing Filter 1...5	-1	None	INT32
	0	Dilation	
	1	Erosion	
	2	Open	
	3	Close	
Filter 1...5 Height	1	Pixels	INT32
Filter 1...5 Width	1	Pixels	INT32
Max. Number of Blobs	-1...32767	Max found blobs	INT32

Figure 5-5: Blob Generation Parameters

\$9 – Blob Options			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	9	Blob Options	UINT8
Minimum Match Score	0...100	Percent	INT16
Not Used	0	Default	UINT8
Blob Features to Use	1... 2800	Binary	UINT16
Angle Format	0	Principal Axis 180°	UINT8
	1	Principal Axis 360°	
	2	Max Point 360°	
	3	None	
Sort Criteria	10	None	UINT8
	2	Position	

	0	Area	
	1	Perimeter	
	8	Radius	
	5	Eccentricity	
	6	Compactness	
	4	Intensity	

Figure 5-6: Blob Options

\$10 -Learned Blob Model Values			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	10	Learned Blob Model	UINT8
Area	0-10000000	Pixels	FLOAT64
Perimeter	0-10000000	Pixels	FLOAT64
Average Intensity	0-255	Grayscale	UINT8
Eccentricity	0...1.0	Eccentricity / 100	FLOAT64
Compactness	0...1.0	Compactness / 100	FLOAT64
Max Point Radius	0...4000	Pixels	FLOAT64
Average Red Intensity	0-255	Red Plane	UINT8
Average Green Intensity	0-255	Green Plane	UINT8
Average Blue Intensity	0-255	Blue Plane	UINT8

Figure 5-7: Learned Blob Model

\$8 – Object Locate Options/Parameters/Pass			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	8	Object Locate Options	UINT8
Show Live Edge Image	0	Disabled	INT32
	1	Enabled	
Accurate Position +/-	0...100	Accurate position*100	INT32
Max Object Count	10000	Max Objects Found	INT32
Pick Point Offset X	0...1000	Pixels	INT32
Pick Point Offset Y	0...1000	Pixels	INT32
Pick Point Offset Angle	0 +/-180	Degrees	INT32
Min Objects	0...10000	Pass Criteria	INT32
Max Objects	0...10000	Pass Criteria	INT32
Locate: Pixel Distance	1...10	Pixels	INT32
Min Match Score	0...100	%	INT32

First Match / Best Match	1...5	Slider Bar	INT32
Min Distance Between Object	0...1000	Pixels	INT32
Min Angle Between Objects	0...180	Angle	INT32
Accurate Position	0	Disabled	UINT8
	1	Enabled	
Mark Object	0	Disabled	UINT8
	1	Enabled	
Translation Only	0	Disabled	UINT8
	1	Enabled	
Limit Rotation Value	0...180	Degrees	INT32
Limit Scale Value	0...20	Percent	INT32
Limit Rotation	0	Disabled	UINT8
	1	Enabled	
Not Used	-1	Default	INT32
Not Used	0	Default	INT8

Figure 5-8: Object Locate Options and Parameters

\$10 -Relearn Object Locate Model Command*			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	10	Relearn Object Locate Model	UINT8
not used	1	Default	INT32
Min Pixel Size	0...1000	Pixels	INT32
Min Gradient	0...200	Gray Scale Intensity Difference	INT32

Figure 5-9: Relearn Object Locate Model Command

\$7 -Object Locate Model Information			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	7	Object Locate Model	UINT8
Object Model Info	...	Multiple Binary	INT32

Figure 5-10: Object Locate Model Information

\$11 - Add to Object Locate Model Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	11	Adds to Model on Next Inspection	UINT8

Figure 5-11: Add to Object Locate Model Command

\$12 - Remove from Object Locate Model			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	12	Relearns Pattern on Next Inspection	UINT8

Figure 5-12: Remove from Object Locate Model Command

\$45 – Relearn Pattern Match on Next Inspection			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Relearns Pattern on Next Inspection	UINT8

Figure 5-13: Relearn Pattern Match Command

\$46 - Relearn Blob			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	46	Relearns Blob on Next Inspection	UINT8
Blob number to learn	0...1000	Index	UINT16

Figure 5-14: Relearn Blob Command

Circle/Line Fit

While the Circle Fit sensor has only one algorithm option, the Line Fit sensor has four algorithm options: Line Fit, Min Distance, Max Distance, Med Distance. The Line Fit option has unique pass conditions and one unique edge locating option. All other options are the same. Note that there are separate pass conditions for the Line Fit and Circle Fit sensors. Use only the parameters for the chosen sensor.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data
\$	1	Algorithm Type

\$	2	Set Option/Advanced Options Parameters
\$	6	Line Fit Pass Conditions
\$	7	Circle Fit Pass Conditions

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16

\$1 - Algorithm Type			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	1	Algorithm Type	UINT8
Algorithm Type	0	Line Fit/Circle Fit	INT8
	1	Min Distance (line fit only)	
	2	Max Distance (line fit only)	
	3	Median Distance (line fit only)	

\$2 - Set Option/ Advanced Options Parameters			
SP <sensor ID> <parameter ID> <Data>			

Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Options and Advanced Parameters	UINT8
Edge Width	1...5...200	Virtual input bit	UINT16
Edge Type to Locate	0	First	UINT8
	1	Best	
	2	All (Line Fit only)	
Scan Line Edge to Consider	0	Any	UINT8
	1	Dark to Light	
	2	Light to Dark	
Outlier Distance	.01...655	Pixels	INT32
Max Iterations	0...50		INT32
Use Hough Transform	0	Disabled	UINT8
	1	Enabled	
Hough Transform Resolution	0...10		INT32
Hough Transform Resolution	0...10	(Repeated)	INT32

\$6 – Line Fit Pass Conditions			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	6	Pass Conditions	UINT8
Minimum Contrast	0...10000	Contrast%*100	INT16
Minimum Distance	-1,0-2000	Offset from start	FLOAT
Maximum Distance	-1,0-2000	Offset from start	FLOAT
Minimum Relative Angle	-100, -90...90	-100 indicates disabled (Line Fit only)	FLOAT
Maximum Relative Angle	-100,-90...90	-100 indicates disabled (Line Fit only)	FLOAT
Maximum Straightness	-1,0...2000	Sum of Pixels (Line Fit only)	FLOAT

\$7 - Circle Fit Pass Conditions			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	7	Pass Conditions	UINT8
Minimum Contrast	0...10000	Contrast%*100	INT16
Minimum Radius	-1,0-2000	Offset from start	FLOAT
Maximum Radius	-1,0-2000	Offset from start	FLOAT

Maximum Straightness	-1,0-2000	Sum of Pixels	FLOAT
-------------------------	-----------	---------------	-------

Counting

This section contains descriptions of the parameters for the Position along Line/Arc, Area Positioning, Line Fit and Circle Fit sensors. Several of these sensors have the option of using different algorithms which have unique parameters. Use only the parameters for the algorithm you are using.

Count Along Line

The Count Along Line/Arc sensor also has 3 algorithm options: Edge Count, Feature Count and Pattern Count. In this case their parameters are similar and thus share the same parameter ID, so the parameter table is split into two sections. Use only the section for the algorithm you are using.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	*10	Model: Pattern Find pattern data
\$	0	Shape, Position Reference and Sensor Transform Data
\$	2	Option/ Advanced option parameters
\$	3	Pass/Fail criteria

* This parameter may not be included in the sensor query response depending on which algorithm is currently selected.

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251		UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16

1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16

Table 5-18: Thresholds

\$10 – Pattern Count Pixel Data			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	10		UINT8
Number of Pixels	1...30000	Pixels in Pattern	INT8
Pixel Intensity 1	0...255	Pixel Value	UINT8
...	0...255	Pixel Value	UINT8
Pixel Intensity X	0...255	Pixel Value	UINT8

Table 5-19: Pattern Count Pixel Data

\$2 – Set Option and Advanced Options Parameters			
SP <sensor ID> <parameter ID> <Data>			
Edge Counting Option Parameters			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2		UINT8
Digital Relearn Input Bit	-1	Not used	INT8
Algorithm Type	1	Edge Counting	UINT16
Edge Type to Find	0	Any	UINT8
	1	Dark to Light	
	2	Light to Dark	
Edge Width	0...9000		UINT8
Feature Counting Option Parameters			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2		UINT8
Digital Relearn Input Bit	-1	Virtual input bit	INT8
Algorithm Type	3	Pattern Counting	UINT16

Feature Type	1	Light	UINT8
	2	Dark	
Minimum Size	0...99999		FLOAT
Maximum Size	0...99999		FLOAT
Noise Size	0...99999		FLOAT
Relearn flag	0,1	Auto learn the feature size	UINT8
Pattern Counting Option Parameters			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2		UINT8
Digital Relearn Input Bit	-1	Virtual input bit	INT8
Algorithm Type	2	Pattern Finding	UINT16
Correlation Type	0	Not Used	UINT8
Minimum Score	0...100	Percent Matching	UINT8
Minimum Separation	0...1000	Best/First Radio Button	INT8

Table 5-20: Options and Advanced Parameters

\$3 - Set Sensor Pass Conditions			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3		UINT8
Minimum Contrast	0...100	As percent	INT16
Minimum Count	0...10000	Offset from start	UINT16
Maximum Count	0...10000	Offset from start	UINT16

Table 5-21: Pass Conditions

Count in Area

The Area Positioning sensor has 3 choices: Blob, Object Locate, and Pattern Match. In this case the parameters for each algorithm have a unique parameter ID. Use only the parameter ID of the chosen algorithm.

Parameter ID		Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings

\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data
\$	2	Digital Relearn Bit and Algorithm Type
\$	3	Experimental (ignore)
\$	4	Pattern Match matching parameters
\$	5	Blob Generator Parameters
\$	*7	Model: Binary Information about Object Model
\$	8	Object Locate Positioning Parameters
\$	9	Blob Positioning Parameters
\$	*10	Model: Learned Blob Model values for Selection Criteria
\$	*11	Command: Add to Object Locate
\$	*12	Command: Remove from Object Locate Model
\$	14	Not Used (ignore)
\$	*20	Model: Pattern Match Binary Pattern information
\$	*45	Command: Relearn Pattern Match on Next Inspection
\$	*46	Command: Relearn Blob

*These parameters may not be included in the sensor query response depending on which algorithm is currently selected. \$11, \$12, \$45, and \$46 are commands and never appear in the response.

\$2 - Digital Relearn and Algorithm Type			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2		UINT8
Digital Relearn Input	-1, 8, 9, 10, 11, 32 - 63		INT8
Select Task	0	Object Locate	UINT8
	1	Pattern Match	
	2	Blob	

Table 5-22: Digital Relearn and Algorithm Type

\$3 – Experimental algorithm (ignore)			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3		UINT8
	0.9		
	3		
	-15		
	15		
	10		

	100		
	100		

Figure 5-15: Experimental (ignore)

\$4 – Pattern Match Options and Pass Conditions			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	4	Pattern Match Options	UINT8
Minimum Matching %	0...100	percent	UINT16
Match to Use	0	Not Used	UINT8
Min Objects to Locate	0...10000	Max # blobs	UINT16
Min Distance	0...1000	Pixels	UINT16
Min Count	0...10000	Pass Condition	INT32
Max Count	0...10000	Pass Condition	INT32

Figure 5-16: Pattern Match Options and Pass Conditions

\$5 - Blob Generator Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	5	Blob Generation Parameters	UINT8
Minimum Contrast	-1...100	Contrast %	INT16
Pixel Connectivity	0	8-way	FLOAT
	1	4-way	
Enable Boundary Blobs	0	Boundary blobs disabled	FLOAT
	1	Boundary blobs enabled	
Minimum Blob Size	-1..400000	Pixel area	INT16
Maximum Blob Size	-1..400000	Pixel area	INT16
Blob Color	0	Dark	UINT8
	1	Light	
Preprocessing Filter 1-5	-1	None	UINT8
	0	Dilation	
	1	Erosion	
	2	Open	
	3	Close	
Filter Height	0...10	Pixels	INT32
Filter Width	0...10	Pixels	INT32
Max. Number of Blobs	-1...32767	Max found blobs	INT32

Figure 5-17: Blob Generation Parameters

\$9 – Blob Options			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	9		UINT8
Minimum Match Score	0...100	percent	INT16
Not Used	0		UINT8
Blob Features to Use	Sum + 1	Area – bit 1	UINT16
	Sum + 2	Perimeter – bit 2	UINT16
	Sum + 16	Average Intensity % - bit 5	UINT16
	Sum + 32	Eccentricity – bit 6	UINT16
	Sum + 64	Compactness – bit 7	UINT16
	Sum + 256	Radius – bit 9	UINT16
	Sum + 2048	Color – bit 12	UINT16
Angle Format	0	Principal Axis 180°	UINT8
	1	Principal Axis 360°	
	2	Max Point 360°	
	3	None	
Sort Order	10	None	UINT8
	2	Position	
	0	Area	
	1	Perimeter	
	8	Radius	
	5	Eccentricity	
	6	Compactness	
Selection Criteria	4	Intensity	UINT8
		Area min	
		Area max	
		Perimeter min	
		Perimeter max	
		Avg Intensity % min	
		Avg Intensity % max	
		Eccentricity min x .01	
		Eccentricity max x .01	
		Compactness min x .01	
		Compactness max x .01	
		Radius min	
		Radius max	
		Red % min	
		Red % max	
		Green % min	
		Green % max	
		Blue % min	
		Blue % max	
Not used	0	Not used	INT16
Min Count	-1...32,767		INT16
Max Count	-1...32,767		INT16

Figure 5-18: Blob Options

\$8 – Object Locate Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	8		UINT8
Show Live Edge Image	0	Don't Show	INT32
	1	Show	
Accurate Position +/-	0...50...100	Accurate position*100	INT32
Max Object Count	0...10000		INT32
Pick Point Offset X	-1000... 1000	Pixels	INT32
Pick Point Offset Y	-1000... 1000	Pixels	INT32
Pick Point Offset Angle	-180...180	degrees	INT32
Min Objects	-1...10000		INT32
Max Objects	-1...10000		INT32
Locate: Pixel Distance	1...10		INT32
Min Match Score	0...100		INT32
First Match / Best Match	5	Default: Best Match	INT32
Min Distance Between Object	0...1000	Pixels	INT32
Min Angle Between Objects	0...180	Angle	INT32
Accurate Position	0	Disabled	UINT8
	1	Enabled	
Mark Object	0	Disabled	UINT8
	1	Enabled	
Translation Only	0	Disabled	UINT8
	1	Enabled	
Limit Rotation Value	0...180	Degrees	INT32
Limit Scale Value	5	Default	INT32
Limit Rotation	0	Disabled	UINT8
	1	Enabled	
Not Used	-1	Not Used	INT32
Object Sort Order	0	Match Angle	INT8
	1	Pick Point X	
	2	Pick Point Y	
	3	Pick Point Angle	
	4	Scale	

Figure 5-19: Object Locate Parameters

\$10 – Blob/Object Locate Learned Model Parameters			
SP <sensor ID> <parameter ID> <Data>			

Blob Model Learned Parameters			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	10	Model Parameters	UINT8
Area	0...16.7M	Pixels	DOUBLE
Perimeter	0...16.7M	Pixels	DOUBLE
Avg Intensity	0...100	%	UINT8
Eccentricity	0...1	Scaled	DOUBLE
Compactness	0...1	Scaled	DOUBLE
Max Radial Point	0...16.7M	Pixels	DOUBLE
Avg Red	0...100	%	UINT8
Avg Green	0...100	%	UINT8
Avg Blue	0...100	%	UINT8
Object Locate Relearn Command and Parameters			
Not Used	1	Not Used	INT32
Min Pixel Size	0...1000	Pixels	INT32
Min Gradient	0...200	Gray scale	INT32

Figure 5-20: Blob and Object Locate Model Parameters

\$7 -Object Locate Model Information			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	7		UINT8
Object Model Info	...	Multiple Binary(NOT DOCUMENTED)	INT32

Figure 5-21: Object Locate Model Information

\$11 - Add to Object Locate Model Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	11	Relearns pattern on next inspection.	UINT8

Figure 5-22: Add to Object Locate Model Command

\$12 - Remove from Object Locate Model Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	412	Relearns pattern on next inspection.	UINT8

Figure 5-23: Remove from Object Locate Model Command

\$45 - Relearn Pattern Match on Next Inspection			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Relearns pattern on next inspection.	UINT8

Figure 5-24: Relearn Pattern Match Model on Next Inspection Command

\$46 - Relearn Blob Model Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45		UINT8
Blob number to learn	0...1000	index	UINT16

Figure 5-25: Relearn Blob Model Command

Pixel Counting

The Pixel Counting sensor parameter data includes both user parameters and color model data for both grayscale and full color cameras. You'll notice that there are far less Pixel Counting parameters exposed through the Intellect UI when using a grayscale camera then when using a color camera. Parameter ID 1 contains all the parameter data for both system types, thus much of the data related to color cameras will be unused if using a grayscale camera.

Parameter ID 3 is used to store both color model parameters and color model data. The structure of this information is very complicated and thus is left undocumented.

Several commands are available for managing the color models. These commands are sent the same way all other parameters are set, by using their corresponding parameter IDs and any associated parameters. These parameter IDs do not however show up when the sensor is queried using the SQ command since there is no data to be queried.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data
\$	1	Training and Pass/Fail Parameters

\$	2	Digital Relearn Bit and Algorithm Type
\$	*3	Color Model Parameters (partially undocumented)
\$	*3	Model: Color Model Data (partially undocumented)

* \$3 is listed twice for a specific color model (parameters and model data) and then repeated for each new color. Both of these are currently undocumented.

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
-1 No Threshold			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251		UINT8
Threshold ID	-1	No Threshold	INT8

Figure 5-26: Sensor Thresholds

\$1 – Training and Pass/Fail Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2		UINT8
Sigma Factor	0...1000	Sigma x 100	UINT16
Matte Areas Flag	0	Disabled	UINT8
	1	Enabled	
Matte % Darker	0-100	%	UINT8
Matte % Brighter	0-100	%	UINT8
Highlight Areas Flag	0	Disabled	UINT8
	1	Enabled	
Highlight % Magnitude	0-100	%	UINT8
Unused	0	Default	UINT
Digital Relearn Input	8...63	Virtual I/O bit position	UINT8
Disable Inspection Flag	0	Disabled	UINT8
	1	Enabled	
Threshold(pixels)	0-1000	# pixels	UINT16
Digital Relearn Task	0	Learn one and append	UINT8
	1	Remove all, learn one	
	2	Merge and append	
	3	Remove all, learn merged	
	4	Multiple add	
	5	Multiple remove	
*Gray Scale Low Thresh	-1	Default	INT32
*Gray Scale High Thresh	-1	Default	INT32
Minimum % Pixels	-1	% x 100	INT32

Minimum Pixel Count	0...1310720	Pixels	INT32
Maximum Pixel Count	0...1310720	Pixels	INT32
Unused	-1	Default	INT32
Match Color Name Flag	0	Disabled	UINT8
	1	Enabled	
Digital Color Selection Input	-1...63	Virtual I/O bit position	UINT8
Color Name	""	Text String	STRING

Figure 5-27: Training and Pass Parameters

*Gray scale systems only

\$2 - Digital Relearn and Algorithm Type			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Digital Relearn and Algorithm Type	UINT8
Digital Relearn Input	-1...63		INT8

Figure 5-28: Digital Relearn and Algorithm Type

\$3 – Color Model Parameters (Undocumented)			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Color Model Parameters	UINT8

Figure 5-29: Color Model Parameters

\$3 – Color Model Data (Undocumented)			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Color Model Data	UINT8

Figure 5-30: Color Model Data

\$44 – Delete All and Learn Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	44	Delete and Learn	UINT8

Figure 5-31: Delete All and Learn Command

\$45 – Learn and Add Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type

Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Learn and Add	UINT8

Figure 5-32: Learn and Add Command

\$47 – Set Color Name Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Set Color Name	UINT8
Existing Color Index	1...	Index base 1	INT32
Name	“ ”	New color name	STRING

Figure 5-33: Set Color Name Command

\$48 – Delete Color at Index Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Delete Color	UINT8
Existing Color Index	1	Index base 1	UINT16

Figure 5-34: Delete Color at Index Command

\$50 – Multi Color Learn Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	45	Multi-learn	UINT8
Existing Color Index	0...	Index base 1	INT32
Min Threshold	1...1000	Pixels	INT32
Action	0	Multi learn and append to list	INT8
	1	Add to multi-color	
	2	Remove from multi-color	

Figure 5-35: Multi-Color Learn Command

\$54 – Delete All and Copy Color by Index Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	ID of destination sensor	UINT16
Parameter ID	54	Delete All and Copy	UINT8
Sensor ID	1...32768	ID of source sensor	INT16
Color Index	1...	Index base 1 in source sensor	INT32

Figure 5-36: Delete All and Copy Color at Index

Note: This command deletes all the colors from the destination sensor indicated by the first Sensor ID and copies the color at Color Index from the sensor indicated by the second Sensor ID to the first.

Measurement

This section contains descriptions of the parameters for the Measure Along Line, Measure in Area, Measure in Circle, Measure with Points and Lines, and Datum sensors. Both the Measure in Area and Measure in Circle sensors have multiple algorithm options. Different algorithms may use more or less parameters so take note of which parameters in the Intellect UI are used with a specific algorithm. The Measure with Points and Lines sensor also inherently exposes a range of measurement options so take note of the differences in usable parameters. The Datum and Statistics sensors contain no significant parameter data and are not currently documented.

Measure Along Line

Measure Along Line

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Threshold Information
\$	253	Timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Sensor Transform Data
\$	2	Edge Finding Parameters
\$	3	Pass/Fail Parameters

\$251 - Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Parameter Specific	UINT8
Threshold Type	0	Fixed, intensity-based threshold	INT8
Intensity: Fixed Value (0)			
Threshold Type	0	Fixed, intensity-based threshold	INT8
Fixed Threshold Value	0...10000	Intensity value * 100	UINT16
Intensity: Percent of Path Contrast (1)			
Threshold Type	1	Percent of Path Contrast.	INT8
Threshold Value	0...10000	Percent of Contrast * 100	UINT16

Intensity: Auto-Bimodal (3)			
Threshold Type	3	Auto-Bimodal Threshold	INT8
Ignore Saturated Pixels	0	Disabled	UINT8
	1	Enabled	
Gradient: Fixed Value (32)			
Threshold Type	32	Fixed gradient value	INT8
Gradient Value	0... 10000	Gradient value * 100	UINT16

Figure 5-37: Sensor Thresholds

\$2 - Edge Finding Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Parameter Specific	UINT8
Not used	-1	Not used	INT8
Algorithm Type	1	Edge Find (only option)	UINT16
Edge Type	0	Either	UINT8
	1	Dark to Light	
	2	Light to Dark	
Not used	0	Not used	UINT16
Edge Width	0... Max	Pixels	UINT16

Figure 5-38: Edge Finding Parameters

\$3 - Pass/Fail Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Parameter Specific	UINT8
Minimum Contrast	0 ... 10000	Min. Contrast *100	INT32
Minimum Distance	-1 or 0 +/- max	Pixels	FLOAT
Maximum Distance	-1 or 0 +/- max	Pixels	FLOAT

Figure 5-39: Pass/Fail Parameters

Measure in Area

Measure in Area allows six different thickness measurements. Only the first, Measure Thickness, exposes advanced parameters through the Intellect UI. It also exposes one extra edge finding option and three extra pass conditions.

Parameter ID	Description
--------------	-------------

\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Sensor Threshold Information
\$	253	Sensor timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data
\$	1	Measurement Task
\$	2	Algorithm Parameters
\$	6	Pass/Fail Parameters
\$	7	Not Used

\$251 - Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16

Figure 5-40: Sensor Thresholds

\$1 - Sensor's Task			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	1	Parameter Specific	UINT8
Algorithm Type	0	Measure Thickness	UINT16
	1	Minimum Thickness	
	2	Maximum Thickness	
	3	Median Thickness	
	4	Min Max Thickness	
	5	Average Thickness	

Figure 5-41: Sensor Task

\$2 - Algorithm Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Parameter Specific	UINT8
Edge Width	0...Max	Edge Width in pixels. Default is 4	UINT16
Edge Type	0	Either	UINT8
	1	Dark to Light	
	2	Light to Dark	
Edge to Use	0	First	UINT8
	1	Best Edge	
	2	All (Measure Thickness only)	
Outlier Distance	0 +/- Max	Distance * 100, -1 to disable	INT32
Number of Iterations	0 ... Max	Number of iterations	INT32
Hough Transform Flag	0	Disable	UINT8
	1	Enable	
Hough Transform Resolution	0 ... 10	Accuracy measure	INT32
Hough Transform Resolution (repeat)	0 ... 10	Accuracy measure	INT32

Figure 5-42: Algorithm Parameters

\$6 - Pass/Fail Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	6	Parameter Specific	UINT8
Minimum Contrast	0 ... 10000	Min. Contrast *100. Default is 1500	INT32
Minimum Distance	-1 or 0 +/- max	Set to -1 when disabled	FLOAT
Maximum Distance	-1 or 0 +/- max	Set to -1 when disabled	FLOAT

Minimum Angle	0 +/- 90	Ranges from -90 to 90 degrees. Setting this to -100 will disable the option. Default is -100	FLOAT
Maximum Angle	0 +/- 90	Ranges from -90 to 90 degrees. Setting this to -100 will disable the option. Default is -100	FLOAT
Straightness	-1 or 0 +/- max	Set to -1 when disabled	FLOAT

Figure 5-43: Pass/Fail Parameters

Notes: This SoftSensor does not use parameter ID 7 even though it is included in the response to the SoftSensor query command.

Measure in Circle

Measure in Circle allows seven different circle measurements. Some measurements expose different edge finding and advanced parameters as well as different pass criteria.

Parameter ID	Description
\$ 240	General: Sensor Name
\$ 241	Inspection Results: User Pass/Fail Bits
\$ 242	General: Sensor Description
\$ 243	General: Perform inspection Flag
\$ 245	General: Scaling: Scale consumer information
\$ 246	General: Scaling: Scale producer information
\$ 251	Sensor Threshold Information
\$ 253	Sensor timeout and Local Marking Mode Settings
\$ 255	General: Image Reference Parameter
\$ 0	Shape, Position Reference and Transform Data
\$ 1	Measurement Task
\$ 2	Algorithm Parameters
\$ 6	Not Used
\$ 7	Pass/Fail Parameters

\$251 - Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251		UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8

0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
32 Fixed Gradient Threshold			
Threshold ID	32	Gradient Threshold	INT8
Gradient Threshold Value	0...10000	Gradient value*100	UINT16

Figure 5-44: Sensor Threshold

\$1 – Sensor's Task			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	1	Parameter Specific	UINT8
Algorithm Type	0	Distance between Centers	UINT16
	1	Minimum Thickness	
	2	Maximum Thickness	
	3	Median Thickness	
	4	Min Max Thickness	
	5	Average Thickness	
	6	Distance between Centers	

Figure 5-45: Sensor Task

\$2 - Algorithm Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Parameter Specific	UINT8
Edge Width	0...Max	Edge Width in pixels. Default is 4	UINT16
Edge Type	0	Either Edge	UINT8
	1	Dark to Light	
	2	Light to Dark	
Edge to Use	0	First Edge	UINT8
	1	Best Edge	

Outlier Distance	0 +/- Max	Distance * 100. Negative values mean that the option is disabled. Default is -500 (disabled)	INT32
Number of Iterations	0 ... Max	Number of iterations	INT32
Hough Transform Flag	0	Not Used.	UINT8
Hough Transform Resolution	0	Not Used	INT32
Hough Transform Resolution (Repeated)	0	Not Used	INT32

Figure 5-46: Algorithm Parameters

\$7 - Pass/Fail Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	7	Parameter Specific	UINT8
Minimum Contrast	0 ... 10000	Min. Contrast *100. Default is 1500	INT32
Minimum Radius/Thickness	-1 or 0 +/- max	Set to -1 when disabled. Used for minimum radius or thickness depending on the task	FLOAT
Maximum Radius/Thickness	-1 or 0 +/- max	Set to -1 when disabled. Used for minimum radius or thickness depending on the task	FLOAT
Roundness	-1 or 0 + max	Set to -1 when disabled. Only applicable to Distance between Centers and Measure Radius tasks but always needs to be included regardless of the task	FLOAT

Figure 5-47: Pass/Fail Parameters

Notes: This SoftSensor does not use parameter ID 6 even though it is included in the response to the SoftSensor query command.

Measure with Points and Lines

Measure with Points and Lines uses only sensor IDs and an angle setting in the case of angle measurements. The sensor IDs correspond to the two sensors that are sourcing the measurement data. If a measurement requires two points, it must reference two sensors that generate points. Make sure the sensors you use generate the correct data.

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag

\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Sensor Threshold Information
\$	253	Sensor timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data
\$	1	Measurement Task and Sensor References
\$	2	Pass/Fail Parameters

\$1 - Task and Sensor References			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	1	Parameter Specific	UINT8
Operation	0	No Operation	UINT16
	1	Distance	
	2	Angle	
	3	Intersection	
	4	Midpoint	
	5	Midline	
	6	Perpendicular Line	
	7	Parallel Line	
	8	Connecting Line	
First Reference SoftSensor	-1 or 0 ... 32768	Sensor ID	INT16
First Reference Output	1720	Point	INT16
	1750	Line	
Second Reference SoftSensor	-1 or 0 ... 32768	Sensor ID	INT16
Second Reference Output	1720	Point	INT16
	1750	Line	
Angle Format ¹	1	0 +180 degrees	UINT16
	2	0 to +360 degrees	
	3	-90 to +90 degrees	
	4	-180 to +180 degrees	

Figure 5-48: Task and Sensor References

¹: This parameter is only used and included when the operation is set to "Angle" (2). For any other operation this parameter should not be included at all. The options here are the following:

\$2 - Pass/Fail Parameters	
SP <sensor ID> <parameter ID> <Data>	

Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Parameter Specific	UINT8
Minimum Distance/Angle	-1000 or 0 +/- max	Set to -1000 when disabled. For the Distance operation this is in terms of pixels. For the Angle operation, this is in terms of degrees	FLOAT
Maximum Distance/Angle	-1000 or 0 +/- max	Set to -1000 when disabled. For the Distance operation this is in terms of pixels. For the Angle operation, this is in terms of degrees	FLOAT

Figure 5-49: Pass/Fail Parameters

Datum

SoftSensor Query for Datum SoftSensor

Parameter ID		Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Sensor Threshold Information
\$	253	Sensor timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data

Notes: This SoftSensor does not have any specific parameters other than the common SoftSensor parameters. Its main purpose is to generate a position that other SoftSensors can refer to as an offset for measurements. Notice that common parameter 251 (Threshold) is not used so it is set to -1

Readers

2D Reader

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Sensor Threshold Information
\$	253	Sensor timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Sensor Shape, Position Reference and Sensor Transform Info
\$	2	Digital Relearn and Algorithm Type
\$	3	Data Matrix Parameters
\$	4	SnowFlake Parameters
\$	5	Vericode Parameters
\$	6	Pass/Fail Parameters

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Parameter specific	UINT8
Threshold Type	-3	Threshold dependent	STRING
Data Matrix			
3 Auto Bimodal Threshold (default)			
Threshold ID	3	Auto Bimodal	STRING
Ignore Saturated Pixels	1	Ignore	UINT8
SnowFlake			
0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8

% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
Vericode			
3 Auto Bimodal Threshold (default)			
Threshold ID	3	Auto Bimodal	STRING
Ignore Saturated Pixels	1	Ignore	UINT8

Figure 5-50: Sensor Thresholds

\$2 - Digital Relearn and Algorithm Type			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Digital Relearn and Algorithm Type	UINT8
Digital Relearn Input	-1...63	Virtual I/O Bit Position	INT8
Select Task	0	Data Matrix	UINT8
	1	Snow Flake	
	2	Vericode	

Figure 5-51: Digital Relearn and Algorithm Type

\$3 – Data Matrix Parameters			
SP <sensor ID> <parameter ID> <Data>			
Data Matrix			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Data Matrix Parameters	UINT8
Symbol Type	0	ECC200	UINT8
	1	ECC000	
	4	ECC050	
	7	ECC080	
	9	ECC100	
	13	ECC140	
	14	ECC000-140	
	15	ECC000-200	
Symbol Size	0	10 x 10	UINT8
	1	12 x 12	
	2	14 x 14	
	3	16 x 16	
	4	18 x 18	
	5	20 x 20	
	6	22 x 22	
	7	24 x 24	
	8	26 x 26	

	9	32 x 32	
	10	36 x 36	
	11	40 x 40	
	12	44 x 44	
	13	48 x 48	
	14	52 x 52	
	15	64 x 64	
	16	72 x 72	
	17	80 x 80	
	18	88 x 88	
	19	96 x 96	
	20	104 x 104	
	21	120 x 120	
	22	132 x 132	
	23	144 x 144	
	30	Auto Detect	
Module Type	0	Auto Detect Modules	UINT8
	1	Dark Modules	
	2	Light Modules	
	3	Auto Detect Outline	
	4	Dark Outline	
	5	Light Outline	
Max Skew	0...45	Degrees	UINT8
Min Symbol Size	25...9999	Pixels	UINT16
Max Symbol size	25...9999	Pixels	UINT16
Not used		Not used	
Mirror Image	0	Auto Detect	UINT8
	1	Normal Image	
	2	Mirror Image	
Min Border Integrity	1...75	Percent	UINT8
Not used		Not used	
Max Axial Non-uniformity	0...200	Value x 100	UINT8
Image Marking Flags	1		
	2		
	4	Mark Decode Errors	
	8		
	16		
	32	Mark Cells	
	64	Mark Cell Values	
	128		
Not used		Not used	
Mark Decode Trial	-1...1000		
Threshold Type	1	Histogram Threshold	UINT8
	2	Local Comparison	
	3	Combination	
	4	All Three	
Sample Size	0	Auto	UINT8

	1	1 x 1	
	3	3 x 3	
	5	5 x 5	
	7	7 x 7	
Probe Width	1...20	Pixels	UINT8
Clean Area	1...200	Pixels	UINT8
Min Gradient	1...10000	% of Intensity Difference	UINT16
Not used		Not used	
Not used		Not used	
Not used		Not used	
Not used		Not used	
Data Matrix Options and Search Parameters Trainable Flags	Sum + 0	Square Matrix	UINT8
	Sum + 1	Symbol Type Trainable	
	Sum + 2	Symbol Size Trainable	
	Sum + 4	Module Type Trainable	
	Sum + 8	Search Parameters Trainable	
	Sum + 16	Threshold Type Trainable	
	Sum + 32	Sample Size Trainable	
	Sum + 64	Mirror Image Trainable	
	Sum + 128	Rectangular Matrix	
Scale Variation	5...100	% Size Change	
Fast Mode	0	Disabled	
	1	Enabled	
Not used		Not used	
Not used		Not used	
Not used		Not used	
Not used		Not used	
Limit Max Rotation	0, 5, 15	Degrees	UINT8
Minimum Grade	-1	Disabled	INT16
	0	F	
	1	D	
	2	C	
	3	B	
	4	A	

Figure 5-52: Data Matrix Parameters

\$4 – Snowflake Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	4		UINT8
Expected Size	0...100	Dot Size Parameter	INT16
Size Tolerance	0...32	Dot Size Parameter	UINT8
Squareability	0...200	Shape Restriction, (Value + 1) x 100	UINT8

Symmetry	0...200	Shape Restriction, (Value + 1 x 100	UINT8
Size Check Flag	0	Disabled	UINT8
	1	Enabled	
Quiet Zone Flag	0	Disabled	UINT8
	1	Enabled	
Neighbor Check Flag	0	Disabled	UINT8
	1	Enabled	
Perimeter Check Flag	0	Disabled	UINT8
	1	Enabled	
Density	0	Auto Detect	UINT8
	1	7 x 16	
	2	7 x 24	
	3	11 x 24	
	4	12 x 24	
	5	16 x 24	
	6	8 x 8	
	7	9 x 9	
	8	10 x 10	
	9	11 x 11	
	10	12 x 12	
	11	13 x 13	
	12	14 x 14	
	13	15 x 15	
	14	16 x 16	
	15	17 x 17	
	16	18 x 18	
	17	19 x 19	
	18	20 x 20	
	19	21 x 21	
	20	22 x 22	
	21	23 x 23	
	22	24 x 25	
Format	255	Not used	UINT8
ECC	0	None	UINT8
	1	Check Sum	
	2	10% ECC	
	3	20% ECC	
	4	40% ECC	
	5	Auto	
Color	0	Dark on Light	UINT8
	1	Light on Dark	
Dynamic Grid Alignment Flag	0	Disabled	UINT8
	1	Enabled	
Cell Sampling	0...5	Index	UINT8
Image Marking Flags	Sum + 0	No Markings	UINT8
	Sum + 1	Intensity	
	Sum + 2	Searching Grid	
	Sum + 4	Corner Check	
	Sum + 8	Grid Alignment	
	Sum + 16	Cell Sample Points	

	Sum + 32	Corner Dots	
	Sum + 64	Left/Right Blobs	
	Sum + 128	Top/Bottom Blobs	
Minimum Contrast	-1...10000	Contrast % x 100 (Vericode only)	INT16

Figure 5-53: SnowFlake Parameters

\$5 –Vericode Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	5	Vericode Parameters	UINT8
Matrix Size Auto Detect	0	Auto Detect	UINT8
	1	Explicit Value	
Matrix Size	10-48	Even values for n x n matrix size	UINT8
	13	Use for 56 x 56 matrix size	
Index Contrast	0-20	0-100% Contrast x .2	UINT8
Symbol Type	0	White on Black	UINT8
	1	Black on White	
	2	Outline	
EDAC Level	0	Auto Detect	UINT8
	1	12.5%	
	2	25.0%	
Compression	0	None	INT16
	-1	Numeric	
	-2	Alphanumeric	
Bits Per Cell	2	Pixels	UINT8
	4	Pixels	
	8	Pixels	
	16	Pixels	
Sample Width	1...8	Pixels	UINT8
Not Used	0	Not Used	UINT8
Prefiltering Flag	0	Disabled	UINT8
	1	Enabled	
Not Used	0	Not Used	UINT8
Not Used	0	Not Used	UINT8
Not Used	1	Not Used	UINT8
Extract Points Flag	0	Disabled	UINT8
	1	Enabled	
Object Map Flag	0	Disabled	UINT8
	1	Enabled	
Border Flag	0	Disabled	UINT8
	1	Enabled	
Symbol Flag	0	Disabled	UINT8
	1	Enabled	

Figure 5-54: Vericode Parameters

\$6 – Pass/Fail Output String Parameters
--

SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	6	Parameter specific	UINT8
Match Option	-1	Match String Exactly	INT16
	0...200	Do Not Check String – Value is ignored when the Output String is empty	
	0...200	Match String Subset – Value is the substring beginning index	
Output String	""	Match String	STRING

Figure 5-55: Pass/Fail Output String Parameters

OCR

Parameter ID	Description
\$ 240	General: Sensor Name
\$ 241	Inspection Results: User Pass/Fail Bits
\$ 242	General: Sensor Description
\$ 243	General: Perform inspection Flag
\$ 245	General: Scaling: Scale consumer information
\$ 246	General: Scaling: Scale producer information
\$ 251	Sensor Threshold Information
\$ 253	Sensor timeout and Local Marking Mode Settings
\$ 255	General: Image Reference Parameter
\$ 0	Sensor Shape, Position Reference and Sensor Transform Info
\$ 3	Font Parameters
\$ 5	Marking Parameters
\$ 6	OCR Parameters
\$ 7	Fielding String Parameters
\$ 9	Pass Parameters

\$251 - Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Threshold	UINT8
Threshold ID	-	Threshold dependent	INT8
-1 No Threshold			
Threshold ID	-1	No Threshold	INT8

0 Fixed Intensity Threshold			
Threshold ID	0	Fixed Threshold	INT8
Fixed Threshold Value	0...10000	Intensity value*100	UINT16
1 Percent Contrast Threshold			
Threshold ID	1	% Path Contrast	INT8
% Contrast Threshold	0...10000	Percent Contrast*100	UINT16
3 Auto Bimodal Threshold			
Threshold ID	3	AutoBimodal	INT8
Ignore Saturated Pixels	0	Don't ignore	UINT8
	1	Ignore	
4 Adaptive Intensity			
Threshold ID	4	Adaptive Threshold	INT8
Threshold contrast	0...10000	Threshold value*100	UINT16
Window Size	0...100	Window in pixels	INT16
Min Window Contrast	1...10000	Contrast value*100	UINT16
64 OCR Computed			
Threshold ID	64	OCR Computed	INT8
Threshold Value	5000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Checkbox	UINT8
Block Size	4	Contrast value*100	UINT8
65 OCR Linear			
Threshold ID	65	OCR Linear	INT8
Threshold Value	0...10000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Don't reduce	UINT8
	1	Reduce	
Block Size	4	Contrast value*100	UINT8
66 OCR Nonlinear			
Threshold ID	66	OCR Nonlinear	INT8
Threshold Value	0...10000	Contrast Value *100	UINT16
Use Floating Point Calculations	0	Don't use FP	UINT8
	1	Use FP	
Reduced Characters	0	Don't reduce	UINT8
	1	Reduce	
Block Size	0...4...50	Number of areas	UINT8

Figure 5-56: Sensor Thresholds

\$3 Font Parameters

SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Threshold	UINT8
Font Name	""	Name of saved font	STRING

Figure 5-57: Font Parameters

\$5 Image Marking Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	5	Threshold	UINT8
Image Marking Flags	4	Mark in original shape (trainable font)	
	5	none	
	6	Mark in original shape (fixed font)	
	7	Show image thresholding	

Figure 5-58: Image Marking Parameters

\$6 Fixed Font and Trainable OCR Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Trainable OCR			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	6	OCR Parameters	UINT8
Algorithm	0	Trainable OCR	
Color Flag	0	Black on White	UINT8
	1	White on Black	
Min Separation: Horizontal	0...255	Pixels	
Min Separation: Vertical	0...255	Pixels	
Not used		Not used	
Aspect Ratio Dependence Flag	0	Disabled	UINT8
	1	Enabled	
Not used		Not used	
Matching Threshold	0...100	%	
Min Character Area	0...1000	Pixels	
Not used		Not used	
Character Width: Max Flag	0	Disabled	UINT8
	1	Enabled	
Use Advanced Max Character Width Flag	0	Disabled	UINT8
	1	Enabled	
Character Width: Max	0...255	Value	
Character Width: Min	-value	Disabled	

	0...255	Value	
Character Height: Max Flag	0	Disabled	UINT8
	1	Enabled	
Character Height: Max	0...255	Value	
Character Height: Min	-value	Disabled	
	0...255	Enabled	
Discard Oversized Characters Flag	0	Disabled	UINT8
	1	Enabled	
Discard Undersized Characters	0	Disabled	UINT8
	1	Enabled	
Force Space Between Characters Flag	0	Disabled	UINT8
	1	Enabled	
Force Space Between Characters Value	0...1000	Pixels	
Not used		Not used	
Aspect Ratio Dependence Value	0...1000	%	
Min Edge Width to Start Character Flag	0	Disabled	UINT8
	1	Enabled	
Min Edge Width to Start Character Value	0...100	Pixels	
Enable Orphan Removal	0	Disabled	UINT8
	1	Enabled	
Enable Character Fill	0	Disabled	UINT8
	1	Enabled	
Searching	0	None	UINT8
	1	First Character Only	
	2	All Characters	
Max Number of Character to Extract	-1	Disabled	
	0...999	Value	
Characters to Train	0	Failed Characters Only	UINT8
	1	All Characters	
Digital Retrain Input	0...64	Digital bit	
Clear Character List During Retrain Flag	0	Disabled	UINT8
	1	Enabled	
Training String	""		STRING
Fixed Font OCR			
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	6		UINT8
Algorithm	1	Fixed Font OCR	
Calibration Horizontal Scale	0...100		

Calibration Vertical Scale	0...100		
Calibration Character	“”	Single character	STRING
Image Condition	0	Normal	UINT8
	2	Two Tone Background	
	4	Halo Shadow	
Checksum	0	None	UINT8
	1	SEMI M12	
	2	SEMI M13	
	3	SEMI T1	
	4	SEMI 59	
	5	AIM630	
Contrast Type	0	Unknown	UINT8
	1	Dark on Light	
	2	Light on Dark	
Minimum Score	0...100		
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Fixed value	Fixed value	Fixed value	
Trainable Flags	1	Image Condition	UINT8
	2	Contrast Type	
Fielding String	“”		STRING

Figure 5-59: Fixed Font and Trainable OCR Parameters

\$7 Fielding String Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	7		UINT8
Enable Auto Fielding for OCV	0	Disabled	UINT8
	1	Enabled	
Fielding String for Each Position*	Characters	String of characters	STRING

Figure 5-60: Fielding String Parameters

*Up to 50 fielding strings are allowed. Each string is inserted one after the other surrounded with quotation marks and with no spaces.

Example

```
#SP 1 7 0 "abcdefghijklmnopqrstuvwxyz"13579""ABD""AaBbq123?"....."
```

This allows all lower case letter in position 1, the numbers 1, 3, 5, 7 and 9 in position 2, the upper case letters A, B and D in position 3, anything in position 4 (indicated by empty double quotes), and alphanumeric characters A, a, B, b, q, 1, 2, 3 and ? in position 5. The remaining 45 positions allow anything as indicated by the double quotes (ignore the ...).

\$9 Pass Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	9	Pass Parameters	UINT8
Min Contrast	-1	Disabled	INT16
	0...10,000	Value x 100	
Min Score	-1	Disabled	INT16
	1000	Value	
Number of Characters	-1	Disabled	INT16
	0...100	Value	
All Characters Must Read Flag	0	Disabled	UINT8
	1	Enabled	
Match String Flag	0	Disabled	UINT8
	1	Enabled	
Digital Relearn Input	0...64	Relearn bit	INT16
Match String	""		STRING

Figure 5-61: Pass Parameters

Flaw Detection

Defect Detection

	Parameter ID	Description
\$	240	General: Sensor Name
\$	241	Inspection Results: User Pass/Fail Bits
\$	242	General: Sensor Description
\$	243	General: Perform inspection Flag
\$	245	General: Scaling: Scale consumer information
\$	246	General: Scaling: Scale producer information
\$	251	Sensor Threshold Information
\$	253	Sensor timeout and Local Marking Mode Settings
\$	255	General: Image Reference Parameter
\$	0	Shape, Position Reference and Transform Data
\$	1	Training Parameters
\$	2	Settings Parameters
\$	4	Digital Relearn
\$	5	Pass/Fail Parameters
\$	30	Position Data

\$251 - Set Sensor Threshold			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	251	Parameter specific	UINT8
Threshold Type	-1	No threshold	STRING

Figure 5-62: Sensor Threshold

\$1 - Training Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	1	Parameter specific	UINT8
Mask Expand Factor	0...10		INT32
Learn images	0...30	# images	INT32
Create Mask Flag	0	Disable	UINT8
	1	Enable	
Image Marking Mode	0	Disable	UINT8

	1	Enable	
	0	Live	
	1	Model	
	2	Mask	
	3	Residual	
Image to Display			INT32

Figure 5-63: Training Parameters

\$2 - Settings Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	2	Parameter specific	UINT8
Not used	...	Not used	INT32
8 way connectivity Flag	0	Disabled	UINT8
	1	Enabled	
Boundary Blobs Flag	0	Disabled	UINT8
	1	Enabled	
Min blob size	0...400000	Pixels	INT32
-1	-1	Default	INT32
1	1	Default	UINT8
Filter Type 1...5	0	None	INT8
	1	Dilation	
	2	Erosion	
	3	Open	
	4	Close	
Filter Width 1...5	10	locked	INT32
Filter Height 1...5	10	locked	INT32
Max num blobs	5000	Default	INT32
Max color/intensity change	0...100	%	INT32
Enable normalization	0	Enable	UINT8
	1	Disable	

Figure 5-64: Setting Parameters

*Only 2 preprocessing filters are exposed in the Intellect UI

\$4 - Digital Relearn Parameters			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	4	Filter to Display	UINT8
Digital Relearn Input	8...63	Virtual Input Bit	UINT8
Digital Add to Mask Input	8...63	Virtual Input Bit	UINT8

Figure 5-65: Digital Relearn Parameters

\$5 - Pass/Fail Criteria			
SP <sensor ID> <parameter ID> <Data>			

Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	5	Pass/Fail Criteria	UINT8
Max Defect Count Flag	0	Disable	UINT8
	1	Enable	
Total Area Flag	0	Disable	UINT8
	1	Enable	
Max Defect Count Value	0...1000		INT32
Total Area Value	100000	Pixels	INT32

Figure 5-66: Pass/Fail Criteria

\$30 – Position Data			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	3	Filter to Display	UINT8
Original Center Point X	0...max	ROI center	UINT16
Original Center Point Y	0...max	ROI center	UINT16
Original Cos θ	-1...1	Rotation	FLOAT
Original Sin θ	-1...1	Rotation	FLOAT
Current Center Point X	0...max	ROI center	UINT16
Current Center Point Y	0...max	ROI center	UINT16
Current Cos θ	-1...1	Rotation	FLOAT
Current Sin θ	-1...1	Rotation	FLOAT
Original Center Point X	0...max	ROI center	UINT16
Original Center Point Y	0...max	ROI center	UINT16
Original Cos θ	-1...1	Rotation	FLOAT
Original Sin θ	-1...1	Rotation	FLOAT

Figure 5-67: Position Data

Note: The last 4 parameters are repeats of the first 4

\$11 – Add to Mask Command			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	11	Add to Mask	UINT8

Figure 5-68: Add to Mask Command

\$12 - Expand Mask Command			
SP <sensor ID> <parameter ID> <Data>			

Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	12	Expand Mask	UINT8

Figure 5-69: Expand Mask Command

\$22 - Shrink Mask			
SP <sensor ID> <parameter ID> <Data>			
Parameters	Data	Description	Type
Sensor ID	1...32768	Sensor specific	UINT16
Parameter ID	22	Shrink Mask	UINT8

Figure 5-70: Shrink Mask Command

6. Terminal Command Reference

DVT systems use a defined set of commands to communicate between a user interface and the camera. These commands are the basis for programming the way the camera uses its installed inspection and communications algorithms. These are low level commands. That means they are easier to use than writing machine code but much more cumbersome to use than programming in a user interface like Intellect. 99% of DVT users don't need to use these commands 99% of the time. The Intellect user interface should **ALLWAYS** be the chosen interface for designing your application.

So when do you use terminal commands? There are three scenarios that may require use of terminal commands. First, if you are designing a custom user interface you may need to use terminal commands. DVT provides tools like ActiveX controls, .NET components and SmartLink to make custom UI design easy, but you may need added functionality. Secondly, if you need to change system, product, or sensor parameters on the fly that can't be changed programmatically, you can send commands over a socket from script. This is described in the Intellect Script Reference. Thirdly, sometimes it is helpful to use terminal commands for debugging or testing.

The Intellect user interface is designed to make designing an inspection easy and flexible, but not every need will be covered. So if you really need functionality that you can only get by using terminal commands, this may be for you.

Access Control Commands

AI Log in a user with a plain text password

AI< User Name> <Password>	
User Name	String
Tab char	String
Password	String

Parameters

User name

The user name that you wish to log in with. This user name must already exist.

Password

The password of the user identified by the user name.

Response

(UINT8)

(UINT32)Access rights

The access rights for this user if the login succeeds. Each bit in this 32bit number represents the enabled/disabled state of a particular access right.

(STRING)User's full name

The user's full name is returned if the login succeeds.

Remarks

Notice that there is no space between the AI command and the user name. If logging is enabled but is not working (perhaps the logging service is down), only the administrator account can log in. No response is returned for failed logins.

Example

```
? #AIAdministrator admin
% 6 4294967295 Administrator
% 0
```

See Also

[AO - Log out user](#)

[AU - Get current user](#)

AO Log out the current user.

AO

No params

No params

Parameters

No params

No params

Response

(UINT32)Access rights

The access rights for the Guest account. Each bit in this 32bit number represents the enabled/disabled state of a particular access right.

(STRING)User's full name

The Guest account name "Guest"

Remarks

The current user is logged out. The current user is automatically set back to the default Guest account, so the response corresponds to the access rights on that account.

Example

? #AO

\$ 4294967295 Guest

% 0

See Also

[AI - Log in user](#)

[AU - Get current user](#)

AU Get the current user

AU

No params

No params

Parameters

No params

No params

Response

(String) User name

The login name of the current user.

(STRING)User's full name

The user's full name is returned.

(UINT32)Access rights

The access rights for this user. Each bit in this 32bit number represents the enabled/disabled state of a particular access right.

Remarks

The current user is logged out. The current user is automatically set back to the default Guest account, so the response corresponds to the access rights on that account.

Example

```
? #AO
$ 4294967295 Guest
% 0
```

See Also

[AI - Log in user](#)

[AO - Log out user](#)

Digital I/O Commands

Dq Query general digital I/O info

DQ <no parameters>

Parameters

No parameters

Response

- 1(type)Number of physical inputs
Correct – always 8?
- 2(type)Number of physical outputs
Correct – always 8?
- 3(type)I/O Isolation
Description – always 8?
- 4(type)Number of virtual inputs
Correct – always 64
- 5(type)Number of virtual outputs
Correct – always 64
- 6(type)Physical input polarity
Correct – state of physical input inversion bits
- 7(type)Physical I/O state
Correct – state including inversion and force
- 8(type)Virtual input state high
Correct – state of virtual inputs after inversion
- 9(type)Physical output polarity
Correct – state of physical output inversion bits
- 10(type)Physical I/O raw state
Correct – state ignoring inversion and force
- 11(type)Virtual output state high
Correct – state of virtual outputs after inversion
- 12(type)Physical I/O state (repeated)
Description – state including inversion and force

Remarks

The first five response values will not change for any given camera. The most useful response values will likely be the Virtual I/O states which reflect the values inside the camera after any polarity inversion. The

Example

```
? #Dq
$ 8 8 12 64 64 1 22 65539 0 23 200 22
% 0
?
```

This is the resulting output when the I/O pins are set to their default mappings, the Trigger, Product Select, Product ID Bit 0, and Pass pins are high (pins 1,2,3 and 5), and pin 1 (Trigger) is inverted. In this case inspections are running and the trigger mode is set to external.

See Also

[DS - Set physical I/O bits](#)

DS Set physical digital I/O bits

DS <Active bits> <Inactive bits>	
Active bits	UINT32
Inactive bits	UINT32

Parameters

Active bits

Decimal number representing the pins to be set high

Inactive bits

Decimal number representing the pins to be set low

Response

No additional response values

Remarks

This command only allows you to set the *physical* I/O states high or low, not the virtual I/O states. Physical I/O is only the I/O mapped to a physical point on the break out board. However, if the I/O pin is mapped as an input, the corresponding virtual input will also be set. If the pin is mapped as an output the pin will be set but the mapped virtual I/O signal will not be set.

The I/O pins are bits in a binary number where each bit can be either on or off. The CON-IBOB has 8 physical I/O pins so together they form an 8-bit binary number. Address each pin as follows:

Pin #	Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8
Value	1	2	4	8	16	32	64	128
Example	1	1	0	1	0	0	0	1

If addressing multiple pins at once, simply add the values for each pin and use that as the active/inactive bits parameter. In the example table above the I/O bit pattern is 11010001, so the equivalent number is $1+2+8+128 = 139$. If you used this as the *Active bits* parameter, pins 1, 2, 4 and 8 would be set high. If you used this as the *Inactive bits* parameter, pins 1, 2, 4 and 8 would be set low.

Notice that any pins not explicitly set high or low are left in their natural state. Depending on the mapping of any pin, it may change state when inspection results are set. For instance if you set the pins assigned to Trigger and Fail high, an inspection will be triggered and the Fail pin may immediately be set low as a result of a passing inspection.

Example

Set pins 1, 2 and 4 high ($1 + 2 + 8 = 11$) and set pin 3 low.

```
? #DS 11 4
% 0
?
```

See Also

[Dq - Query digital I/O](#)

Image Management Commands

Ii Query inspection statistics

li	
No Params	NoParams

Parameters

No Parameters

Response

(UINT32) Total inspections

Total number of inspections since the last power cycle of the camera.

(UINT32) Missed inspections

Number of inspections missed due to inability to acquire an image (no buffers free for acquisition).

(UINT32) Acquisition conflicts

Number of conflicts due to triggers received during acquisition.

(UINT32) Timeout failures

Number of inspections that fail due to a timeout.

(UINT32)Min time in capture queue

Minimum time any image has spent in the acquisition buffer before being inspected.

(UINT32)Min inspection time

Minimum time any image has spent being inspected.

(UINT32)Mean inspection time

Average inspection time of all images.

(UINT32)Max inspection time

Maximum time any image has spent being inspected.

(UINT32)Min time between acquisitions

Minimum time between consecutive image acquisitions

(UINT32)Mean time between acquisitions

Average time spent between consecutive image acquisitions

Remarks

Use this command to gather statistics about all inspections since the last power cycle of the camera. All statistics reflect data gathered since the last power cycle of the camera or the last reset of the statistics (liz command). All times are reported in milliseconds. Statistics are only gathered while inspections are running on the camera. *Minimum time* statistics have a default value of $2^{32}-1$.

Example

```
? #li
? 99 0 0 0 448 0 17 18 250 250
% 0
?
```

See Also

[li - Query specific inspection statistics](#)
[liz - Reset inspection statistics](#)
[liz - Reset specific inspection statistics](#)

Ii Query specific inspection statistics

SP<StatID>	
StatID	char

Parameters

StatID

Character statistic identifier to indicate which statistic should be returned.

- a: Minimum time between acquisitions
- b: Mean time between acquisitions
- c: Acquisition conflicts
- f: Timeout failures
- i: Minimum inspection time
- m: Missed inspections
- n: Mean inspection time
- q: Minimum time in capture queue
- t: Total inspections
- x: Max inspection time

Response

(UINT32) Statistic

Value corresponding to the requested statistic.

Remarks

Use this command to report only specific inspection statistics. All statistics reflect data gathered since the last power cycle of the camera or the last reset of the statistics (liz command). All times are reported in milliseconds. Statistics are only gathered while inspections are running on the camera. *Minimum time* statistics have a default value of $2^{32}-1$.

There is no space between the command and the parameter. Leaving a space will cause an error.

Example

Get inspection count: StatID = t

```
? #lit
$ 105
% 0
?
```

Get max inspection time: StatID = x

```
? #lix
$ 18
% 0
?
```

See Also

[li - Query inspection statistics](#)

[liz - Reset inspection statistics](#)

[liz - Reset specific inspection statistics](#)

Iiz Reset all inspection statistics

Iiz

No params No params

Parameters

No parameters

Response

No Response

Remarks

This command resets all inspection statistics to their defaults. All values default to zero, except the three *Minimum time* statistics which default to $2^{32}-1$.

Example

Before and after clearing inspection statistics.

```
? #li
? 99 0 0 0 448 0 17 18 250 250
% 0
? #liz
% 0
? #li
? 0 0 0 0 2147483647 2147483647 0 0 2147483647 0
% 0
?
```

See Also

[li - Query inspection statistics](#)

[li - Query specific inspection statistics](#)

[liz - Reset specific inspection statistics](#)

Iiz Reset specific inspection statistics

Iiz <StatID>

StatID	char
--------	------

Parameters

StatID

Character statistic identifier to indicate which statistic should be returned.

- a: Minimum time between acquisitions
- b: Mean time between acquisitions
- c: Acquisition conflicts
- f: Timeout failures
- i: Minimum inspection time
- m: Missed inspections
- n: Mean inspection time
- q: Minimum time in capture queue
- t: Total inspections
- x: Max inspection time

Response

No Response

Remarks

This command resets specific inspection statistics to their defaults. All values default to zero, except the three *Minimum time* statistics which default to $2^{32}-1$.

Example

Before and after resetting only the inspection count to zero.

```
? #li
? 99 0 0 0 448 0 17 18 250 250
% 0
? #lizt
% 0
? #li
? 0 0 0 0 448 0 17 18 250 250
% 0
?
```

See Also

[li - Query inspection statistics](#)

[li - Query specific inspection statistics](#)

[liz - Reset inspection statistics](#)

IP Set system image buffers

IP <Buffers>	
Buffers	UINT8

Parameters

Buffers

Number of buffers

Response

No Response

Remarks

This command sets the number of system image buffers to be reserved for acquisition. This command is only necessary in Framework because this could not be controlled from the UI. In Intellect this setting is accessible in the menu System Administration >> Imaging.

Increasing system image buffers allows the camera to buffer the acquisition so that it can acquire a burst of images while inspecting previous images. The default value is 2. This allows one image to be acquired while a previous image is being inspected. If another image trigger is received and the first inspection is not complete a resource conflict will be triggered and the missed inspection count will increment.

In versions of Framework 2.7.3 beta4 and later this buffer value is saved to flash automatically and the camera should be power cycled so that the new memory allocations can be read and set. In previous versions of Framework, the value is set automatically but it is not saved in flash, thus power cycling the camera causes the value to be lost and the IP command must be resent. In all versions of Intellect the value is only stored and set once the camera is power cycled. In no version of software is the buffer setting saved to the system or product files. As stated above, this command is not necessary in Intellect.

Example

Set the number of system image buffers to 4.

```
? #IP4
% 0
?
```

See Also

IQ Query image information

IQ	
No params	No params

Parameters

No parameters

Response

(UINT8) *Delete strategy*

- 0: Delete newest
- 1: Delete oldest
- 2: Delete newest by priority
- 3: Delete oldest by priority

(INT32) *Image ID_{1...n}*

The image ID associated with each image stored in the retained images queue

(INT8) *Result code_{1...n}*

The result of the inspection on the image

(INT16) *Product ID_{1...n}*

ID of the product the image was inspected with

(INT16) *Image handle_{1...n}*

Internal handle to the image

(INT16) *First failing sensor ID_{1...n}*

Sensor ID of the first sensor that failed in the inspection

(INT8) *First failing sensor result_{1...n}*

Result of the first sensor that failed in the inspection

Remarks

This command returns image information for all images currently stored in image buffer memory on the camera. The first line of the response contains the delete strategy which is common to all images. The following lines contain image information for each image in memory. Result codes vary for each sensor, so refer to the individual sensor for more information.

Example

```
? #IQ
$ 3
$ 219 16 1 12 2 -1 0 0
$ 218 16 1 13 1 -1 0 0
$ 217 -1 1 16 -1 0 0 0
% 0
?
```

The *Delete strategy* '3' means the oldest passing images will be deleted first, followed by the oldest failing images. There are three images in memory with image IDs 217, 218, and 219. Images 219 and 218 were inspected by product ID 1 and failed with error code 16. Their image handles are 12 and 13. Image 219 failed inspection because sensor 2 failed with error code -1, and image 218 failed inspection because sensor 1 failed with error code -1. Image 217 passed the product 1 inspection (-1 = PASS for product results) so the *First failing sensor ID* is -1 (non-existent) and the *First failing sensor result* is zero (0= PASS for sensor results).

See Also

[PQ - General product query](#)

[Pq - Specific product query](#)

[SQ - Sensor query](#)

[Sq - Sensor results query](#)

[YQ - Query system status](#)

[Yq - Query system type](#)

Script Management Commands

ZL List background scripts

ZL

No params No params

Parameters

No parameters

Response

(UINT8) Script ID_{1...n}

Internal ID of each script in the system file

(UINT8) State_{1...n}

Running/stopped state of the script

0: Stopped

2: Running

< 0: Compilation error

(UINT8) Start at powerup_{1...n}

Whether the script is set to automatically start when the system powers up

1: True

0: False

(UINT8) Save state_{1...n}

Whether the script is saved to flash

1: Saved

0: Not saved

(STRING) Name

Script name

Remarks

This command lists the parameters of each background script in the system file. Each script's parameters appear on a separate line. Scripts can reside in RAM, meaning they have been created but are not yet saved to flash, so check the save state to determine its persistence.

Example

```
? #ZL
$ 0 2 1 1 "NewScript"
$ 1 0 0 0 "NewScript2"
% 0
?
```

This system file contains two scripts named NewScript and NewScript2. NewScript (ID 0) is running (2), will start at powerup (1) and is saved to flash (1). NewScript2 (ID 1) is not running, saved, or set to run at powerup.

See Also

[ZN - Start/stop background script by name](#)

[ZR - Start/stop background script by ID](#)

ZN Start/stop background script by name

ZN <Name> <State>	
Name	String
State	UINT8

Parameters

Name

Name of the background script to be started/stopped, surrounded in quotes

State

Start/stop state to be set

Response

No response

Remarks

This command allows you to start and stop background scripts by passing the script name and a switch that determines whether it should be started or stopped.

Example

Start script "NewScript"

```
? #ZN "NewScript" 1
```

```
% 0
```

```
?
```

Stop script "NewScript"

```
? #ZN "NewScript" 0
```

```
% 0
```

```
?
```

See Also

[ZL - List background scripts](#)

[ZR - Start/stop background script by ID](#)

ZR Start/stop background script by ID

SP <ScriptID> <State>

ScriptID	UINT8
----------	-------

State	UINT8
-------	-------

Parameters

ScriptID

Internal ID of the script

State

Start/stop state to be set

Response

No response

Remarks

This command allows you to start or stop scripts by passing their internal ID and a switch that determines whether the script should be started or stopped. The internal ID can be listed using the #ZL command.

Example

Start script ID = 0

```
? #ZR 0 1
```

```
% 0
```

```
?
```

Stop script ID = 1

```
? #ZR 1 0
```

```
% 0
```

```
?
```

See Also

[ZL - List background scripts](#)

[ZN - Start/stop background script by name](#)

Product Management Commands

PA Add new product

PA <Name>	
Name	String

Parameters

Name

New name of the product to be added

Response

(INT16) ID

Internal product ID assigned to the product

(String) Name

Name assigned to the script. The script is auto-named if no name was specified as a parameter

Remarks

This command creates a new product and assigns it the specified name. The name does not need to be in quotes. If no name is specified the product is automatically given a new name. The script is also given an internal product ID that uniquely identifies each product. The product ID is not necessarily the same as the product digital ID used to switch products using the virtual I/O signals ProductSelect and Product ID Bit n.

Example

```
? #PA HoleCount
$ 3 " HoleCount"
% 0
? #PA Roundness Check
$ 4 " Roundness Check"
% 0
? #PA
$ 5 " Product3"
% 0
?
```

You can see that all the text entered after #PA is used as the product name. The new product is assigned the next available product ID and the assigned name is reported back in the response. If you allow the product to be auto-named by leaving the name parameter empty, the name is built using "Product" + n where n is the next available product index.

See Also

[PD - Delete product](#)

[SA - Add sensor](#)

PD Delete product from RAM and Flash

PD <ProductID>	
ProductID	INT16

Parameters

ProductID

Internal ID of the product to be deleted

Response

No response

Remarks

This command deletes a product and its sensors from RAM and Flash memory.

Example

Delete product 3 from memory

```
? #PD 3
```

```
% 0
```

```
?
```

See Also

[PA - Add product](#)

[PQ - General product query](#)

PF Save product to flash

PFS <ProductID>	
ProductID	INT16

Parameters

ProductID

Internal ID of product to be saved

Response

No response

Remarks

This command will save the indicated product to flash memory. All product IDs can be listed using the #PQ command. If the *ID* parameter is left empty, all products will be saved to flash. Flash is permanent memory, as opposed to RAM which 'forgets' its contents when power is cycled. Saving products to flash is necessary if you want them to persist after a power cycle.

Example

Save product with ID = 3

```
? #PFS 3
% 0
?
```

Save all products to flash

```
? # PFS
% 0
?
```

See Also

[PQ - General product query](#)

[Yf - Query flash file system](#)

PI Select inspection product

SP <ProductID>	
ProductID	INT16

Parameters

Product ID

Internal ID of the product to be set as the inspection product

Response

No response

Remarks

This command allows you to select which product is used for inspections. This is the product that the image is inspected with and its results determine the pass/fail result. This is not necessarily the product being displayed at any given terminal, but rather the product the camera is using to determine all outputs when inspections are running.

Example

Select product with ID = 3 as the inspection product

```
? #PI 3
```

```
% 0
```

```
?
```

See Also

[PQ - General product query](#)

[YI - Trigger inspection](#)

PN Rename product

PN <ProductID > <Name>

ProductID	INT16
Name	String

Parameters

Product ID

Internal ID of the product to be renamed

Name

New name to be assigned to the product

Response

No response

Remarks

This command allows you to rename an existing product. The name does not need to be quoted.

Example

Rename product with ID = 1 to "Measure Width"

```
? #PN 1 Measure Width
```

```
% 0
```

```
?
```

See Also

[PQ - General product query](#)

[Pq - Specific product query](#)

PQ General product query

PQ

No Params

No Params

Parameters

No parameters

Response

System-wide product information

(INT16) Selected product ID – Internal ID of the product selected for UI transactions

(INT16) Inspection product ID – Internal ID of the product being used for inspections

(INT16) Power on product ID – Internal ID of the product that is selected for inspections at power up

Product specific information

(INT16) Product ID_{1...n} – Internal ID of each product in the system

(STRING) Name_{1...n} – Name of the corresponding product

Remarks

This command queries the system for its products, their names, their IDs and their functions. Use the IDs obtained from this query as the *ProductID* parameter for other commands.

Example

```
? #PQ
$ 2 1 -1
$ 1 MeasureLargeStamp
$ 2 MeasureSmallStamp
$ 3 DetermineStampSize
% 0
?
```

This system has 3 products named 'MeasureLargeStamp', 'MeasureSmallStamp', and 'DetermineStampSize', with product IDs 1, 2 and 3 respectively. Product 2 is the currently selected product that is being worked on in the UI. Product 1 is the product selected for inspection on the camera. (-1) indicates that no product is selected to be used when the camera is powered up, so the user must manually select an inspection product when the camera is powered up.

See Also

[Pq - Specific product query](#)

[SQ - Sensor query](#)

[Sq - Sensor results query](#)

[YQ - Query system status](#)

[Yq - Query system type](#)

Pq Specific product query

Pq <ProductID>	
ProductID	Int16

Parameters

ProductID

Internal ID of the product to be queried

Response

(UINT32) *Pass count*

Number of passing inspections for this product

(UINT32) *Fail count*

Number of failing inspections for this product

(INT8) *Latest result*

Result of the last inspection

Remarks

This command queries a specific product for its inspection results. The *ProductID* can be obtained using the #PQ command. See the result codes table for details on the *Latest result*.

Example

```
? #PQ
$ 2 1 -1
$ 1 MeasureLargeStamp
$ 2 MeasureSmallStamp
$ 3 DetermineStampSize
% 0
? #Pq 2
$ 22653 177 -1
% 0
?
```

In this example the system is queried for its products and then using the #Pq command product 2 is queried for its inspection results. It has had 22,653 passing inspections, 177 failed inspections, and the last inspection passed with result code -1.

See Also

[PQ - General product query](#)

[SQ - Sensor query](#)

[Sq - Sensor results query](#)

[YQ - Query system status](#)

[Yq - Query system type](#)

PR Reset product results

PR <ProductID>	
ProductID	Int16

Parameters

ProductID

Internal ID of the product to be reset

Response

No response

Remarks

This command resets the inspection results for pass, fail and last result.

Example

```
? #Pq 2
$ 22653 177 -1
% 0
? #PR 2
% 0
? #Pq 2
$ 0 0 -1
% 0
?
```

In this example product 2 is queried for its inspection results which are 22653, 177 and -1 passing, failing and latest result respectively. Then the product 2 results are cleared by sending the #PR 2, and re-queried to show that they have been cleared to their defaults. The default values are zero accept for latest result which defaults to -1 (pass).

See Also

[PQ - General product query](#)

[Pq - Specific product query](#)

PS Select product for sensor command access by product ID

PS <ProductID>	
ProductID	Int16

Parameters

ProductID

Internal ID of the product to be selected

Response

No response

Remarks

This command tells the camera to which product sensor commands will be sent. Each connection must specify this before any sensor commands are sent. Obtain the *ProductID* using the #PQ command.

Example

```
? #PQ
$ 1 1 -1
$ 1 GaugeWidth
% 0
? #PS 1
% 0
?
```

This example queries the system for its products and then selects product 1 for sensor command access using #PS 1.

See Also

[PQ - General product query](#)

[PSI - Product select by digital ID](#)

[PSN - Product select by name](#)

PSI Select product for sensor command access by digital ID

PSI <DigitalID>	
DigitalID	Int16

Parameters

DigitalID

ID assigned to the product for digital product selection in the Product Manager

Response

No response

Remarks

This command is the same as #PS accept the parameter passed is the digital ID instead of the internal product ID. The internal product ID is the ID automatically assigned to each product when it is created. The digital ID used here is a user assigned ID that allows it to be digitally selected using physical I/O inputs. Digital IDs can be managed in the Intellect Product Manager.

Example

```
? #PSI 3
```

```
% 0
```

```
?
```

This example sets the selected product using the digital ID 3 that was user assigned in Intellect's Product Manager.

See Also

[PS - Product select by product ID](#)

[PSN - Product select by name](#)

PSN Select product for sensor command access by name

PSN <Name>	
Name	STRING

Parameters

Name

The name of the product to be selected

Response

No response

Remarks

This command is the same as #PS accept the parameter passed is the product name instead of the internal product ID. The name is either the auto-assigned name or that to which the user renamed it. Obtain product names in the Intellect UI or by sending the #PQ command.

Example

```
? #PQ
$ 1 1 -1
$ 1 GaugeWidth
$ 2 CountEdgeCracks
% 0
#PSN GaugeWidth
% 0
?
```

This example queries the system for its products and then selects the product named GaugeWidth for sensor command access. Notice that no quotes are necessary around the name.

See Also

[PS - Product select by product ID](#)

[PSI - Product select by digital ID](#)

Register Access Commands

Rq Register query

RQ <Register> <TypeFlag> <Block>	
Register	UINT16
TypeFlag	Char
Block	UINT16

Parameters

Register

Starting register number to be queried

TypeFlag

Data type to be queried, determines length of block to be read

Block (optional)

Number of registers to query in a block of single bytes

Response

(Type)Register data

Data contained in registers

Remarks

This command queries the specified starting register for a specific data type *OR* a block of single byte registers. *Register* indicates the starting register of the query. The data type determines the number of blocks read and how they are interpreted according to the following table.

Type flag	Type	Data length
B	UINT8	1
S	UINT16	2
I	UINT32	4
F	FLOAT32	4
D	DOUBLE	8
L	UINT64	8
S	String	Null terminated
K	Block as UINT8s	Block length

All but the last are discrete data types. The optional *Block* parameter can be left out for this case. K signifies that you want to read a continuous block of bytes so the optional *Block* parameter must be included to indicate the block length to be read. In this case the response will contain an array of the bytes read from memory.

Notice that there are no signed data types. All data is read as an unsigned number. To convert values stored as negative numbers, simply subtract the read value from the maximum value of that data type and then add 1.

Example

```
? #Rq100I
```

```
$ 1234  
% 0  
#Rq100k4  
$ 0 0 4 210  
% 0
```

This example queries register 100 for an integer value, which is 4 bytes in length. The value stored there is 1234. Then register 100 is queried as a 4 byte block. The 4 registers are returned as individual bytes 0 0 4 210.

See Also

[Rs - Register set](#)

Rs Register set

Rs <Register> <TypeFlag> <Value>

Register	UINT16
TypeFlag	Char
Value	TypeFlag dependent

Parameters

Register

Starting register number to be set

TypeFlag

Data type to be set, determines length of block to be set

Value

Data value to be written to registers

Response

No response

Remarks

This command allows you to set specific registers with a specific data type. *Register* is the starting register of the data to be written. The data length is determined by the data type as specified in the data type table on the previous page. *TypeFlag* corresponds to the chosen data type to be written.

Example

```
? #Rs100F1.23
```

```
% 0
```

```
? #Rq100F
```

```
$ 1.23% 0
```

```
?
```

See Also

[Rq - Register query](#)

Sensor Management Commands

SA Add new sensor

SA <Module>	
Module	INT16
Name	STRING

Parameters

Module

Type of sensor to be added

Name

The name of the sensor

Response

Sensor ID

The internal ID assigned to the sensor

Name

The name given to the sensor

Remarks

This command allows you to add a new sensor and name it. If no name is specified, the sensor is auto named. The sensor type is specified by a module ID. Each type of sensor has a unique module ID. This can be determined by issuing the #SM command, which lists all installed sensor modules. Because not all cameras support the same sensors, installed modules will differ.

After adding as sensor you must define its position and shape or else it will not have a presence in the image nor will its parameters be accessible through the Intellect UI. See the #SP function for details on setting sensor position and shape.

Example

```
? # PS 1
% 0
? #SA 22 filter1
$ 3 filter 1
% 0
```

This example selects product with ID=1 and then creates a preprocessing filter sensor (module = 22) with the name 'filter1'. It is given a sensor ID = 3.

See Also

[SM - List sensor modules](#)

[PS - Product select by product ID](#)

[PSI - Product select by digital ID](#)

[PSN - Product select by name](#)

SD Delete sensor

SD <SensorID>	
SensorID	INT16

Parameters

SensorID
ID of the sensor to be deleted

Response

No response

Remarks

This command allows you to delete sensors. Obtain the sensor's ID by sending the #SQ command.

Example

```
? #PQ
$ 1 1 -1
$ 1 Product1
% 0
? #PS 1
% 0
? #SQ
$ 1 22 Sensor1
$ 2 51 Sensor2
% 0
? #SD 2
% 0
?
```

This example first queries the system for products (#PQ), selects Product1 (#PS), queries Product1's sensors (#PQ), and finally deletes the sensor (#PD) with ID = 2 based on the sensor IDs obtained from the sensor query on Product1.

See Also

[PQ - General product query](#)

[SQ - Sensor query](#)

[PS - Product select by product ID](#)

[PSI - Product select by digital ID](#)

[PSN - Product select by name](#)

SI Change sensor ID

```
SI <SensorID> <NewID>
```

SensorID	INT16
----------	-------

NewID	INT16
-------	-------

Parameters

SensorID

ID of the sensor to edit

NewID

ID to be assigned to the sensor

Response

No response

Remarks

This command allows you to change a sensor's ID. You may want to do this if you need to iterate through a group of sensors that can be accessed by an incremented counter.

Example

```
? #PQ
$ 1 1 -1
$ 1 Product1
% 0
? #PS 1
% 0
? #SQ
$ 1 22 Sensor1
$ 2 51 Sensor2
% 0
? #SI 2 5
% 0
? #SQ
$ 1 22 Sensor1
$ 5 51 Sensor2
% 0
```

This example illustrates before and after changing a sensor's ID. After selecting the product and querying its sensors (#SQ), Sensor2 has *SensorID* = 2. After the #SI command, Sensor2 has *SensorID* = 5.

See Also

[PQ - General product query](#)

[PS - Product select by product ID](#)

[SQ - Sensor query](#)

SM List installed sensor modules

SM <no params>

No params No params

Parameters

No parameters

Response

(INT16) *Number of installed modules*

Number of modules installed on the camera

(INT16) *ModuleID_{1...n}*

Module ID of each sensor type

(INT16) *Version_{1...n}*

Version information for the sensor

(STRING) *Name_{1...n}*

Sensor module name

Remarks

This command lists all the sensor modules installed on a camera. Use the associated ID to reference a sensor type when creating new sensors by command. Modules vary depending on the camera being used because different cameras support different modules. For instance, the spectral camera has a special Spectrograph sensor that no other camera has, but it does not have the majority of the sensors other cameras have. The following table contains the full list of modules.

ModuleID	Module Name
21	Bead Tracking Sensor
71	Spectral Sensor Module
19	Color Sensor Module
17	Warping Sensor
24	Filters Sensor
22	Defect Detection
20	Color Monitoring Module
21	Counting Line Sensor
50	2D Reader Sensor
41	Caliper Line Sensors
63	New Positioning Sensors
40	2D Counting Sensor
51	2D Position Sensor
42	Edge Line Caliper Sensor
64	Edge Line Find Sensor
43	The Tool Sensors
9	Econ Sensor
13	Statistics Sensors
65	Math Sensors
5	Datum Sensor
62	One-D Bar Code
11	Bead Tracking Sensor

See Also

[PQ - General product query](#)

[PS - Product select by product ID](#)

[SQ - Sensor query](#)

[SA - Add sensor](#)

SN Rename sensor

SN <SensorID> <Name>

SensorID	INT16
----------	-------

Name	STRING
------	--------

Parameters

Sensor ID

ID of the sensor to be renamed

Name

New name for the sensor to be renamed

Response

No response

Remarks

This command allows you to rename existing sensors.

Example

```
? #PQ
$ 1 1 -1
$ 1 Product1
% 0
? #PS 1
% 0
? #SQ
$ 1 22 Sensor1
$ 2 51 Sensor2
% 0
? #SN 1 Frank
% 0
? #SQ
$ 1 22 Frank
$ 5 51 Sensor2
% 0
```

This example shows before and after changing the sensor name.

See Also

[PQ - General product query](#)

[PS - Product select by product ID](#)

[SQ - Sensor query](#)

[Sensor Name](#)

Sm Set match string for all reader sensors

Sm <Match string>	
Match string	STRING

Parameters

Match string

String used as pass/fail criteria for reader sensors

Response

No response

Remarks

This command allows you to set the match string for all the reader sensors at once. Reader sensors include 1D, 2D, and OCR sensors.

Example

```
? #Sm A12b3456
% 0
?
```

See Also

[2D Reader](#)
[OCR](#)

SP Set sensor specific parameters

SP <SensorID> <ParameterID> <Data>

SensorID	INT16
ParameterID	UINT8
Data	Sensor specific

Parameters

SensorID

Internal ID of the sensor

ParameterID

ID of the parameter to be set

Data

Data associated with the provided *ParameterID*

Response

No response

Remarks

SP, or setparams, is probably the most widely used command. It allows the user to set nearly any parameter for a sensor. This means there is lots of data that can be associated with this command and thus there is no standard format for usage. For instance SP can be used to set a sensor's position, shape, threshold, pass/fail criteria and match string. There are many different sensor shapes. Not all sensors have a user definable threshold. Only reader sensors have a match string. The pass/fail criteria are different for nearly every sensor. Clearly it is difficult to define the parameters for using this function since it must be useful for all these occasions. Consequently the user must understand the general usage of SP and how to obtain the data and formats to use with it.

The best way to find out what parameters you can set with SP is to look at the results of a sensor query (#SQ). The sensor query results display both the available parameter IDs and their associated data in the format that needs to be used when setting these parameters using SP.

Example

```
?#SP 3 0 7 1 -1 2544.092312 0 0 1 0 -1 0 0 1 1 1 1 1 50 0 0 5  
354 180 424 180 424 244 354 244 354 180 100 100
```

This example shows how to use SP to set the position and shape of a blob counting sensor. The leading data '3 0' is the *SensorID* and *ParameterID* (0 is the ParameterID for shape and position data). 7 is the parameter for a double rectangle. The rest of the numbers are all the data that goes with a double rectangle. The second line is all position/shape data. The blob sensor is a double rectangle. The learning rectangle is defined by the first 10 values which define 5 points (the last is repeated). The final two values define the search rectangle as a horizontal and vertical offset from the learn rectangle.

To change the position, most of this data should be copied and pasted without change. By changing only the second line, the position data, and sending that with the SP command, the blob sensor's position can be changed.

See Also

[Determining a Sensor's Parameters](#)

[Common Sensor Parameters](#)

[Sensor Specific Parameters](#)

SQ Sensor query

SQ <SensorID>	
SensorID	INT16

Parameters

Sensor ID (optional)
Internal ID of the sensor to be queried

Response

Without *SensorID*
(INT16) *SensorID*_{1...n}
Internal ID of each sensor in the currently selected product
(INT16) *ModuleID*_{1...n}
Sensor module ID of each sensor in the currently selected product
(STRING) *Name*_{1...n}
Name of each sensor in the currently selected product

With *SensorID*
(INT16) *ParameterID*_{1...n}
ID that indicates which parameter's data is listed on that line
*Data*_{1...n}
Parameter data associated with specific ID

Remarks

This command allows you to query a sensor for its parameter data and format. This data can then be used partially or as a template for sending other commands like #SP. Parameter data can be anything associated with the sensor's inspection parameters, pass/fail criteria, shape, position/image reference, name, description, scale data, etc... Each parameter is headed by a '\$' and a parameter ID that indicates what parameter the following data represents. The list of possible parameter IDs is set, but not all sensors use each parameter so the response will differ for each sensor and it will also differ depending on the state of that particular sensor.

This command can be sent with or without a *SensorID* parameter. If sent with a *SensorID* the response will be specific to that sensor. If sent without a *SensorID* parameter the response will contain a list of the *sensorIDs*, modules, and names associated with the currently selected product. A module is a sensor type available on a specific camera.

Example

```
?#PQ
$ 3 4 -1
$ 4 Product4
% 0
?#PS 4
% 0
?#SQ
$ 1 43 Sensor1
% 0
?#SQ 1
$ 240 "Sensor1"
```

```

$ 241 1 0
$ 242 ""
$ 243 1
$ 245 0 1 -1 "" 1
$ 246 0 1 0 0 -1 "" 1
$ 251 3 0
$ 253 2000 1
$ 255 -1
$ 0 7 1 -1 2544.092312 167.529205322266 157.104873657227 -0.2468383163213
73 -0.969056665897369 -1 0 1 0 1 0 1 1 1 50 0 0 5 35.854335784912
1 170.014175415039 239.854309082031 150.014175415039 260.145660400391 356.9
85809326172 56.1456756591797 376.985809326172 35.8543357849121 170.014175415
039
$ 1 0
$ 2 5 0 0 -500 0 0 2 2
$ 6 1500 -1 -1 -100 -100 -1
$ 7 1500 -1 -1 -1
% 0

```

This example shows the query results for a Positioning Line Fit sensor and the necessary commands to first query and select the sensor's product.

See Also

[PQ - General product query](#)

[PS - Product select by product ID](#)

[Determining a Sensor's Parameters](#)

Sq Sensor inspection results query

Sq <SensorID>	
SensorID	INT16

Parameters

SensorID
Internal ID of sensor to be queried

Response

(INT16) *SensorID*
Internal ID of the queried sensor

(UINT4) *Enabled flag*
Flag determining whether the sensor is enabled for inspection
1: Enabled
0: Disabled

(STRING) *Name*
Sensor name

(INT8) *Latest result*
Pass/Fail result of the latest inspection

(UINT32) *Pass count*
Number of passing inspections since last power cycle

(UINT32) *Fail count*
Number of failing inspections since last power cycle

(INT8) *Specific last result*
Result code indicating cause for failure if failure occurred

Remarks

This command allows you to query a sensor for only its inspection result data. This is useful for gathering statistics on a specific sensor. The pass/fail counts are accumulated only while inspections are running and are reset when the camera is power cycled or when the inspection statistics are manually reset.

Example

```
? #Sq 1
$ 1 1 Sensor1
$ 16 11 4 -3
% 0
?
```

This example shows the results of a sensor results query on a positioning edge fit sensor. The *SensorID* is 1 and the sensor is enabled for inspection. Error 16 indicates that the sensor failed the last inspection. The specific result -3 indicates that the failure cause was the maximum offset being exceeded. There have been 11 passing inspections and 4 failed inspections.

See Also

[PQ - General product query](#)
[PS - Product select by product ID](#)
[SQ - Sensor query](#)

[Pq - Specific product query](#)
[li - Query inspection statistics](#)
[liz - Reset inspection statistics](#)

SR Reset sensor results

SR <SensorID> <ProductID>	
SensorID	INT16
ProductID	INT16

Parameters

SensorID

Internal ID of the sensor whose results are to be reset

ProductID (optional)

Internal ID of the product containing the target sensor

Response

No response

Remarks

This command allows you to reset sensor pass/fail statistics. The *ProductID* parameter is optional. If it is left off, the command will apply to the currently selected product. If no sensor with that ID exists in the currently selected product the command will reply with a %1 and do nothing.

Example

```
? #Sq 1
$ 1 1 Sensor1
$ 16 11 4 -3
% 0
?
? #SR 1
% 0
? #Sq 1
$ 1 1 Sensor1
$ -1 0 0 0
% 0
?
```

This example shows the results of a sensor results query before and after resetting the inspection statistics for that sensor.

See Also

[PQ - General product query](#)

[PS - Product select by product ID](#)

[SQ - Sensor query](#)

[Sq - Sensor results query](#)

[Iiz - Reset inspection statistics](#)

[PR - Reset product results](#)

System Management Commands

Y Enable/Disable inspections

Y <InspectionFlag>

InspectionFlag	Char
----------------	------

Parameters

(char) *InspectionFlag*

+/- character determining whether inspections are enabled or disabled

+: Enabled

-: Disabled

Response

No response

Remarks

This command allows you to turn inspections on or off. Only when inspections are running is data actually processed for output to I/O and registers. Any other time images will be acquired and data may be processed for output to user interfaces only, but no states will change in the camera.

Example

```
? #Y+
% 0
?
```

This example shows how to turn inspections on.

See Also

[YI - Trigger inspection](#)
[Perform Inspection](#)

Yf Query flash file system information

YF

No params

No params

Parameters

No params

Response

(UINT32)Flash mount result

Result of mounting the flash file system

(INT32) Volume ID

ID of the flash volume

(UINT32)Total user space

Total space available to the user

(UINT32) Space available

Total space not yet used

(UINT32) Space used

Total space already used

(UINT32) Space in bad sectors

Total unusable space due to bad memory sectors

(UINT32) System reserved space

Space reserved for system use

(INT32) Application saved flag

Flag indicating whether the current UI application is saved to flash

(UINT32) Total flash

Total flash installed in the system

(UINT32) Space reserved for boot code

Total space reserved for boot code storage

(UINT32) Space reserved for application

Total space reserved for application storage

(STRING) File-system version string

Version of the currently installed flash file system

Remarks

This command allows you to query the flash file system for memory usage. All memory sizes are reported in bytes.

Example

```
? #Yf
```

```
$ 0 0 4849664 4845568 4096 0 393216 1 10485760 0 0 HCC_EFFS_STD ver:1.46
```

See Also

[Yq - Query system type](#)

YI Trigger an inspection

YI	
No params	No params

Parameters

No params

Response

No params

Remarks

This command allows you to trigger an inspection if the camera is in external trigger mode. Triggering an inspection causes the camera to acquire 1 image and perform an inspection on the newly acquired image if inspections are on.

Example

```
? #YI  
% 0  
?
```

This example demonstrates how to trigger a new inspection.

See Also

[Y - Enable/disable inspections](#)

[YP - Set trigger source](#)

[Yp - Set trigger mode](#)

[YQ - Query system status](#)

YN Get/set image sensor name

YN <Name>

Name	STRING
------	--------

Parameters

Name (optional)

New name if setting the camera name

Response

(STRING) Name

Name assigned to the image sensor

Remarks

This command allows you to query the image sensor for its assigned name or assign a new name. If the *Name* parameter is left off, the command queries and returns the current camera name.

Example

```
? #YN
$ DVT Smart Camera
% 0
?#YN DVT Intelligent Camera
$ DVT Intelligent Camera
% 0
?
```

This example demonstrates how to query the camera for its name and then set a new name.

See Also

[YV - Query firmware](#)

[YQ - Query system status](#)

[Yq - Query system type](#)

[PQ - General product query](#)

[SQ - Sensor query](#)

YP Set internal acquisition trigger mode

YP <SourceFlag>	
SourceFlag	INT32

Parameters

SourceFlag

Flag determining the trigger source

-1: acquire new image automatically after old image is processed

0: acquire new image automatically immediately after old image is acquired

Value: acquire image automatically every *value milliseconds*

Response

No response

Remarks

This command allows you to set the trigger source when the trigger mode is set to internal. The image will be acquired at a regular interval of *value* milliseconds if a time is specified. If the camera is not first set to internal trigger mode, nothing will happen.

Example

```
? # Yp+  
? #YP 250  
% 0  
?
```

This example sets the camera to internal trigger and sets the trigger rate to 250ms.

See Also

[Y - Enable/disable inspections](#)

[Yp - Set trigger mode](#)

[YQ - Query system status](#)

Yp Set trigger source

Yp <ModeFlag>

ModeFlag	Char
----------	------

Parameters

ModeFlag

Flag indicating whether the camera will trigger internally or from an external source

+: Internal

-: External

Response

No response

Remarks

This command allows you to change the trigger mode of the camera. Internal trigger mode causes the camera to be triggered automatically according to the trigger source setting. External trigger mode causes the camera to wait for an external trigger input.

Example

See previous example

See Also

[Y - Enable/disable inspections](#)

[YP - Set trigger source](#)

[YQ - Query system status](#)

YQ Query system status

YQ

No params

No params

Parameters

No parameters

Response

(char) Inspecting state

Flag indicating whether camera is performing inspections

+: Inspecting

-: Not inspecting

(char) Trigger mode

Flag indicating whether the camera is triggering from an internal or external source

+: Internal

-: External

(char) Digital selection state

Flag indicating whether the camera has digital selection enabled

+: Enabled

-: Disabled

(INT32) Internal trigger period

Trigger period in milliseconds if the camera is set to trigger internally with a period

(UINT8) CPU load

CPU load from 0-100% over the last 1024 milliseconds

(INT32) Unused flash

Amount of unused flash in bytes

(UINT32) System clock

Current system clock value in milliseconds

(INT16) CCD ID

ID indicating the CCD type in the camera

0: 630, 530, 540, 550

1: 510, 520

2: 542c, 552c

3: 544, 554, 544c, 554c

7: LS

(INT16) Inspection product ID

Internal ID of the current inspection product

Remarks

This command allows you to query various system states. This can be particularly useful for monitoring the CPU load and retrieving the system clock time for data logging.

Example

```
? #YQ
$ +++ 343 0 9942528 1190036074 7 3
```

This example shows the result of querying the system with a #YQ. The '+++' values indicate that inspections are running on product 3 (last value), the camera is triggering internally and digital selection is enabled. Digital selection is enabled by default in Intellect. This camera is a

LineScan as indicated by the 7 and it is set to trigger every 343ms. The recent CPU load is 0% so the camera probably isn't doing anything at the moment. There are about 10Mb of unused flash and the system clock has been running for 1190036074ms or 13days 18hours 33min 56s 74ms, unless the clock has already rolled over.

See Also

[Y - Enable/disable inspections](#)

[YP - Set trigger source](#)

[Yp - Set trigger mode](#)

[YN - Get/set image sensor name](#)

[Yq - Query system type](#)

Yq Query system type

Yq	
No params	No params

Parameters

No params

Response

(UINT8) CPU type

Processor type in the camera

3: Motorola MPC860 or compatible

4: Intel x86

5: Hitachi SH-4, SH7750

6: TI DSP

(UINT16) CPU clock

Clock speed of the installed processor in MHz

(UINT8) FPU present

Flag indicating whether a floating point unit is present

(UINT32) Installed RAM

Amount of RAM installed in the camera

(UINT32) Installed flash

Amount of flash installed in the camera

() Unused

Always 0

(UINT16) System type

Camera series

2: 630

3: 520 or 530

4: 54x

5: SmartReader

6: Intelligent Scanner (IS)

7: Spectral Camera (SC)

9: 55x or LineScan (LS)

10: 510

11: Extreme Scanner (XS)

(UINT16) Imager type

3: 640 x 480 gray scale (630)

4: 640 x 480 gray scale CCD (530, 540, 550, SC, IS or XS)

5: 640 x 480 gray scale CMOS (510 or 520)

8: 640 x 480 color CCD (542c or 552c)

11: 1280 x 1024 gray scale CCD (544 or 554)

12: 1280 x 1024 color CCD (544c or 554c)

13: 2048 x 1 gray scale CCD(LineScan)

Remarks

This command allows you to query the system type. Combine the last two values, *System type* and *Imager type* to determine the exact camera model. The amount of RAM and flash installed on a particular camera model can vary depending on when it was purchased.

Example

```
? #Yq  
$ 5 0 0 33554432 8388608 0 4 8
```

This example shows the results of system type query. This camera is using an Hitachi SH-4 processor, has 32Mb RAM and 8Mb flash. The 4 indicates that it is a 54x series camera and the 8 indicates it is a 542c or 552c, so this is a 542c camera.

See Also

[Yf - Query flash file system](#)

[YN - Get/set image sensor name](#)

[YQ - Query system status](#)

YR3742 Reset and reboot to diagnostics

YR3742
No params

No params

Parameters

No params

Response

No params

Remarks

This command allows you to put the camera in diagnostics mode without physically cycling power. There is no response or indication that this has worked other than the LED status light on the back of the camera which will blink red once per second. '3742' is an ASCII string, not a decimal number.

Diagnostics mode is used to reformat flash from the command prompt and to load firmware from Intellect. Diagnostics mode can however be entered from Intellect and flash can be reformatted in Intellect without use of the command prompt. Additionally there is no command for loading firmware from the command prompt. Thus this command should only be used as a diagnostics and troubleshooting tool.

Example

```
? #YR3742
```

This example shows the command to reboot the camera to diagnostics mode. There is no prompt after issuing the command.

See Also

[No references](#)

YS1bdc Reformat and save application to flash

YS1bdc	
No params	No params

Parameters

No params

Response

No params

Remarks

This command allows you to reformat flash and save the application from RAM to flash. This command returns immediately with a % 0 and ? prompt if the command is accepted, but no indication of the reformat and save progress is given. Use the #Ys command to check the progress. '1bdc' is an ASCII string, not a hex string.

Example

This example shows how to issue the command to reformat flash and save the application in RAM to flash.

```
#YS1bdc
```

```
% 0
```

```
?
```

See Also

[Ys - Query format progress](#)

[PF - Save products](#)

Ys Query YS1bdc progress

YS1bdc

No params

No params

Parameters

No params

Response

(INT32) Stage

The current state of the reformat and save

0: Idle

1: Erasing old data from flash

2: Formatting volume

3: Saving application

4: Saving system settings

5: Flash re-format complete

6: Format error

(FLOAT64) Progress

Decimal number indicating the % completion of the current stage

Remarks

This command allows you to query the progress of the YS1bdc command. Because that command gives no feedback you can query the camera to see what stage it is in. The response contains the stage followed by the percent completion.

Example

```
#YS1bdc
% 0
?#Ys
$ 1 0.82931697600984
% 0
?#Ys
$ 3 0.897334018919131
% 0
!#Ys
$ 5 0
?
```

This example shows how to query the state of the flash reformat and save. After the #YS1bdc command is issued the state is checked by sending the #Ys command several times. The first query response indicates that the system is currently 82.9% complete erasing the old flash data. The second query response indicates that the system is currently 89.7% complete saving the application to flash. The third query response indicates that the process is complete with no errors.

See Also

[YS1bdc - Reformat and save](#)

YV Query firmware version

YV

No params No params

Parameters

No params

Response

(UINT8) Alpha/Beta flag

Flag indicating whether the firmware is released, alpha or beta

0: Release

1: Beta

2: Alpha

(UINT8) Major

Major firmware version

(UINT8) Minor

Minor firmware version

(UINT8) Refresh

Refresh firmware version

(UINT16) Build

Build firmware version

(UINT32) Build date

Date of build in decimal

()

Remarks

This command allows you to query the firmware version and build date. The build date is in decimal. Convert this number to hexadecimal to get a number that reads ddmmyyyy.

Example

```
?#YV
$ 1 1 2 0 4 688267269 0
% 0
?
```

This example shows how to query the system firmware version. This is version 1.2.0 Beta4 built on Jun 29, 2005 (688267269d → 29062005h).

See Also

[YN - Get/set image sensor name](#)

[Yq - Query system type](#)

[Yf - Query flash file system](#)

[YQ - Query system status](#)

7. Intellect 1.2 Command line Options

Command Description	Command Line Parameters																																
Auto Connect	/c <IP Address> <Port> Example: Intellect /c 127.0.0.1 3246 Comments: Port is optional. Use different ports if connecting to multiple cameras behind a firewall or router with a single IP address. The firewall or router must have port forwarding setup to forward the indicated port connection to the correct camera IP and port.																																
Download System File	-CreateSystemFile <IP Address> <File Path> Example: Intellect -CreateSystemFile 127.0.0.1 "c:\DVT\Backup.dvtisys" Comments: You must put parameters in quotes if they contain spaces.																																
Upload System File	-SendSystemFile <IP Address> <File Path> [<User>] [<Password>] [<Options>] Example: Intellect -SendSystemFile 192.168.1.157 "c:\dvt\test.dvtisys" "Administrator" "" M Comments: User name, password and option parameters are optional. You must put parameters in quotes if they contain spaces. Use "" if you want to specify options without a password or a user name Options are a string of zero or more characters as described below: Options: <table><tr><th>Settings</th><th>Append Char</th><th>Ignore Char</th><th>Overwrite Char</th></tr><tr><td>Ethernet Terminals</td><td>'E'</td><td>'e'</td><td>None</td></tr><tr><td>Modbus Masters</td><td>'M'</td><td>'m'</td><td>None</td></tr><tr><td>Background Scripts</td><td>'S'</td><td>'s'</td><td>None</td></tr><tr><td>Terminal Specs</td><td>N/A</td><td>'t'</td><td>None</td></tr><tr><td>Digital IO</td><td>N/A</td><td>'d'</td><td>None</td></tr><tr><td>User Access</td><td>N/A</td><td>'a'</td><td>None</td></tr><tr><td>Products</td><td>'P'</td><td>N/A</td><td>None</td></tr></table>	Settings	Append Char	Ignore Char	Overwrite Char	Ethernet Terminals	'E'	'e'	None	Modbus Masters	'M'	'm'	None	Background Scripts	'S'	's'	None	Terminal Specs	N/A	't'	None	Digital IO	N/A	'd'	None	User Access	N/A	'a'	None	Products	'P'	N/A	None
Settings	Append Char	Ignore Char	Overwrite Char																														
Ethernet Terminals	'E'	'e'	None																														
Modbus Masters	'M'	'm'	None																														
Background Scripts	'S'	's'	None																														
Terminal Specs	N/A	't'	None																														
Digital IO	N/A	'd'	None																														
User Access	N/A	'a'	None																														
Products	'P'	N/A	None																														
Default	If INI file is set to "Auto Connect", Intellect will try to connect to the last camera or emulator, otherwise Intellect will open with no connection.																																